TEL AVIV UNIVERSITY אוניברסיטת תל-אביב

The Raymond and Beverly Sackler Faculty of Exact Sciences

The Blavatnik School of Computer Science

Machine Learning Algorithms and Robustness

Thesis submitted for the degree of Doctor of Philosophy

by

Mariano Schain

This work was carried out under the supervision of

Professor Yishay Mansour

Submitted to the Senate of Tel Aviv University

January 2015

Abstract

The advent of the Internet introduced many business opportunities and related challenges, many of which can be framed as learning problems: browsing-users' behavior and preferences, bidding patterns, classification of user-generated content, and many more. Learning algorithms are based on a model of reality (the environment in which they operate and are tested), and their performance depends on the degree of agreement of their assumed model with reality. The Robustness of an algorithm is its sensitivity to discrepancies between the assumed model and reality. This research investigates key aspects of robustness as related to the following specific learning settings.

One instance of such needed robustness is modeling and optimization in multi-agent settings (in which a poor model of reality might lead to meager decisions). The first aim of this thesis in that context is building learning agents, in particular for the Trading Agent Competition (TAC) - A synthetic environment for evaluating trading agent's strategies. We base our trading agents on Machine-Learning *model-light* methods that tend to refrain from assuming a statistical model for the generation of observations, and as such reduce the risk of model mismatch and are expected to better tolerate adversarial environments. Such methods are usually also simpler to implement and are therefore more robust in that sense. We compare some of the fundamental qualities of our model-light agents to those of agents built using other techniques which may exploit the model-specific parameters and structure of the TAC simulated environment. By taking part in the TAC/Ad-Auctions competition (adding more parametric components to our agents throughout the years, eventually winning the 2013 competition) we are able to quantify the value of using parametric methods. We establish the robustness of model-light methods by showing the diminishing importance of modeling accuracy (showing that for our top performing agent, a perfect knowledge of the model parameters results in insignificant optimization improvements).

This thesis also considers robustness as a mean to assess the applicability to real-world settings of TAC-like competition insights regarding agent strategies and mechanisms. By changing the game environment (that is, the simulation parameters) of

the (unaware) competing agents, we demonstrate that top performing agents are (surprisingly) robust to changes in game setting. Our positive findings suggest that the (inherently simplistic) TAC-like games may be used to draw general conclusions about real-life situations. Finally, the thesis considers the Ad-Exchange setting of the Internet brand-advertising ecosystem. Taking the perspective of the Ad-Network (and the challenges it faces as it bids for display ads opportunities announced by an Ad-Exchange in order to carry out targeted advertising campaigns) we designed and implemented a new TAC game launched during the summer of 2014. The strategies employed by agents implemented for TAC-AdX competitions provide insights regarding winning strategies for this pivotal setting of the Internet's economy.

Another instance of robustness investigated in this thesis is the Domain Adaptation problem, in which the learning algorithm is expected to perform (generalize) well in one (target) domain but may only be trained using data from a significantly different (source) domain. Such situations abound in practice due to inherent limitations, cost considerations, or convenience (e.g., speech recognition, machine vision, sentiment analysis in natural language processing, and many other settings), hence the importance of quantifying the related performance penalty (through generalization bounds) and providing algorithmic methods to adapt existing algorithms to address the domain adaptation problem. For domain adaptation, we use the robust-optimization approach and algorithmic-robustness properties to derive related generalization bounds and design new domain-adaptation variants of classical machine learning algorithms (SVM) for classification and regression. Naturally, such algorithms and bounds depend on statistical properties of the discrepancies between the source and target domains (which can also be regarded as quantifying domain-specific prior knowledge regarding source-target similarity) and we introduce a related measure. Our approach is generic in the sense that it may be similarly applied to other Machine Learning algorithms, optionally using other statistical similarity measures.

Finally, the thesis considers Social Learning - methods to accomplish, through collaboration, a computation that depends on the aggregate of private information accessible to self interested agents. Specifically, we investigate sequential processes based on market scoring rules, where each agent updates the current state of the computation based on his private signal. Such settings (e.g., Prediction Markets, where options regarding the occurrence of a future event are traded, and the option's outstanding

price is the state of the common computation) may be regarded as parameter estimation platforms, where the resulting computation represents the *wisdom of the crowd* aggregate of the agents' private signals. Inspired by trusted recommendation chains, where private observations (e.g., regarding the quality of a product) are used to make one-on-one recommendations among trusted agents, we introduce a model of simple history independent agents (that is, agents of limited strategy space, having no access to past states) and provide an analysis of the performance of the resulting estimators pertaining to different equilibrium and socially optimal strategy profiles. The performance of those estimators is compared to that of the optimal (Bayes) estimator having access to the update history and private signals, thereby quantifying the penalty of history unavailability.

Contents

List of Tables

List of Figures

Chapter 1

Machine Learning - A Robustness Perspective

Learning, *"the acquisition of knowledge or skills through experience, study, or by being taught*[1] *"* is crucial for the survival and prosperity of humans, as individuals, as a society, and as a species. A baby learns to signal regarding his immediate necessities, a toddler learns to speak and understand a language, a boy learns to avoid danger, an adult learns to drive. The vast number of situations and the inherently non-deterministic nature suggests that learning is unavoidable. [2] We therefore learn, by example, by transfer.[3] Our long term and short term memory is constantly aggregated and reorganized to support the different needs and criteria (e.g., time, risk) of the myriad situations requiring decision making, be it intuitive or deeply thought upon. The constantly changing environment (due to our actions, statistical nature, or otherwise) must be considered as learning takes place. To some degree, what worked well in the past (or in a somewhat different setting) might not work as well when faced again. In that sense, to achieve its ultimate goal, learning should be *Robust* - durable, long-lasting, resilient, solid.

Machine Learning, the application of computational methods underlying experience-based decision making, has similar conceptual goals and requirements. Not all settings require machine learning. Sorting problems in a static environment, for example, have dedicated efficient algorithms, merely computer language translations of recipes that could be otherwise interpreted and executed by humans (if time, for

[1] Oxford Dictionary.
[2] Anyway, a codex mapping situations to the 'right' actions would be impractical.
[3] That is, associating and reflecting upon different, although conceptually similar situations.

example, was not a concern). In many settings, however, such algorithms are hard to devise. Vague relations between the representation of inputs (e.g., a picture's bitmap) to the concepts underlying the required output decisions (e.g., "Dog or Cat?"), inherent uncertainty in data, in the actual model, and in the environment in general, makes it extremely hard to design such recipe-like algorithms. This is where machine learning methods save the day, aiming at mimicking the generalization consequences of human learning (and sometimes the related underlying biological structures), considering and being robust to different underlying uncertainties.

Machine learning was successfully applied during the last few decades to enable autonomous decision making by robots, medical diagnosis, network optimization, and many other modern tasks. The recent prevalence of the Internet and the related scale and abundance of data from an exponentially growing number of sensors brings new opportunities for applying machine learning. Some, such as those used for search results, advertising, and content matching, already established as the cornerstone of the Internet economy, and others, such as recommendation systems based on semantic analysis, are promising to revolutionize our daily life experience through related services that become more and more personalized.

This work is mainly concerned with robustness aspects of machine learning methods. In the rest of this introductory chapter, a short overview of the essence of machine learning and some of the key characteristics of two of its disciplines, namely, supervised and on-line learning,[4] are reviewed. This is followed by a thematic exposition of the fundamental related methods and algorithms, as practices aimed at achieving robustness to the different uncertainties faced in the various settings. The chapter concludes with a short overview of the specific research reported in the three parts of this thesis.

1.1 Supervised Learning

This section briefly summarizes the key concepts and fundamental results in supervised learning. The reader is referred to [90] for a thorough exposition of the topic.

We are interested in learning a relation between the values of some input variables (from an *input* space X) and an output value (from an *output* space Y). Specifically, for a deterministic input-output relation, we are interested in approximating an unknown

[4] Other methods, less relevant to this thesis, mainly Clustering and Ranking are not covered.

function $f : X \to Y$. A setting in which the output space Y is discrete is referred to as *Classification*, alternatively, in a *Regression* setting, the output space is continuous.

Learning is based on a finite training set S of examples

$$S = \{(x_i, y_i) : x_i \in X, y_i = f(x_i) \in Y, i = 1 \dots m\} ,$$

assumed to be drawn i.i.d. from an (unknown) underlying distribution D over the input space, and the result (the output of the learning algorithm) is a *hypothesis* $h \in H$ from a predefined hypothesis class H. The resulting hypothesis may then be used to associate an *approximation* $\hat{y} = h(x)$ for $y = f(x)$ to any $x \in X$.

A loss function $l : Y \times Y \to \mathcal{R}_+$ is usually associated to the setting and is used to measure the performance of a learned hypothesis h, where

$$\mathcal{L}(h, (x, y)) \triangleq l(h(x), y) , \tag{1.1}$$

is the loss of h on an input-output pair (x, y). The overall performance of a hypothesis h is then defined as its expected loss, i.e.,

$$\mathcal{L}_D(h) \triangleq E_{(x,y)\sim D} l(h(x), y) .$$

This expected loss is also termed the *Risk* or the *Generalization Error* (since it measures the loss over the *general* input-output space, rather than on the training set) of the hypothesis h. For example, in the *Linear Regression* setting (over Euclidean input and output spaces $y \in \mathcal{R}$ and $x \in \mathcal{R}^n$, respectively) an underlying linear input-output relation is used and therefore the output of a learning algorithm in this setting is some $w \in \mathcal{R}^n$, characterizing a member of $H = \{h(x) = w^T x : w \in \mathcal{R}^n\}$.

A learning algorithm aims at finding a hypothesis h of small generalization error. This of course depends on the hypothesis class H available to the algorithm. The target and benchmark $\mathcal{L}^*_{D,H}$ for the performance of a supervised learning algorithm is the best achievable expected loss, i.e., by a hypothesis h^*_D of the class H,

$$\mathcal{L}^*_{D,H} \triangleq \min_{h \in H} \mathcal{L}_D(h) \triangleq \mathcal{L}_D(h^*_D) . \tag{1.2}$$

When the true underlying relation f is a member of H, then $h^*_D \in H$, and $\mathcal{L}^*_{D,H} = 0$. In the more general case, where f belongs to some class F different from H, and $f \notin H$

(this is the *unrealizable* setting), the optimal expected error might be non-zero.

In the *Probably Approximately Correct* (PAC) learning framework, a learning algorithm is required to output with high confidence[5] an approximately optimal hypothesis. More precisely, the output $h_A(S)$ of a learning algorithm A based on a training sample S is required to be *approximately correct* (being ϵ-far from optimal loss) with high probability (at least $1-\delta$, over the realization of S). Namely,

$$Pr_{S\sim D^m}[|\mathcal{L}_D(h_A(S)) - \mathcal{L}^*_{D,H}| \leq \epsilon] \geq 1-\delta \ , \tag{1.3}$$

for any underlying $f \in F$ and any distribution D. Such a learning algorithm A that runs in time polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$ is said to *efficiently learn* F.

A crucial step in the methodology of designing a supervised learning algorithm is the choice of the underlying input-output relation to model (that is, the selection of the features used to characterize the input and output space - thereby the class F of possible input-output relations) and the related hypothesis space H for the algorithm to chose its output from.

In practice, the choice of hypothesis class H is a *modeling* decision. In the linear regression setting for example, some $w \in \mathcal{R}^n$ characterizes each member of H, and the learning algorithm aims to find an optimal value for the parameter w. Alternatively, *non-parametric* (sometimes also designated *model-free*) learning algorithms do not assume an explicit underlying parametrization of the input-output relation. Such algorithms (e.g., SVM and Nearest Neighbor, presented later in this section) utilize hypothesis classes whose elements are explicit (possibly weighted) combinations of the training samples.

Now, the essence of a supervised learning algorithm is the method it uses, given the training set S, to choose a member h out of the underlying hypothesis class H. *Empirical Risk Minimization* (ERM) is a natural method, which outputs a hypothesis h^*_S that minimizes the empirical risk, i.e.,

$$\hat{\mathcal{L}}_S(h) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(h,(x_i,y_i)) \ .$$

[5]That is, with high probability, viewing the learning algorithm's output hypothesis as a random variable, over the realization of the sample set S and its own coin tosses.

That is,

$$h_{\mathrm{ERM}}(S) = h_S^* \triangleq \arg\min_{h \in H} \hat{\mathcal{L}}_S(h) \ . \tag{1.4}$$

Considering again, for example, linear regression, where the loss function used is the quadratic loss function $l(y_1, y_2) = (y_1 - y_2)^2$, the empirical loss to be minimized by ERM is

$$w_{\mathrm{ERM}} \triangleq \arg\min_{w \in \mathcal{R}^n} \frac{1}{m} \sum_{i=1}^{m} (w^T x_i - y_i)^2 \ . \tag{1.5}$$

The applicability of the ERM method for PAC-learning is theoretically justified by the following fundamental result, a uniform learning bound (that is, applicable to all members of the hypothesis class H) relating the empirical risk and generalization error of hypotheses from a finite class H.

Theorem 1 (Th. 2.2 of [90]). *Let H be a finite hypothesis class and let S be a finite set of m examples drawn i.i.d. over D. Then, for any $\delta > 0$, with probability at least $1 - \delta$ (over the realization of $S \sim D^m$) the following holds for all $h \in H$*

$$\mathcal{L}_D(h) \leq \hat{\mathcal{L}}_S(h) + \sqrt{\frac{\log |H| + \log \frac{2}{\delta}}{2m}} \ . \tag{1.6}$$

The bound (1.6) of Theorem 1 can be further improved when the *realizable* (i.e., $f \in H$) setting is assumed. Also, similar bounds exist for the case of infinite (and even uncountable) hypothesis classes H. In that case, the $\log |H|$ in the bound (1.6) is replaced by the VC-Dimension (see Chapter 3 of [90]), a combinatorial measure for the effective size of H.

Now, consider again the ERM algorithm. For any $\epsilon > 0$, by taking a sample set S of size m such that the rightmost term of the bound (1.6) is smaller than $\frac{\epsilon}{2}$ (that is, $(m > \frac{2}{\epsilon^2} \log \frac{2|H|}{\delta})$ we have with probability at least $1 - \delta$ (by applying Theorem 1 twice and by the definition of ERM)

$$\mathcal{L}_D(h_S^*) \leq \hat{\mathcal{L}}_S(h_S^*) + \frac{\epsilon}{2} \leq \hat{\mathcal{L}}_S(h_D^*) + \frac{\epsilon}{2} \leq \mathcal{L}_D(h_D^*) + \epsilon \ .$$

This establishes that with probability at least $1 - \delta$ the ERM algorithm results in a hypothesis with generalization error at most ϵ far from optimal, as required by (1.3). Furthermore, an efficient number (polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$) of training samples is needed.

In many settings and for a variety of loss functions, however, solving the optimiza-

tion problem (1.4) might be infeasible. For example, in classification settings, where the 0-1 loss[6] is a natural choice, (1.4) is an intractable combinatorial problem in many natural cases. A common practice in such situations (as employed, for example, by the SVM algorithm detailed next) is to replace the optimization objective (that is, the loss function) with an upper bounding convex surrogate, rendering the optimization problem tractable.[7] The *Hinge* loss

$$l_{\text{Hinge}}(h_{w,b},(x,y)) = \max\{1 - y(w^T x + b), 0\} \ ,$$

is one such popular surrogate loss for classification settings using the 0-1 loss and linear separators as hypotheses (i.e., where each member $h_{w,b}$ of the hypothesis class H is a linear separator $h_{w,b}(x) = 1_{[w^T x+b>0]}$).

The ERM method relies on the training set S being representative of the (unknown) underlying distribution D, over which the algorithm's output is to be tested. A too small sample set, compared to the complexity of H, might result in *over-fitting*, where a returned hypothesis has low empirical risk $\hat{\mathcal{L}}_S(h)$ and high true risk $\mathcal{L}_D(h)$. Indeed, by (1.6), the larger the complexity[8] of the hypothesis class H, the larger the potential gap between training set' and actual risk. Intuitively, this is due to a richer set of hypotheses from which the algorithm can choose one that fits unrepresentative samples in S. Many supervised learning algorithms therefore add a *regularization* term (penalizing complex hypotheses) to the empirical risk as the minimization objective, which now resembles the actual generalization bound - the right-hand side of (1.6).

The *Support Vector Machine* (SVM) algorithm for linear classification optimizes the Hinge loss and an additional complexity term as follows:

$$h_{\text{SVM}} = \arg \min_{w\in\mathcal{R}^n, b\in\mathcal{R}} \sum_{i=1}^{m} l_{\text{Hinge}}(h_{w,b},(x_i,y_i)) + \lambda\|w\|_2^2 \ , \tag{1.7}$$

where again λ is a parameter controlling the trade-off between the surrogate objective and the regularization term $\|w\|_2^2$. An equivalent formulation of (1.7) is the following

[6] The 0-1 loss $l_{01}(y_1, y_2)$ is 0 if $y_1 = y_2$ and 1 otherwise.

[7] Nevertheless, even when employing such a modified optimization objective, the performance benchmark (1.2) remains.

[8] The complexity of H is $\log|H|$ for a finite H, and the VC-Dimension, for infinite H.

quadratic program

$$\min_{w,b,\xi} \sum_{i=1}^{m} \xi_i + \lambda \|w\|^2$$

subject to

$$y_i(w^T x_i + b) \geq 1 - \xi_i \qquad \forall i = 1..m$$

$$\xi_i \geq 0 \qquad \forall i = 1..m$$

This alternative formulation can also be derived by optimizing h_{SVM} for maximal geometrical *margin*, the distance of the linear separator from the closest sample in S.

Ridge and Lasso regression are two regularized variants of linear regression, that add to the objective $\sum_{i=1}^{m}(h^T x_i - y_i)^2$ the regularization terms $\lambda\|h\|_2^2$ and $\lambda\|h\|_1$ respectively:

$$h_{\text{Ridge}} = \arg\min_{h \in \mathcal{R}^n} \sum_{i=1}^{m}(h^T x_i - y_i)^2 + \lambda\|h\|_2^2 \tag{1.8}$$

$$h_{\text{Lasso}} = \arg\min_{h \in \mathcal{R}^n} \sum_{i=1}^{m}(h^T x_i - y_i)^2 + \lambda\|h\|_1 \ . \tag{1.9}$$

Note that in both cases above the objective to be minimized is a convex function, and therefore can be computed efficiently.

The *K-Nearest Neighbor* (KNN) is another popular algorithm in which the algorithms output maps x (e.g., for the case of real output space $X = \mathcal{R}$) to the average of the K nearest samples in S that are closest to x in a given metric (usually Euclidean).

$$\hat{Y}_K(x) \triangleq \frac{1}{K} \sum_{(x_i,y_i) \in N_K(x)} y_i \ ,$$

where $N_K(x)$ is the subset of S containing the K samples neighborhood of x (i.e., K closest).

Due to the unbounded complexity of the hypothesis class H used by KNN (e.g., the class of Voronoi diagrams, for $K = 1$), the bound of (1.6) is inapplicable in this setting. Nevertheless, generalization bounds exist for KNN if a *stochastic* setting is assumed. In a stochastic setting, the input-output relation is probabilistic rather than deterministic, and is represented by a conditional probability distribution of the output space value y being associated to the input space value x. In a stochastic binary

classification setting, for example, the input-output relation is fully characterized by $\mu(x) = Pr[Y = 1|X = x]$. The *Bayes Classifier*

$$h_{Bayes}(x) \triangleq \arg\max_{y \in Y} Pr(y|x) \tag{1.10}$$

is optimal for a stochastic input-output relation, and its inherent risk (the *Bayes error*, denoted $\mathcal{L}^*_D$) is the best one can hope for in stochastic settings.

Finally, for binary classification, and under a smoothness assumption[9] regarding $\mu(x)$, the following is a generalization bound for the 1-NN algorithm (a similar bound exist for the more general case of KNN):

Theorem 2 (Th. 19.3 of [111]). *Let h_{1NN} be the rule yielded by the 1NN algorithm using m i.i.d. samples in a stochastic binary classification setting characterized by a c-Lipschitz $\mu(x)$. Then,*

$$E_{S\sim D^m}[\mathcal{L}_D(h_{1NN})] \leq 2\mathcal{L}^*_D + 4c\sqrt{d}m^{\frac{-1}{d+1}} \ , \tag{1.11}$$

where the c-Lipschitz condition requires that $\forall x_1, x_2 \in X, \ \ |\mu(x_1) - \mu(x_2)| \leq c\|x_1 - x_2\|$, and the input space is $X = \mathcal{R}^d$.

Note that although the form of the bound (1.11) resembles that of the bound (1.6) of Theorem 1, there are two significant differences. First, the bound (1.6) holds with high probability while (1.11) only holds on average over the realization of the training sample S. Furthermore, a polynomial number of samples is required in (1.6) to achieve approximately optimal error with high confidence by an empirical error minimizer, however an exponential number of samples (as a function of the dimension of the input space X) is required[10] for the 1NN algorithm to yield a rule with approximately twice the optimal error (specifically, $m > (\frac{4c\sqrt{d}}{\epsilon})^{d+1}$ samples are required to have the rightmost term of (1.11) be smaller than ϵ).

1.2 On-line Learning

In *On-line Learning* settings, the learning algorithm is required to make a decision upon each observation (e.g., when playing a game). That is, instead of accumulating a

[9] Smoothness, in the sense that two close inputs have similar probability of being labeled 1.
[10] This is *The curse of dimensionality*.

training set S and batch processing its elements to come up with an hypothesis $h \in H$ as detailed above, the learning algorithm sequentially (i.e., at times $t = 1, 2, ..., T$) takes action $a_t \in A$ from an action space A and gets as feedback a loss function $l_t : A \to \mathcal{R}$. Again, only a short overview of the main concepts and key results is given here. See [40] for a comprehensive treatment of on-line learning in the context of prediction and specifically [37] for a thorough review of Multi Armed Bandits settings.

Now, the goal of the learning algorithm is to maintain a small as possible cumulative loss

$$L_T \triangleq \sum_{t=1}^{T} l_t(a_t) \ . \tag{1.12}$$

This rather general formulation captures many on-line learning settings, mainly depending on the domain of the feedback loss function (e.g., containing only the chosen action a_t - the *Multi Armed Bandit* setting, or all the actions in A - the *full information* setting), the nature of the way the loss feedback is generated (e.g., by a stochastic process, or by an adversary - aware of the algorithm's choices history, or not), the properties of the loss function (e.g., convex or not[11]), and the size and structure of the action space A (affecting the feasibility of algorithms that perform computations for each possible member of A).

This on-line framework captures the sequential (that is, on-line) version[12] of any supervised learning setting as presented in the previous section by using the hypothesis space H as the action space A, using the loss function $l_t(h) \triangleq \mathcal{L}(h, (x_t, y_t))$, and for every sample (x_t, y_t) challenge the on-line learning algorithm with x_t (while keeping y_t undisclosed). Indeed, this on-line formulation coincides with on-line versions of popular supervised learning algorithms (such as linear regression and logistic regression) that use the Stochastic Gradient Descent (SGD) method to minimize the risk (1.4). The usage of SGD is justified since the loss function used in those settings is convex, ensuring convergence to the global optimum.

The on-line learning setting may also be viewed as an extension of supervised learning to situations where the distributional assumption regarding the sample generation process (specifically, that they are i.i.d.) does not hold. Since in such situations the sample distribution is no longer fixed (actually, training and testing phases are now

[11] The search of optimal action in settings with non-convex loss functions are harder to address in general.

[12] An on-line framework may be preferable due to its relative simplicity and efficiency, especially when dealing with massive amounts of data.

mixed!), the notion of generalization is replaced with the cumulative loss (1.12) or the more indicative *regret*, quantifying the performance of the online learning algorithm compared to that of the best constant action in hindsight, i.e.,

$$R_T \triangleq L_T - \min_{a \in A} \sum_{t=1}^{T} l_t(a) \ . \tag{1.13}$$

Such accumulated loss and regret metrics depend by definition on the properties of the loss function, which is usually chosen to be of bounded range in $[0, 1]$. Algorithms that achieve an average regret per observation that vanishes ($lim_{T\to\infty} \frac{R_T}{T} = 0$) are called *no-regret* algorithms. Actually, any no-regret on-line learning algorithm can be transformed to a supervised batch learning algorithm using the *On-line to Batch* (OTB) conversion as follows.[13] As before, let $\{h_t\}_{t=1}^m$ be the sequence of actions taken by the online learning algorithm when given sequentially the x_t values of the sample set $S = \{(x_t, y_t)\}_{t=1}^m$ (whose samples were generated i.i.d. according to some distribution D) and incurring bounded loss $l_t(h) \triangleq \mathcal{L}(h, (x_t, y_t)) \leq M$. Let the resulting hypothesis of the OTB algorithm be $h_{OTB}(S) = \frac{1}{T} \sum_{t=1}^{T} h_t$. Then it can be shown (e.g., Theorem 7.13 of [90]) that for every $\delta > 0$, with probability at least $1 - \delta$ (over the realization of the sample set S), the following bound holds

$$\mathcal{L}(h_{OTB}(S)) \leq L_F(h_F^*) + \frac{R_T}{T} + 2M\sqrt{\frac{2\log\frac{2}{\delta}}{T}} \ .$$

That is, the performance of $h_{OTB}(S)$ approaches the optimal achievable, requiring a manageable number of samples to achieve high accuracy and confidence, as defined in the PAC framework for batch supervised learning.

Another on-line learning abstraction captured by the above model is the *Prediction using Experts* model. In this setting, the on-line learning algorithm has access at every time step t to the advice (predictions) $\{y_{i,t}\}_{i=1}^N$ of N experts, and in each time step t acts by *predicting* $\hat{y}_t$. Thereafter, the actual outcome y_t is revealed, and associated losses $l_{i,t} = l(y_{i,t}, y_t), l_t = l(\hat{y}_t, y_t)$ are incurred to each expert and to the learning algorithm, respectively.

In the realizable binary classification setting (where there exists an underlying hypothesis h^* from a finite class H such that $y_t = h^*(x_t)$) with 0-1 loss, for example, each

[13] Assuming a convex hypothesis set H, and a loss function $l(\cdot, \cdot)$ that is convex in its first argument.

hypothesis h may be regarded as an expert, predicting $y_{i,t} = h(x_t)$. A natural approach for an on-line learning algorithm in this setting is to maintain a subset $V_t \subset H$ of all experts consistent with all past observed examples. Indeed, the *Halving* algorithm (so called since upon every mistake the size of V_t is at least halved, resulting in the mistake bound) that reacts to the challenge x_t by predicting according to the majority vote $\hat{y}_t = \arg\max_{y \in \{0,1\}} |\{h \in V_t : h(x_t) = y\}|$ is guaranteed to make at most $\log_2(|H|)$ mistakes (thereby no-regret).

No-regret deterministic learning algorithms for the expert setting exist also for the non-realizable setting. One such algorithm for the binary classification setting, a generalization of the Halving algorithm, is the *Weighted Majority* (WM) algorithm. The WM algorithm maintains a weight $w_{i,t}$ for each expert i that is initialized to 1 and multiplicatively decreased by a constant β whenever the expert prediction is wrong. At each time step t the WM algorithm predicts according to the weighted majority vote (hence the name) $\hat{y}_t = \arg\max_{y \in \{0,1\}} \sum_{i:y_{i,t}=y} w_{i,t}$. No-regret generalizations of the WM algorithm exist for other non-realizable setting. The *Exponential Weighted Average* (see Section 7.2.4 in [90]) is one such algorithm for bounded loss functions $l(\cdot, \cdot)$ that are convex in their first argument.

In a non-realizable setting, for the 0-1 loss (being non-convex), such no-regret guarantee does not hold. Actually, since the values of y_t might be adversarially chosen, the regret of any deterministic on-line algorithm in this setting can be made $O(T)$. Therefore, randomization by the algorithm is crucially required to achieve no-regret.[14] The Randomized Weighted Majority (RWM) algorithm introduced by [80] is an example of such a no-regret algorithm. The algorithm acts at every time step t by randomly choosing the prediction of one of the experts according to the probability distribution induced by a set of weights $\{w_{i,t}\}_{i=1}^N$ maintained for each expert and updated every time step t as in the WM algorithm. Specifically, the resulting probability distribution $p_t = (p_{1,t}, \ldots, p_{N,t}) \in \Delta_N$ over the experts at time step t is the following

$$p_{i,t} = \frac{e^{-\mu L_{i,t-1}}}{\sum_{j=1}^N e^{-\mu L_{j,t-1}}} \tag{1.14}$$

where $\mu > 0$ is a tunable learning parameter and $L_{j,t-1}$ is the accumulated loss (that is, the number of mistakes) of expert j along the first $t-1$ time steps. The regret of the

[14] Since the regret now is a random variable, the no-regret condition is with high probability, approaching 1 as $T \to \infty$.

RWM algorithm can be shown to be bounded (for any loss sequence!) $R_T \leq 2\sqrt{T \log N}$, therefore it is *no regret*. This $O(\sqrt{T})$ regret performance is optimal in a sense, since for $N = 2$ no learning algorithm can achieve average regret lower than $\sqrt{\frac{T}{8}}$, even when the adversary is stochastic (see Theorem 7.5 in [90]). A more general lower bound (for N experts) of $\Omega(\sqrt{T \log N})$ can be proven, matching the upper bound.

An even more demanding on-line setting is the Multi Armed Bandit setting, where the on-line learning algorithm only gets $l_t(a_t)$ as feedback (the loss related to the chosen action), instead of the entire loss function $l_t : A \to \mathcal{R}$. To address the extra freedom granted to the environment in this setting, successful strategies of on-line learning algorithms for the MAB setting combine *Exploration* (that is, a search for beneficial actions) with *Exploitation* (performing actions identified to be beneficial). Two prevalent algorithms employing such strategies are UCB [15] for the stochastic case (where the outcomes of each arm are i.i.d. from a predefined but unknown probability distribution) and EXP3.P [16] for the more general adversarial setting. Both achieving no-regret by cleverly mixing exploration and exploitation. UCB uses the *Optimism in face of uncertainty* heuristic (a design principle suggesting that among the set of plausible environments consistent with observed data, the most favorable is assumed for making a decision), and performs exploration by assigning favorable expected loss ranges to experts (bandit arms) less sampled. EXP3.P achieves $O(\sqrt{TN \ln \frac{N}{\delta}})$ regret with high probability (at least $1 - \delta$ over its own internal randomization) by choosing weights that are a mix of (1.14) and a uniform distribution over the experts (for exploration) [15]. Finally, no-regret algorithms exist also for settings in which an infinite number of arms are modeled. However, structure relating the losses of different arms has to be assumed (since only a finite number of arms can be sampled). In *Online Convex Optimization* for example, the decision space A is assumed to be a compact subset of $\mathcal{R}^n$, and variation limitations (such as bounded gradients) are assumed on the convex loss functions $l_t : A \to \mathcal{R}$.

1.3 Bayesian Methods for Classification

In practice, the input-output relation sought after by a supervised learning algorithm might not be genuinely represented in the training set. This might be due to mea-

[15] Also, the past accumulated losses in (1.14) are replaced by estimates, as required for arms not sampled.

surement imperfections (e.g., noise, sensitivity, range) or data loss and corruption of statistical nature,[16] occurring in other preliminary stages of the data processing and preparation.

Bayesian methods are applied in *parametric* learning settings where the input-output relation (specifically, its stochastic nature, e.g., sampling noise) is modeled using a parameterized conditional probability distribution $Pr_\theta(y|x)$. In the most general setting of *Bayesian Reasoning*, a *prior* probability distribution $Pr(\theta)$ over possible values of the parameter θ is assumed, and a *posterior* probability distribution $Pr(\theta|S)$ is estimated from the training data S. This derived posterior induces a conditional distribution for the input-output relation[17]

$$Pr(y|x) = \int d\theta Pr(\theta|S) Pr_\theta(y|x) \ . \tag{1.15}$$

Alternatively, instead of estimating a posterior probability distribution, still assuming a parametrized stochastic input-output relation, a specific value of the parameter $\hat{\theta}(S)$ that best fits the training data S may be derived. In this case, the related $Pr_{\hat{\theta}}(y|x)$ is immediate and the application of (1.15) is skipped. Either way, the resulting conditional distribution $Pr(y|x)$ is combined with a loss function $l(\cdot,\cdot)$ to define the expected *risk*

$$R(\hat{y}|x) = E_{y|x}[l(\hat{y}, y)] \ , \tag{1.16}$$

of a classifier making the decision y when given input x. Finally, given an input x, the *Bayes Classifier* in this setting returns the output of minimal expected risk. Namely,

$$h_{Bayes}(x) \triangleq \arg\min_{\hat{y}} R(\hat{y}|x) \ . \tag{1.17}$$

In a sense, the classifier (1.17) is the best one can hope for in a stochastic setting. Note however, that Bayes error is usually not achievable (even if the conditional label probability is known upfront), since it might be that $h_{Bayes} \notin H$.

Now, the Bayes rule and the data i.i.d. assumption may be used for estimating a posterior distribution $Pr(y|x)$ over the parameter space given the data $S = \{(x_i, y_i)\}_{i=1}^m$ (or some fixed, optimal value $\hat{\theta}$) to be used in (1.16) and ultimately in (1.17), as follows

[16] Data may be affected also intentionally, due to rounding or discretization, for example.
[17] Assuming a continuous parameter space. A similar formula applies discrete θ.

$$Pr(\theta|S) = \frac{\Pi_{(x,y)\in S} Pr((x,y)|\theta) Pr(\theta)}{\Pi_{(x,y)\in S} Pr((x,y))} \ . \tag{1.18}$$

Two methods that return a specific $\hat{\theta}$ estimate rather than a posterior distribution are *Maximum Likelihood* (ML) and *Maximum A posteriori* (MAP). Both methods aim for the parameter value that best explains the i.i.d sampled data, and return the parameters $\hat{\theta}_{\text{ML}}$ and $\hat{\theta}_{\text{MAP}}$, respectively:

$$\hat{\theta}_{\text{ML}} \triangleq \arg\max_{\theta} Pr(S|\theta) = \arg\max_{\theta} \sum_{(x,y)\in S} \log Pr(x,y|\theta) \tag{1.19}$$

$$\hat{\theta}_{\text{MAP}} \triangleq \arg\max_{\theta} Pr(\theta|S) = \arg\max_{\theta} \big(\log Pr(\theta) + \sum_{(x,y)\in S} \log Pr(x,y|\theta) \big) \ . \tag{1.20}$$

The key difference between the two methods is that the MAP model specifies a *prior* probability $Pr(\theta)$, whereas the ML method does not assume an underlying statistical nature in the generation of θ.

To allow for feasible characterization of the estimated conditional distribution $Pr(\theta|S)$ in (1.18) or the $\hat{\theta}$ estimates in (1.19) and (1.20), some structure on the statistical nature of the input-output relation has to be assumed (Note that in the most general case, for a discrete setting, individual estimates of $Pr((x,y)|\theta)$ are required for any combination of the values of x, y, and θ).

Naive Bayes is a generative model that assumes independence of the individual attributes of the input x given the class y. i.e., $Pr(x|y) = \Pi_{j=1}^{d} Pr(x_j|y)$. Under this assumption, (1.19) simplifies to

$$\hat{\theta}_{\text{ML}} = \arg\max_{\theta} \sum_{(x,y)\in S} \big[\sum_{j=1}^{d} \log Pr(x_j|y,\theta) + \log Pr(y|\theta) \big] \ ,$$

exponentially reducing the number of parameters to be estimated. Similar savings are achieved for (1.18) and (1.20).

A different (more general) simplifying assumption, for a continuous input space, is made in the *Full Bayes* model. Here, the attributes of x are assumed to follow a multivariate Gaussian distribution given the class y (that is, with mean and covariance matrix depending on the specific class y). Under this simplification, for example, a closed form for $\hat{\theta}_{\text{ML}}$[18] can be derived.

[18]The parameter $\hat{\theta}_{\text{ML}}$ in this case actually comprises of the mean $\hat{\mu}_{\text{ML}}$ and covariance matrix $\hat{\Sigma}_{\text{ML}}$.

1.4 Robustness Through Regularization

The optimization formulas (1.7), (1.8), and (1.9), for h_{SVM}, h_{Ridge}, and h_{Lasso} respectively, all share a common structure, balancing (through a trade-off regularization parameter λ) between an empirical error of a potential hypothesis from H and its size. As noted above, for a given sample set, a large hypothesis set H available to the learning algorithm might result in over-fitting, denoted *estimation error* (that is, an empirical error that is not representative of the true underlying risk). On the other hand, limiting the hypothesis class H available to the algorithm to choose from introduces inherent *approximation error* (also called *inductive bias*), i.e., the excess risk due to precluding the learning algorithm from using the true underlying hypothesis $f \notin H$. Therefore, the parameter λ used in regularization-based algorithms as in (1.8), (1.9), and (1.7), can be interpreted as controlling the trade-off between the improved estimation error (achievable by using regularization) and the potential bias introduced.

It is important to note that the restriction of the learning algorithm to choose its output from a hypothesis class H that is a strict subset of the set of possible input-output relations F is crucial for PAC learnability, whereas otherwise, for rich enough classes F, by the *no free lunch* theorem (see e.g., *The Statistical No Free Lunch Theorem* in Section 5.1 of [111]), there is no learning algorithm that can satisfy the requirement (1.3) above for any accuracy and confidence levels.

Regularization is key for achieving robustness in the on-line learning setting as well. A key challenge to the on-line learner, distinguishing it from classical supervised batch learning is the potential change, upon every sample, in the environment generating the samples (whereas in batch learning the samples are assume to be i.i.d. from some distribution). As already noted in section 1.2, In the *predicting using experts* setting for classification, any deterministic algorithm is subject to a malicious opponent[19] that can adapt and choose the sequence y_t to result in $O(T)$ regret (1.13). As already indicated, robustness to such adversarial environment is achieved by introducing randomization to the on-line learning algorithm, as in the RWM algorithm (1.14), for example.

Actually, the RWM action rule (1.14) is the solution of the *Follow The Regularized Leader* update rule

$$p_t = \arg\min_{p \in \Delta_N} \mu \sum_{i=1}^{N} p_i L_{i,t-1} + R(p) \ , \tag{1.21}$$

[19] In an unrealizable setting, no longer tied by a predetermined h^* governing the input-output relation.

where the convex regularization term $R(\cdot)$ added to the empirically observed past loss $\sum_{i=1}^{N} p_i L_{i,t-1}$ of some candidate weighted action p, is the negative entropy $R(p) = -\sum_{i=1}^{N} p_i \log \frac{1}{p_i}$. Again, as in Section 1.4 above, robustness is achieved by optimizing through regularization, since in the absence of the regularization term, the optimization (1.21) reduces to choosing the best past expert, a strategy shown to be highly vulnerable.

1.5 Overview of The Thesis

Evidently, learning algorithms are based on a model of the environment in which they operate, and their performance depends on the degree of agreement of their assumed model with reality. This thesis mainly investigates key aspects of robustness - the sensitivity of a learning algorithm to discrepancies between the assumed model and reality - as related to three specific learning settings: The applicability of model-light methods (i.e., prone to simple models, if at all) to agent-design for the Trading Agent Competition (TAC - a simulated environment for confronting trading agents' strategies), Robust algorithms to cope with domain discrepancy (that is, the Domain Adaptation setting, where the algorithm's output hypothesis is required to operate in an environment that is different from the one in which it was trained), and finally, collaborative social learning as a platform for parameter estimation that given a very simple strategy space comparably performs at equilibrium as the optimal estimator.

The model-free KNN and RWM algorithms, introduced in Sections 1.1 and 1.2 respectively, are used to implement TAC agents for the Ad Auctions game, eventually winning the competition. The robustness achieved through regularization, as detailed in Section 1.4, is used to design a robust domain adaptation algorithm (a variant of SVM, introduced in Section 1.1) and associated generalization bounds. The thesis concludes with an analysis of the equilibrium resulting from a model for social computation through information propagation using simple (that is, *history-independent*, thereby robust) strategies, and the comparison of the performance of the related estimator to the optimal achievable using the Bayesian approach introduced in Section 1.3.

1.5.1 Part I - Autonomous Bidding Agents

In the first part of the work, *Autonomous Bidding Agents*, the strategies of software agents implementing algorithms for optimizing execution of marketing campaigns is considered. Such agents, becoming ubiquitous in the Internet economy, bid for the opportunities to have ads displayed to users engaged in Internet search or browsing. The bids of the agents depend on many factors - the attributes and purchasing behavior of the users, the characteristics of the browsed web site, the targets of the outstanding advertising campaigns, the behavior and characteristics of competing advertisers and their targets, and many more.

The complex structure of the setting and its game-like nature make the evaluation of related strategies very hard to conduct. Therefore, synthetic platforms, such as the *Trading Agent Competition* (TAC) described in chapter 2, where research teams regularly compete by facing their strategies against each other's, are controlled environments that serve to evaluate and analyze related algorithms.

First, a *model-light* approach to the design and implementation of a top-performing trading agent in a dedicated TAC game (TAC-AA) is described in chapter 3. It is shown that such an agent can enjoy many of the robustness and simplicity benefits of being model free while still achieving top scores. The applicability of strategies used in TAC games to the more demanding and uncontrolled real-world setting is then considered in chapter 4 through a study of the robustness of some of the most successful TAC-AA agents to unexpected changes in the simulated environment. The analysis shows that the top-performing agents are among the most robust. This encourages the possibility of technology transfer of successful TAC strategies to the real world, and is somewhat surprising since it could be expected that top performance is achieved by a (potentially fragile) practice of tailoring the algorithms to specific game parameters.

Finally, a new TAC game, *Ad Exchange* (TAC-AdX) is introduced in chapter 5, where its implementation and competitions based on its description are described. TAC-AdX simulates the prevalent model for display-advertising in the Internet through a central exchange that coordinates ad supply (by publishers, that is, web sites) and demand (by advertisers) through real-time bidding in dedicated auctions. TAC-AdX was designed in an attempt to capture some of the key trade-offs faced by real-world advertisers. Specifically, the way the advertiser's ability to successfully execute advertising campaigns (i.e., by achieving targeting ad reach goals) and related costs effects

their ability to win future campaign contracts (through some rating) and their related revenues. Reports from first competitions of the TAC-AdX game show key mechanisms (analogous to strategies that may be implemented in the real world in similar settings) employed by such trading agents. Furthermore, The TAC-AdX game may serve as a platform to compare the many alternative mechanisms available for implementing key components of the (real-world) Ad Exchange scenario, mainly the type of auction performed by the Ad Exchange, the reserve price setting by the publishers, and the pricing of information regarding users' attributes by dedicated third-parties.

1.5.2 Part II - Robust Domain Adaptation

In the second part of this thesis, a generalization bound and related learning algorithm are derived for the *Domain Adaptation* setting. In domain adaptation, introduced in chapter 6, the underlying uncertainty faced by the supervised learning algorithm is regarding the stationarity of the underlying domain - the probability distribution governing the input-output relation. Specifically, the training samples may be from one domain (the *source* domain), while the algorithm's output hypothesis might be tested in a different domain (the *target* domain). Naturally, the source and target domains have to be somehow related to make learning possible if at all.

This thesis addresses the domain adaptation problem using the *Robust Optimization* paradigm, where the assumed prior relation between the source and target domains (parametrized through a newly introduced λ-*shift* distance) serves as the input range over which robust optimization occurs. Specifically, *Algorithmically Robust* SVM variants for classification and regression are described in chapter 8, in which the prior uncertainty regarding domain discrepancy translates into the optimization constraints. The *Algorithmic Robustness* framework and characterization (presented, together with the *Robust Optimization* paradigm, in chapter 7) is further used to derive related domain adaptation generalization bounds.

1.5.3 Part III - Multiagent Learning

The performance of estimators stemming from social learning platforms is investigated in the last part of this thesis. In social learning, a set of self interested agents collaborate to perform some computation that depends on the aggregate of the privately held information by each. The agents may have stake in the result of the computation

(e.g., voting), or in the probability of some external event that their signals depends upon (e.g. events effecting the value of a stock, in stock market trading) which the computation is assumed to approximate.

Focusing on the latter case, where agents sequentially update the outstanding value of a computation regarding the probability of an external event, the result of the computation may be interpreted as an estimator for the probability of the event occurring, aggregating the private signals of the agents. The mean square error is generally used to quantify the performance of such estimators. The self interested agents are elicited to participate (and truthfully use their private signal) through a market scoring rule that offers an expected positive payoff to agents that improve the quality of the computation.

The elements of the social learning setting and *Prediction Markets* (as a specific example) are introduced in chapter 9, where we show that the sum of the expected total payoff of the participating agents and the performance of the resulting estimator (a generalization of the mean square error) is constant, thereby quantifying the inherent cost-accuracy trade-off in such settings.

In chapter 10, aiming to capture trusted recommendation chains settings (where private observations - e.g., regarding the quality of a product - are used to make one-on-one recommendations among trusted agents), we introduce a simple model for sequential social learning where agents are history independent (that is, they are unaware of past actions, or their own arrival order, but know the total number of agents T). Inherently, the strategy space of such history-independent agents is limited to operate on their private signal and the observed state of the computation only. We consider a simple space of strategies that update the current state to a linear convex combinations of those two quantities (effectively consisting of moving averages), and investigate the socially optimal and equilibrium strategies for settings using the quadratic scoring rule in which the update strategy is dictated and where strategic agents are free to choose their strategy, respectively.

The performance of the resulting estimators (e.g., expected error, probability of bias), essentially indicating the convergence rate as a function of the total number of agents T, is analyzed and compared to the performance of the optimal (Bayes) estimator having full access to all the agent's private signals, thereby quantifying the performance loss incurred due to performing a sequential history-independent social computation instead. Similar analysis is also carried for some model extensions, in

particular where a distribution over the total number of agents is assumed.

1.5.4 Published Papers

- **Chapter 3** is mainly based on [109]: *A Model-Free Approach for a TAC-AA Trading Agent*. Schain, Mariano and Hertz, Shai and Mansour, Yishay. Lecture Notes in Business Information Processing 2012, and Trading Agents Design and Analysis (TADA) Workshop 2012.

- **Chapter 4** expands upon [66]: *An Empirical Study of Trading Agent Robustness*. Hertz, Shai, and Schain, Mariano and Mansour, Yishay. Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems.

- **Chapter 5** is mainly based on [110]: *Ad Exchange - Proposal for a New Trading Agent Competition Game*. Schain, Mariano and Mansour, Yishay. 14th International Workshop on Agent-Mediated Electronic Commerce and Trading Agents Design and Analysis 2012.

- **Chapter 8** is mainly based on [84]: *Robust domain adaptation*. Mansour, Yishay and Schain, Mariano. Annals of Mathematics and Artificial Intelligence, pages 1-16, 2013. Springer.

Part I

Autonomous Bidding Agents

Chapter 2

The Trading Agents Competition

As autonomous bidding agents became more and more prevalent in the Internet economic arena, the related mechanisms and strategies implemented by the involved entities gained the interest of academic researchers (and the industry, of course).

The choice of mechanisms for a trading setting (e.g., for pricing and allocating goods among interested potential buyers) is a market-design problem that has been researched extensively for centuries, mainly in the economic community. Internet-based trading, however, introduced new aspects (mainly the existence of digital reproducible goods, trading scales, and computational abilities and limitations) that called for a computer science perspective and as such attracted researchers in related (or perhaps newly created, e.g., *algorithmic mechanism design*) fields. Similarly, computational abilities (such as speed, accuracy, and scale) available to autonomous trading agents called for algorithmic analysis of trading methods and gave rise to the development of adaptive strategies that use massive data for their decision making.

While the search for optimal mechanisms (a search that has been successful in some basic settings) aims at universal properties (that is, properties that hold regardless of agent' strategies) such as dominance of truthful actions, the performance of a trading agent's strategy inherently depends on the strategies of its competitors and therefore can't be universally optimal. Furthermore, the increasing structure and complexity of real settings adds to the challenge of analyzing and evaluating trading agent's strategies. This was a key motivation [121] for the introduction of the Trading Agent Competition (TAC) [122] in which a community of researchers and industry professionals implement trading agents for a predefined set of well-specified common settings (TAC *games*) and meets yearly to confront their agents and analyze the results. By focusing on a

common synthetic setting that abstracts the essence of some real trading scenario (but of manageable complexity and structure) researchers may evaluate and compare their solution approaches to the trading agent's problem. This also gives hope for technology transfer - the applicability of the strategies that perform well in the game setting to the original real scenario.

Since the year 2000, several TAC games were introduced and related competitions took place. The first TAC game *TAC-travel* [62] introduced an online travel shopping problem in which competing agents (implementing the strategy of travel agents) trade travel goods (such as hotel rooms and flights) in dedicated markets to assemble trip itineraries, as requested by customers. Subsequent games are *TAC-SCM* [105] - a supply chain management scenario, *CAT* [93] - a market-design game, *TAC-AA* [70] - a sponsored search (Ad Auctions) setting, and most recently *PowerTAC* [72] - an energy trading platform, and *TAC-ADX* [110] - an Ad Exchange based Internet display-advertising scenario. The TAC-Ad Auctions game is now presented in some detail (see [7] for the complete specification and [70] for a detailed account of the game design). This will serve later in this chapter to illustrate common aspects of TAC games (structural and conceptual) and also as a reference for future chapters in which the **tau** agent for the TAC-AA game and related key aspects of the game are discussed.

2.1 The TAC Ad-Auctions Game

Sponsored Search is the main business model for search service providers such as Google - where advertisers pay, through an auction mechanism, for their ads to be displayed alongside the neutral (*organic*) search results. The Ad-Auctions game is a simulated Sponsored Search environment in which the competing agents implement the strategies of advertising merchants - they try to sell items from a limited inventory by bidding for keywords. Specifically, the advertisers bid for their ads to be displayed alongside search results of queries containing the keywords, performed by a set of user populations. The population of searching/clicking/purchasing users and the associated keywords auctions are simulated by the game server. In what follows, we refer to the functionality of managing user queries and related auctions, ad display, and clicks as the simulated *publisher*. As described below, to be successful, bidding agents should assess the marginal value of winning a keyword's impression, which in turn depends on their inventory level

and the purchasing *state of mind* of the users.

Each simulated user, making at most one query per simulated day, has a predetermined preference to one of three manufactures and one of three product types, thus there are 9 different users populations (in the standard game setting, there are 10000 users in each simulated population). Users queries (a total of 16 types) are characterized each by the (possibly missing and not necessarily the preferred ones) manufacturer and product type. The specific query type made by a user depends on his state, reflecting his tendency to search, click, and make a purchase.[1] All users start at a Non-Searching (NS) state, from which they transition with a predefined probability[2] to an Informational Search (IS) state. In this state, users submit queries and click ads, but do not convert (that is, do not make a purchase, do not *transact*). From the Information Searching state, users may transition to any of the Focused Searching states (F0, F1, F2). The focused states reflect the users different search sophistication, or its degree of knowledge about its internal preference, ranging from null query in focus level F0 to detailed search in focus level F2. Users may transition from a low level Focused Search to a higher one, and may also transact. After a user transacts, he may return to the Non Searching state and start the process all over again. The user's state transition probabilities (except the purchase transaction) are governed by a Markovian model that is known to all competing agents through the game specification. To model bursts of search behavior, the transition probability from the Non Searching state to the Information Searching state may be increased significantly, with probability `Pburst`. A burst can last up to three days, after which the transition probability returns to its former state.

Each simulated user query results in a publisher's created *impression* - a list of ads ranked according to the results of a Generalized Second Price (GSP) auction among the advertisers' submitted bids (the auction result determines also the price to be paid to the publisher upon a user click on their ad, this price is termed *CPC* - Cost Per Click). The highest ranked ads may be designated (depending on the bid amounts) as *promoted slots* - having higher click probability. The user views the ads sequentially and may click on one ad or more, with probability determined by a Click-Through Rate (CTR). The CTR for every query is determined by three factors: the advertiser effect (a baseline value randomly chosen by the server for each competing advertiser at game

[1] The user's state is therefore referred to as his *state of mind*.

[2] All such predefined parameters are detailed in the game specification.

start); whether the ad is placed in a promoted slot or not; and whether the ad targeting (which is designated in conjunction with the bid submitted by the advertiser) matches the users product preference.

Upon an ad click,[3] a user may convert (purchase the product offered by the advertiser) with a probability (designated Conversion Rate - CVR, the fraction of actual sales out of the total clicked ads) that depends on three factors: the users state (higher focus levels convert at higher rates), whether the advertiser's product preference matches the users product preference (each advertiser also has a designated preferred product, denoted it's *specialty* product), and the outstanding usage degree of the advertiser's sales capacity - a soft constraint that introduces Inter-query dependencies as follows: Each competing agent is assigned one of three discrete capacity levels (Low, Medium and High) at the beginning of the game session. An advertiser that sells over his capacity (during a moving window of five days) suffers a decrease of users conversion probability, which in turn reduces the Return on Investment (ROI), since the CPC remains the same. Finally, the advertiser collects a predetermined revenue for each conversion.

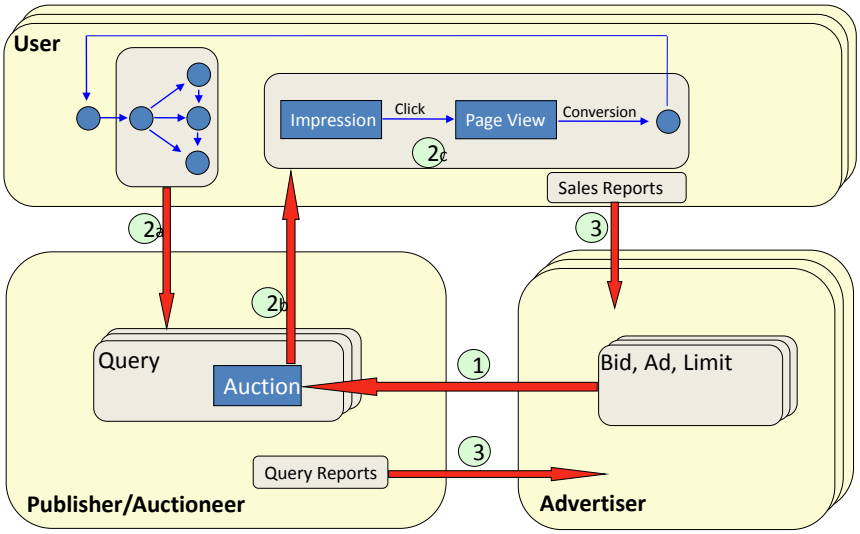


Figure 2.1: The Ad-Auctions game flow dynamics

The competing agents implementing the advertisers' strategies interact with the

[3] If an impression is not clicked, the probability that a user impressed by a certain ad will continue to view the next ad is determined by a hidden continuation parameter for each query type.

game server that simulates the user populations and the publisher actions. The game flow and dynamics (as illustrated in Figure 2.1) are such that each advertising agent provides a daily bid bundle consisting of bids, limits and ad specifications to the publisher. The publisher uses the advertisers bid bundles during the following simulated day to run an auction for every search query performed by a user. The users may click on ads and purchase products from the winning advertisers: For every user query in which the advertiser got his ad displayed (an *impression*) and the user clicked and further made a purchase (*convert*), the advertiser collects a fixed revenue. Now, the advertiser's revenue is significantly higher upon a purchase of a product by its preferred manufacturer (each advertiser also has a preferred manufacturer, denoted it's *specialty* manufacturer - this and the advertiser's specialty component are randomly allocated and indicated to each competing agent at game start).

Reports regarding past performance are provided daily to the advertisers so they can adjust their future actions accordingly. The advertiser problem is therefore to maximize his accumulated net revenue (during the 60 simulated days) by providing optimal bid bundles considering the potential costs and revenues for each query (affected mainly by the user populations size and state, and by the competing agents bids).

2.2 Characteristics of TAC Games

TAC games share similarities in many aspects. In terms of infrastructure, they all implement a game server that simulates the setting and interacts with *remote* and *independent* competing agents. The server also provides a web based interface for administration, log access, and game viewing. The schematic TAC infrastructure is illustrated in Figure 2.2.

TAC games also share the theme of a daily cycle in which reports are sent from the server[4] to the competitors regarding the previously completed simulation and then while the server simulates the current day the competing agents calculate their decisions[5] - to be sent back to the server upon the end of the simulation. This schematic cycle is illustrated in Figure 2.3:

Conceptually, by their very nature, the TAC games present to competing agents a

[4] In the Ad Auctions game for example, the server notifications include average position, clicks, revenue, and cost for each query.

[5] Again, in the Ad Auctions game, the advertiser's decisions are the bid and ad to use for each query, and related spend limits.

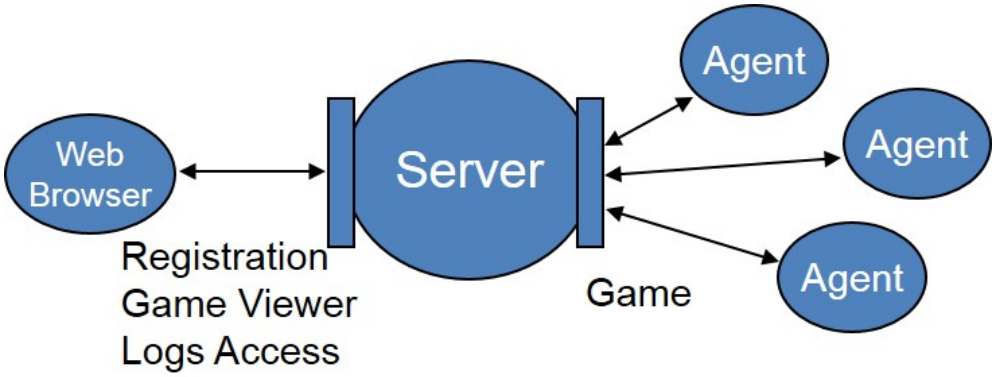


Figure 2.2: Generic TAC game infrastructure

game-theoretic challenge, and this is where they depart from some other presumably similar but actually remotely related competitions [4, 5, 116] that rather pose a decision-theoretic problem. Moreover, other games such as [18, 32, 104] that *do* present a game theoretic challenge to competing agents (that is, the performance of a competing agent depends on the strategies of the competitors and therefore may not be evaluated in isolation) are significantly simpler in terms of the strategy space - Mainly, their setting requires a single decision by the competitors (e.g., a bid in an auction, an action in a simple game, etc.). As a result, such games lack a key characteristic of TAC games - ***multiple*** and ***interdependent*** decision venues. In the Ad Auctions game, for example, the daily decision of the advertiser is comprised of bids for each separately auctioned keyword (resulting in sales of inventory items and related revenue, depending on the keyword-specific applicable click and conversion rates). The consequences of such keyword auctions, however, may influence the other keywords related revenue since overselling due to high bids on one of the keywords might result (due to the capacity constraints) in a reduced CVR for the others.

The problem faced by competing agents in TAC games is therefore of a game theoretic nature. To be successful, agents should implement adaptive strategies that address inherent trade-offs in the game based on feedback from a dynamic market. In that sense, maybe the closest line of competitions to TAC is RoboCup [74]. Now, another distinctive property of TAC games is that agents have incomplete information regarding the game played (let alone their competitors strategies). In the Ad Auctions game, for example, some of the parameters of the game are given in the game specification, while others are randomly chosen at the beginning of each game from a known distribution.

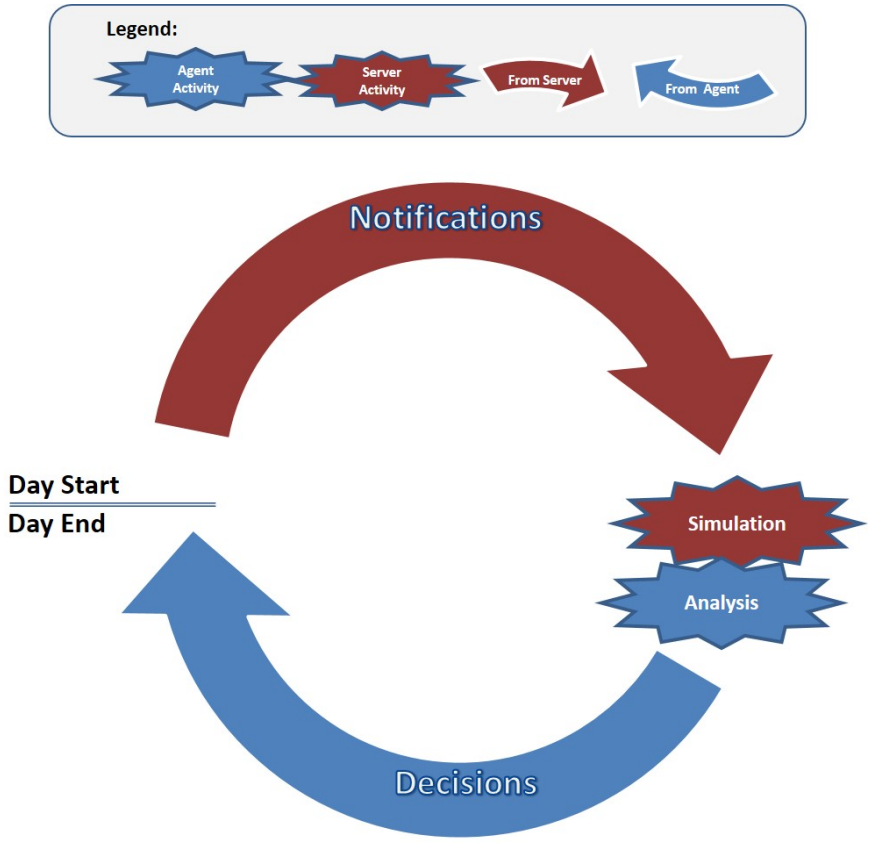


Figure 2.3: Schematic TAC game daily flow

Hence, agents should apply statistical modeling techniques[6] (in addition to strategic reasoning) as the basis of their search for optimal action decisions.

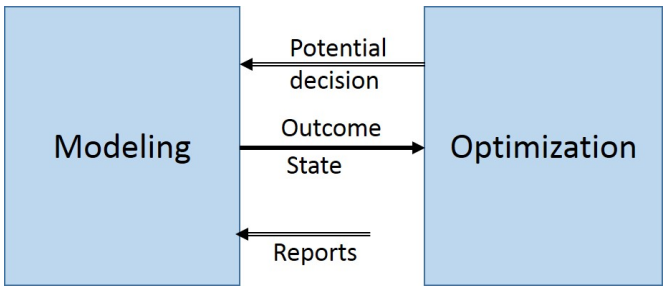


Figure 2.4: Schematic TAC agent's Architecture

As a consequence, the generic architecture of a top performing TAC agent (as illustrated in Figure 2.4) includes a *modeling* component that estimates the game state and an *optimization* component that uses the state estimates to come up with the optimal decisions (considering the inherent uncertainty in the game) to get the game

[6]In the Ad Auctions game, one example is the estimation of the distribution of user populations over states, which has crucial influence on the profitability of bidding on a query since (for example) in a certain state users may click (incurring costs) but never convert.

to a desirable state. Note that such state estimates by the modeling component may be regarding past states (based on reported observations) but also regarding future states given hypothetical decisions. Note also that the estimated game state may include the estimated state of competing agents in addition to those associated to stochastic hidden game parameters. In the Ad Auctions game, for example, the modeling component of our **tau** agent estimates (among many other things - as detailed later in Section 3.4.2) the hidden CTR of queries and the resulting costs and revenues as a function of the bid level on a query, and the **TacTex** agent for the 2010 TAC AA competition (as reported in [101]) employs modeling of competing agents strategies.

Finally, The random nature of the setting in TAC games also implies that TAC competitions are conducted as a sequence of several simulated games in order to average out the effect of random occurrences on the results of a specific game execution.

2.3 A Short Discussion

Since its introduction in the beginning of the millennium, the TAC series of competitions became a hub for research teams interested in the many aspects of designing trading agents for complex dynamic market settings. Post-competition reports are typical, both from agent teams and game designers (see e.g., [69] for an analysis of the first Ad Auctions competition, also conducting a game-theoretic analysis and suggesting a modified auction mechanism for optimal publisher revenue). As evident in the review of the competitions' first few years [121], it provided (and continues to do so) a controlled environment in which agents' strategies and a variety of market mechanisms[7] may be applied and evaluated in a transparent and repeatable manner. This is facilitated by the voluntary TAC agents repository [9] and the availability of game server sources, allowing any interested party to locally run competitions (and also make modifications to the game server implementation, as required by research goals) and observe the results. Indeed, as detailed in the next three chapters (and based on the TAC infrastructure), a top performing agent for the Ad Auctions game is implemented, the Ad Auctions game server is modified as part of research regarding the applicability of insights gained in the Ad Auctions game to real settings, and a new TAC game for an Ad Exchange setting is introduced.

[7] For example, Posted Prices, Simultaneous Ascending Auctions, and Double Auctions - all in the original TAC Travel game

Chapter 3

A TAC-AA Top-Performing Agent - A Model-Light Approach

We describe a *model-light* approach to bidding in the Ad-Auctions Trading Agents Competition: First, a simple and robust yet high-performing agent using a *Regret Minimization* optimization algorithm for the 2010 competition, followed by our top performing agent for subsequent competitions in 2011, 2012 and 2013, still using simplified modeling and optimization methods. This chapter is based mainly on [109].

3.1 Introduction

During the past few decades, the Internet transformed from an academic information-sharing tool into a world-wide business platform in which people and companies of all sizes conduct an ever-growing portion of their activities. It has become evident that the competitive advantage of many of the Internet companies relies on their ability to apply machine learning algorithms (for example, learning user preferences to improve user experience, learning user interests to increase the effectiveness of web advertisements, and so on). In the Google AdWords setting, for example, advertisers bid on keywords[1] with the goal of maximizing the net revenue resulting from purchases made by users who clicked the displayed advertisements (offset by the payments to Google for the clicks, as determined by the outcome of the bidding mechanism). Learning algorithms that allow agents to bid such that profits are optimized constitute significant ingredients of the competitiveness of agents.

[1] The bidding activity is usually implemented by automated trading agents.

Optimization, however, relies on a model of the environment (nature). Therefore, to be successful, agents first have to establish a good approximation of the state of nature. In contrast to parametric methods that are tailored to the specific (usually statistical, but also structural) attributes of a model of nature, in the model-light approach minimal assumptions are made regarding the model of nature. As such, model-light algorithms are usually simpler and more robust to model errors or changes. However, ignoring the parameters and structure of a valid model might limit the achievable performance of an algorithm (compared to algorithms that make use of the model). This is the very trade-off considered in our work - we explore the extent of using model-light methods while maintaining a limited performance hit.

In this chapter, a model-light approach to implement a bidding agent for TAC-Ad Auctions [70] is described. Specifically, we research the usage of model-light methods to address the main challenge of the advertiser: to control - by setting the bids - its revenues and costs which are determined by the states of the user populations, the behavior of competing advertisers, and a predetermined capacity constraint.

A conceptually similar model-free approach which uses a simulation-based iterative best-response technique is detailed in [119]. Other previous approaches to the TAC-AA bidding optimization problem (e.g., [99]) rely on the game description to accurately estimate the game parameters, model the user populations state and competitors actions. Both [28, 99] formulate the bidding optimization problem as a combinatorial (intractable) optimization problem and heuristically search for the best solution. Although such methods do achieve top scores in the given game, they might be sensitive to modeling errors, both parametric (estimation errors) and structural (wrong model used).

Our first attempt, for the TAC-AA 2010 competition, almost entirely ignored the game description and based its (almost trivial) modeling and estimation on simple moving averages. Our optimization scheme used a regret minimization algorithm to perform a fractional allocation of the available capacity. This very simple scheme that only performed top-aiming bids on a subset of the available queries resulted in a very simple agent that was implemented very fast and performed quite well - among the 13 competitors it achieved 6th position in the semifinals (therefore qualifying for the final 8 competitors) and 7th in the finals, scoring $\sim 30\%$ behind the top performers.

For the 2011 competition, following [99], we implemented particle filters to model

the states of the user populations. Our implementation however, following the model-light approach, did not use the methods of [99] (which are highly tailored to the game specification) to compute the particle filter input but used a Nearest Neighbor (NN) estimator instead. The NN estimator was trained on historical games data (data from 100 games provides ~ 200000 samples) and achieved $\sim 30\%$ relative estimation error[2]. However, using dedicated techniques we were able to keep the overall modeling accuracy (and the overall agent performance) comparable with the model-heavy methods. The implementation of our top-performing agent also remains model-light in the sense that we do not attempt to model the competing agents behavior for estimating costs and associated game parameters.

We implement a simple and effective bid optimization algorithm by heuristically assuming convexity of the target revenue as a function of the keyword bid levels and applying the equimarginal principle. Specifically, our optimizer was implemented to search for the equimarginal utility bid (see [28] for motivation, and also Section 3.4.4). Using simple linear models for estimating cost-bid relation allowed for an efficient implementation of the optimization as a simple one-dimensional search. Furthermore, to mitigate the risk of under-utilized capacity resulting from over-estimation of the sales potential in a user population we introduced a tunable regularization factor that favors allocations across a high number of queries (i.e., allocations of high perplexity).

All the above resulted in a top performing agent, achieving the third place in the final rounds of the TAC-AA 2011 and 2012 competitions, scoring within 3% of the winner, and winning the 2013 competition.

3.2 High-Level Agent's Architecture and Strategy

As illustrated in Figure 3.1 below, a refinement of Figure 2.4, we partition our agent (named **tau**) to two main components: A *Modeler* responsible to provide effects of bidding decisions by assessing and predicting the state of the changing environment and hidden game parameters (this includes the user population and competitors state, reserve prices, continuation probability, baseline click-through rates, and baseline conversion rates), and an *Optimizer* that uses the services provided by the modeler to come up with optimal actions (the daily bid bundles).

[2]Relative error measures the ratio between the deviation of the estimated value from the actual value to the actual value.

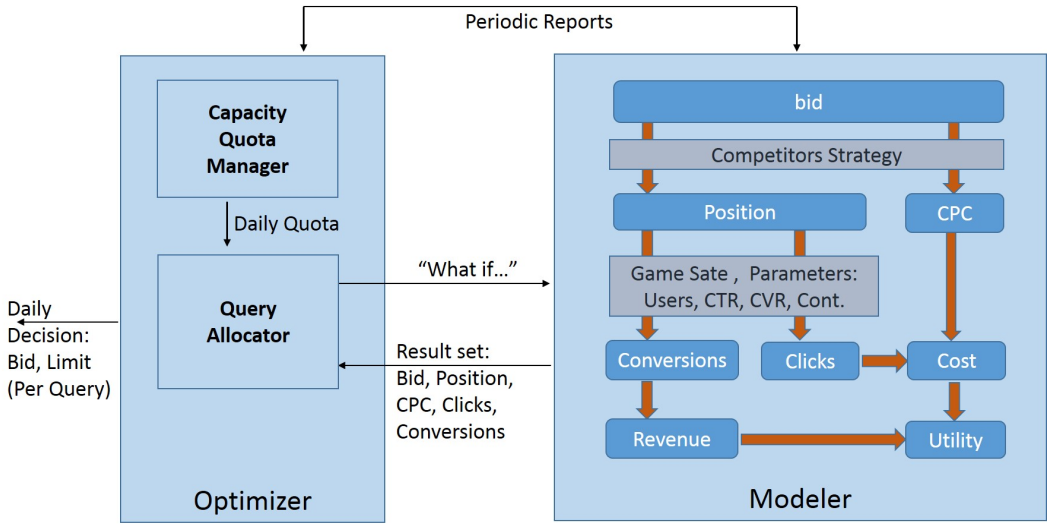


Figure 3.1: High level **tau** agent's Architecture for TAC-AA

We call the daily bid for a query and the related outcomes (auction position, cost per click, number of conversions, number of clicks, daily cost, daily revenue, and utility) the *result set*. The relation of result set items is illustrated within the modeler component in Figure 3.1: Given our **bid**, the competitors bidding strategy (their bids and limits) determines the **position** and **cost per click** in query auctions. Given the position in an auction, the game state and parameters (specifically, the number of users in every state, the continuation probability, click through rate, and conversion rate) determine the total number of **clicks** and **conversions** (and the total **revenue**, since the revenue per conversion is known). Multiplying the CPC and total number of clicks results in the daily **cost**, and the difference between total revenue and cost is the **utility** achieved. It is therefore evident that for the modeler to achieve its goal it has to consider[3] the relation between the bid and the resulting position and CPC (by modeling the competitors' strategies, or any other method, using the daily reports) and estimate the game state and hidden parameters. As will be shown later, the relation between the items in the result set is monotone and estimating any of them fixes (to some degree) the others. Therefore, conceptually, the optimizer may query the modeler "what if" regarding any of the items and get back the full set. As detailed in the following sections, an almost trivial modeler was implemented for the **tau2010** agent, which was later replaced (for the TAC-AA 2011, 2012, and 2013 competitions) by a significantly enhanced modeler that assesses users state and estimates hidden game

[3]The two shaded rectangles of Figure 3.1.

parameters (while - along the lines of our model-light approach - keeping as much of its mechanisms independent of the game specification).

The *Optimizer* implementation of all our **tau** agents are based on a common Capacity Quota Manager (*CQM*) to determine the target daily number of items to be sold (detailed below). Subsequently, a query allocator component sets the number of items to be sold through each query (and the related decision - the bid level and spend limit for each query). As detailed in later sections, a simple allocator over a restricted actions space was implemented for the **tau2010** agent, which was later replaced (for the TAC-AA 2011, 2012, and 2013 competitions) by an enhanced (yet simple) optimization scheme over a significantly richer action space.

The CQM for all **tau** agents implements the following daily capacity allocation scheme: Let $\tilde{Q}_t$ and S_t be respectively the target daily capacity allocated for day t (by the CQM) and the actual units sold during day t (as reported by the game server). Note that since the actual simulation of day t is concurrent with the analysis period of day t, the actual units sold S_t during day t is not known to the agent and is therefore estimated to be $\hat{S}_t$ (details below). Now, to remove the interdependency of queries (specifically, the effect of selling through one query on the conversion rate - and hence the utility - of other queries, as dictated by the capacity constraint) the CQM algorithm aims at total sales of at most a γ-increase of the 5-day capacity C (the slight addition over the capacity is a tunable operation parameter to allow for operation over the capacity at a range that will not hurt the conversion rates significantly). All in all, the target daily capacity allocated by the CQM for the following day $\tilde{Q}_{t+1}$ is set such that it complements the estimate of units sold during the previous days $S_{t-3}, S_{t-2}, S_{t-1}$, and $\hat{S}_t$ to $(1+\gamma)C$, specifically:

$$\tilde{Q}_{t+1} = \max\{\frac{C}{10}, (1+\gamma)C - \hat{S}_t - \sum_{i=1}^{3} S_{t-i}\} \ , \tag{3.1}$$

where in addition, the allocation is never less than a minimal amount $\frac{C}{10}$ which is half the average daily sales required to achieve the capacity quota. Finally, the estimate for *today* sales $\hat{S}_t$ is the average of the quota allocated for today $\tilde{Q}_t$ and the actual allocation by the allocator component[4] $\check{Q}_t$.

[4]Note that the actual allocation for day t was set at day $t-1$.

3.3 A simple Model-Light Agent for TAC-AA 2010

Aiming for an agent as model-free as possible, we used a simple RWM regret minimization scheme in an On-line full-information setting (as detailed in Section 1.2) for the optimization component of our first TAC-AA agent, competing in 2010. The optimizer key decision - how much of the available capacity to allocate to each search keyword, merely fractionally allocated the capacity according to the weights maintained by the RWM algorithm as detailed below. This very simple method, combined with an almost trivial modeler, surprisingly performed in the top half of the competition scoreboard, earning only $\sim 30\%$ below the top scoring agent.

3.3.1 Modeler

Our first agent, taking part in the TAC-AA 2010 tournament, completely ignored the user populations model and the game parameters in its modeler. It only maintained a look-ahead (for day $t+1$, on day t) estimation of the overall conversion rate and click through rate $\hat{v}_q^{t+1}$ and $\hat{c}_q^{t+1}$ respectively, for each query q by adjusting using the actual figure reported for day $t-1$ and using a tunable learning rate parameter τ:

$$\hat{v}_q^{t+1} = \hat{v}_q^t + \tau(v_q^{t-1} - \hat{v}_q^t) \ , \quad \hat{c}_q^{t+1} = \hat{c}_q^t + \tau(c_q^{t-1} - \hat{c}_q^t) \ .$$

3.3.2 Optimizer

Using the generic CQM, for the allocator of the optimizer we significantly reduced the action space by only bidding high on queries. Furthermore, we only bid for queries that have either our agents' preferred component or manufacturer or both (a total of seven queries - five targeted queries are associated with targeted ads and two other queries associated with generic ads) - This is since such queries carry the highest profit potential for our agent and therefore are best suitable to our *high bid* strategy. As a result, setting the capacity allocation of each query, subject to the overall daily quota set by (3.1), was the only remaining daily decision do be made.

We use a *regret minimization* scheme (see Section 1.2 and e.g., [40]) to fractionally allocate the overall capacity across queries: Noting that our problem may be interpreted as a setting of learning from expert advice using *regret minimization* algorithms [34], our query quota allocation is based on the Randomized Weighted Majority (RWM)

algorithm [80]. That is, the different queries are the experts, the gains are the observed profits gained by bidding at each of the queries, and the portion of the overall daily quota to be allocated to each query is the weight w_i^t of expert (query) i on iteration (day) t. Specifically, we apply a regret minimization scheme by fractionally allocating the available daily capacity $\tilde{Q}_{t+1}$ based on a running average of the per-unit sold utility of each query: Let u_q^t be the utility (i.e., costs subtracted from revenue) per unit sold reported for query q for day t. Upon receiving reports for day $t-1$ on day t, a daily score for each query q is set $s_q^{t-1} = e^{\eta u_q^{t-1}}$, where η is a predefined tunable learning rate parameter. Using yet another tunable learning rate parameter α, the overall (adjusted) score of query q for day $t-1$ is updated according to:

$$\hat{s}_q^{t+1} = \hat{s}_q^t + \alpha(s_q^{t-1} - \hat{s}_q^t) \ .$$

The portion of the estimated available capacity for day $t+1$ allocated to query q is the query's portion of the total score. Therefore, the following units quantity m_q^{t+1} will be allocated to query q for day $t+1$:

$$m_q^{t+1} = \frac{\hat{s}_q^{t+1}}{\sum_q \hat{s}_q^{t+1}} \tilde{Q}_{t+1} \ .$$

Finally, using $\hat{c}_q^{t+1}$ and $\hat{v}_q^{t+1}$ (the estimates of the cost per click and the conversion rate for each query, respectively), we set the daily budget spend limit parameter l_q^{t+1} of each query q accordingly[5]:

$$l_q^{t+1} = \frac{m_q^{t+1}\hat{c}_q^{t+1}}{\hat{v}_q^{t+1}} \ ,$$

and the bid b_q^{t+1} of query q for day $t+1$ is set using $b_q^{t+1} = \hat{c}_q^{t+1} + \delta p_q^{t-1}$, where p_q^{t-1} is the reported average position of our bid for query q in day $t-1$ and δ is randomly chosen such that our position is kept high (the bid increases if our position deteriorates).

3.3.3 Results

Upon qualifying for participation during May 19$^{\text{th}}$ and 20$^{\text{th}}$ 2010[6], our *tau* agent scored the sixth highest score out of 15 participants in the semifinal rounds that took place

[5] It is set to control the number of units to be actually sold for each query on day $t+1$, such that the budget is exhausted upon selling the allocated amount of units.

[6] Qualification rounds usually take place in advance of a competition to ensure that agents behave well. That is, score reasonably while not hurting the operation of the game server or other competitors.

on June 7th 2010 and made it to the finals. Out of the 8 agents that took part in the finals on June 8th 2010, our *tau agent* reached the 7th place scoring about 40% less than the winner TacTex [101]. Although an encouraging result overall, above the median of all competing agents, it became evident that in order to score as the top performing agents the action space of our agent should be significantly expanded, as detailed in the remainder of this chapter.

3.4 Tau Agent for TAC-AA 2011 and Beyond

Our target for the 2011 competition was to improve our agent's performance as much as possible while still employing model-light methods. Inspired by the methods and results reported in [28] and [99] we concluded that usage of a good model of the user populations is essential for top performance. We therefore implemented particle filters to model the user populations states. Particle filters, (see [52] for a timely tutorial) are a family of Monte-Carlo methods to recover the Maximum Likelihood estimate of the hidden states of a Markov Chain based on related (that is, statistically dependent) observations. In our case (as suggested by[99]), the hidden state is the distribution of users across states and the related observations are the total number of impressions in a day. Those observations, however, are not directly reported and have to be deduced. Contrary to the methods presented in [99], that rely on specification-tailored computations of the total impressions, our particle filter estimates this quantity using the KNN model-free method. Now, based on the estimates of the users populations distribution across states, the essential quantity to be assessed for each query is the number of users submitting the query while at a *purchasing* state, and the number of users that may submit a query but never convert. As will become evident shortly, those two quantities (specifically, their ratio) are the key metric in evaluating the potential profitability of bidding on a query[7].

In addition to the user's distribution across states (the *Game State* - see Figure 3.1), the modeler also maintains estimates of the hidden game parameters for each query (reserve prices, baseline click through and continuation probabilities), and a monotone (and therefore two-way) mapping of bids to resulting costs and ad positions. As it turns out, a simple linear relation between bids and costs suffices (as an alternative to

[7]Users that click but never convert may result in significant loss to the advertiser

modeling the competitors' bids strategies) to achieve top scores.

The Query Allocator of the agent's optimizer relies on the following key relation between the marginal utility of a unit sold and the total number of units sold (as a function of modeler-provided values of game state and hidden parameters)

$$U(m) = m\left(R - \frac{CPC(b(m))}{CVR}(1 + \frac{m_n}{m_b})\right) \ , \tag{3.2}$$

where m is the number of units sold to the users, R is the revenue associated to a unit sold, $U(m)$ is the total profit from selling m units, $b(m)$ is the bid level that results in m units sold, $CPC(\cdot)$ is the mapping of the bid level to the cost per click, CVR is the conversion rate and m_b and m_n are (respectively) the maximal (potential) number of *buying* impressions and the maximal number of *non-buying* impressions estimated by the modeler's particle filter. Indeed, the sales for a query are achieved by $m_b \cdot CTR$ clicks from the 'buying' population and (since the users are simulated in a random order) $m_n \cdot CTR$ clicks from the 'non-buying' population. Therefore, $m_b \cdot CTR \cdot CVR$ sales requires $(m_b + m_n)CTR$ clicks and the number of clicks required to achieve a sale is $\frac{1}{CVR}(1 + \frac{m_n}{m_b})$. We conclude that when m units are sold, the cost of making a sale is $\frac{CPC(b(m))}{CVR}(1+\frac{m_n}{m_b})$ and the relation (3.2) follows. Note that since $b(m)$ (the bid resulting in at least m sales) is a step function, $U(m)$ is piecewise linear in m. Moreover, the slope of $U(m)$ negatively depends on $CPC(b(m))$ and is therefore decreasing in m.

Making the relaxing assumption that the relation (3.2) is concave, the Query Allocator then uses the equimarginal principle to replace a multidimensional search over all bids combinations (e.g., as suggested in [28]) by a simple one dimensional search for the highest marginal utility level (equal over all queries) that achieves the daily quota of the optimizer's CQM (see Figure 3.9). To reduce variability and address the inherent uncertainty, our utility level optimization is regularized by the number of queries that take part in the bid (i.e., we give preference to utility levels that result in the daily quota spread over queries).

In what follows, each of the agent's components is described in detail.

3.4.1 Modeling CPC, Position and Bid Relation

The resulting cost and ad position given an advertiser's bid on a query depends on the bids (and spend limits) of the competing advertisers. Instead of modeling the

Figure 3.2: The estimation of CPC (left) and position (right) as a function of the bid

competitor's strategies, we make the simplifying assumptions that the game is in a stationary state (that is, the competitors strategies are fixed and do not evolve) and the CPC and the bid are linearly related. Therefore, we only need to maintain the ratio and the upper and lower thresholds (i.e., the bid threshold beneath which our ad is not shown at all, and the bid threshold above which our ad is the first ad shown).

The ratio estimation as well as the lower and upper thresholds are initialized based on previous games and are updated after each query report. Namely, when a bid lower than our minimal bid results in showing our ad we lower our minimal bid towards it. When a bid higher than our maximal bid doesn't result in our bid shown first we raise our maximal bid. The ratio estimation is updated by averaging with the previous estimate.

For the estimation of the resulting position given a bid we maintain *num_bidders*, an estimate of the number of bidders for each query. The number is initialized to the maximal value (i.e., 5) and updated after each query report by averaging with the previous estimate (this time a weighted average that prefers the latest estimate). As with the bid-cost relation, we assumed an underlying stationary system and linear relation and used the minimal bid and the maximal bid estimates. Figure 3.2 illustrates the CPC and position relations with the bid. Note that although the CPC and position are highly correlated, each quantity (specifically, it's relation to the bid) is separately estimated.

3.4.2 Hidden Game Parameters

In the TAC-AA game, the probability that a user clicks on an ad (the ad specified by the advertiser that won the query auction) is determined by a baseline CTR that is

unknown to the competitors (the game specification defines a known range for each query class - Level 2, Level 1 and Level 0 queries, pertaining respectively to the user's search focus levels F2, F1 and F0 described in Section 2.1 - from which the baseline CTR is uniformly sampled at the beginning of the game). The conversion probability, on the other hand, only depends on a known baseline CVR, the capacity utilization level, and the matching between the user's preferred component and the component element of the query. Therefore, assuming that the capacity utilization level is controlled, for a given user and query, the conversion rate may be directly calculated. Another hidden game parameter is the continuation probability, which is uniformly sampled form a specified range (a different range for each query class, but contrary to the baseline CTR the same value is used for all advertisers). The last two hidden game parameters that the modeler estimates are the reserve prices - one for regular slots (again, uniformly sampled from a known specified range, one for each query class) and another for promoted slots (the number of promoted ad slots, 0, 1, or 2, is communicated to the competing advertisers at game start). Estimates of the values of those hidden parameters are used to quantify the relation between the position of an ad and the resulting number of clicks and impressions (and subsequently, also the cost, revenue and utility to be expected upon a bid - the *Result set* dependencies of Figure 3.1). The exact method of estimating each of the hidden parameters is now detailed:

The CTR for every query q is determined by three factors: the parameter e_q^a (a baseline value randomly chosen by the server for each competing advertiser a at game start), whether the ad is placed in a promoted slot or not (f_{pro}), and whether the ad targeting matches the user population (f_{target}).

For games that have at least one promoted slot, we find e_q^a together with the two reserve prices ρ_{reg} and ρ_{pro} by considering the following three relations:

$$\rho_{pro} = \rho_{reg} + 0.5 \ , \quad cpc_r = \frac{\rho_{reg}}{(e_q^a)\chi} \ , \quad cpc_p = \frac{\rho_{pro}}{(e_q^a)\chi} \ ,$$

where cpc_r and cpc_p are the minimal cost per click observed on regular slots and promoted slots, respectively. The first (left) relation is a simplifying assumption since by the game specification we always have $\rho_{\text{reg}} \leq \rho_{\text{pro}} \leq \rho_{\text{reg}} + 0.5$. The second and third are due to the generalized second price auction, the minimal price that an advertiser would have to bid in order to to get his ad shown is the squashed[8] reserve price for the

[8] In a pay-per-click setting, *squashing* the bid - multiplying it by the related click probability -

minimal regular or promoted position, respectively. The squashing parameter χ is given at game start so we can easily solve the three unknown variables and get estimates for both reserve prices and for $e_q^a = (2(cpc_p - cpc_r))^{-\frac{1}{\chi}}$. As this value is an approximation, it is averaged with the previous approximation whenever recomputed. Now, given the bid level we can assess whether our ad will be placed in a promoted slot or not. This allows us to use a good approximation of the true f_{pro}. Finally, knowing our ad type and the relevant population type we set f_{target} and get the desired approximation of the click through rate. Now, for an estimate of the continuation probability we use the reported position and clicks and the modeler assessment of total impression *imps* (see Section 8.1) to solve

$$imps_{\text{eff}} = imps \cdot [\gamma(1 - CTR \cdot CVR)]^{position-1} \quad , \quad CTR = \frac{clicks}{imps_{\text{eff}}} \quad , \tag{3.3}$$

where the only unknowns are γ and $imps_{\text{eff}}$ (the effective number of impressions - the number of users that actually considered clicking our ad, after passing higher positioned higher ads and continue without clicking or converting).

For games with no promoted slots we calculate the CTR and continuation probability iteratively, by first applying (3.3) assuming the previous CTR (this give an updated γ estimate) and subsequently re-applying (3.3), now with the updated γ, to get an updated CTR estimate.

3.4.3 Particle Filters for Users Population Distribution Over States

As indicated above and from (3.2), $m_b^q(d)$ and $m_n^q(d)$ (specifically, their ratio[9]) are key in evaluating the utility of selling a unit by bidding on query q and as a consequence a key service used by the bid optimization algorithm. In this section, we describe the usage of particle filters to estimate $n_s^p(d)$, the number of users in state s for population p on day d (recall that there are 9 user populations, one for each combination of manufacturer and product type) for all states s and 9 populations p. Given estimates for n_s^p, calculating m_n^q and m_b^q for all 16 queries $q = (M, C)$ combinations of manufacturer M and component C (either M or C may be ϕ) is easy, using the following relations that are a direct consequence of the users behavior as described in the game specification:

normalizes the bids according to the expected revenue to the publisher.

[9] In 3.2 we omitted notation for the specific query q and day d. For clarity, in what follows we omit from the notation the dependence on the day d unless required to avoid ambiguity.

For each level-2 query $q = (M, C)$ we have

$$m_b^{(M,C)} = n_{F2}^{(M,C)}, \text{and } m_n^{(M,C)} = \frac{1}{3} n_{IS}^{(M,C)} \ ,$$

for each level-1 query $q = (M, \phi)$ or $q = (\phi, C)$ we have

$$\begin{aligned} m_b^{(M,\phi)} &= \frac{1}{2}(n_{F1}^{(M,C1)} + n_{F1}^{(M,C2)} + n_{F1}^{(M,C3)}) \\ m_n^{(M,\phi)} &= \frac{1}{6}(n_{IS}^{(M,C1)} + n_{IS}^{(M,C2)} + n_{IS}^{(M,C3)}) \\ m_b^{(\phi,C)} &= \frac{1}{2}(n_{F1}^{(M1,C)} + n_{F1}^{(M2,C)} + n_{F1}^{(M3,C)}) \\ m_n^{(\phi,C)} &= \frac{1}{6}(n_{IS}^{(M1,C)} + n_{IS}^{(M2,C)} + n_{IS}^{(M3,C)}) \end{aligned}$$

and finally, for the (only) level-0 query (ϕ, ϕ) we have (summing over the 9 populations):

$$m_b^{(\phi,\phi)} = \sum_p n_{F0}^p, \text{and } m_n^{(\phi,\phi)} = \frac{1}{3} \sum_p n_{IS}^p$$

We maintain a particle filter for each of the 9 populations p, each providing estimates of n_s^p for all states s. Before moving on to describe our specific particle filter implementation we provide a short overview of the particle filtering scheme as used in this context.

3.4.3.1 The Particle Filtering Method

Particle Filtering is a Monte-Carlo (simulation-based) method for estimating the Maximum Likelihood hidden sequence of states given a related series of observations. Specifically, given the transition model $p(x_k|x_{k-1})$ of a Markov Chain of hidden states x_k, and a series of observations y_k characterized by $p(y_k|x_k)$ as below:

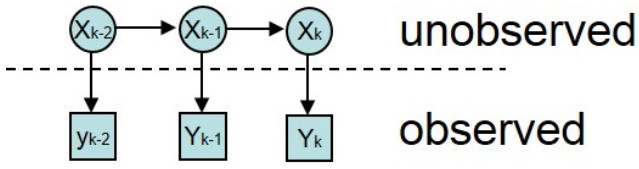


Figure 3.3: Hidden Markov Model

we recover the (computationally intractable in general) Maximum Likelihood series of hidden states by generating a population of N *particles* (where the value of each

particle s_k^i, $i = 1 \ldots N$ represents a possible value of the hidden state x_k at time k) and evolving each particle's value and weight according to the observations such that the population of particles collectively represent the posterior quantity of interest

$$s_k^i \sim p(x_k|y_1, y_2, \ldots y_k) \tag{3.4}$$

This is achieved using the Sampling-Importance Re-sampling (SIR) variant of particle filtering [14], by the following sequence of actions applied iteratively upon each new observation y_k:

First, each particle's value s_{k-1}^i is *advanced* to a new value s_k^i according to the Markov model $p(x_k|x_{k-1})$. Thereafter, a *weight* w_k^i is associated to each particle i such that w_k^i is proportional to $p(y_k|s_k^i)$ (the weight of a particle represents the plausibility of the observation given that the actual value of the hidden sate was the particle's value). Finally, using *Importance Sampling* [51] the population of particles is re-sampled by the weights to result in a new population that adheres to 3.4. As a result, an estimate of the hidden state x_k is made available by taking an average of their values s_k^i.

3.4.3.2 Applying Particle Filtering

In our setting, the state x_k of a particle represents a distribution of the user population[10] over the possible states. That is, the state x_k is actually a vector $(x_{NS}, x_{IS}, x_{F0}, x_{F1}, x_{F2}, x_{TR})_k$.

As mentioned above, the distribution represented by the particle filter is the weighted average of the distributions of its particles, and the weight assigned to each particle reflects the plausibility of an observed quantity y_k given its represented distribution. In our case, as suggested by [99], the observation y_k (due to lack of a relevant directly observed quantity) is an estimate of the total number of impressions that occurred upon level-2 queries by users of the corresponding population.

The particle filter algorithm is therefore the following: we maintain a separate set of particles for each day *yesterday*, *today*, and *tomorrow*. When a new day starts, the particle set for *yesterday* is discarded, and the previous particle set for *today* is used for creating the new particle set for *yesterday* by reweighing and re-sampling it (upon receiving the query report and having an estimate of the total number of impressions).

[10] we maintain a dedicated particle filter for each of the 9 user populations

The new particle sets for *today* and *tomorrow* are created by advancing the new particle set for *yesterday* once and twice respectively. All this creates an updated estimate of $n_s^p(d-1)$, $n_s^p(d)$ and $n_s^p(d+1)$ at every day d of the game, allowing the modeler to have estimates for m_b^q and m_n^q for days $d-1$, d, and $d+1$.

In what follows we review in more detail the particle filter update steps and related implementation concerns. Of special interest (w.r.t. our model free approach) is the usage of Nearest Neighbor learning to estimate the particle filter input (in contrast to the direct calculation described by [99]).

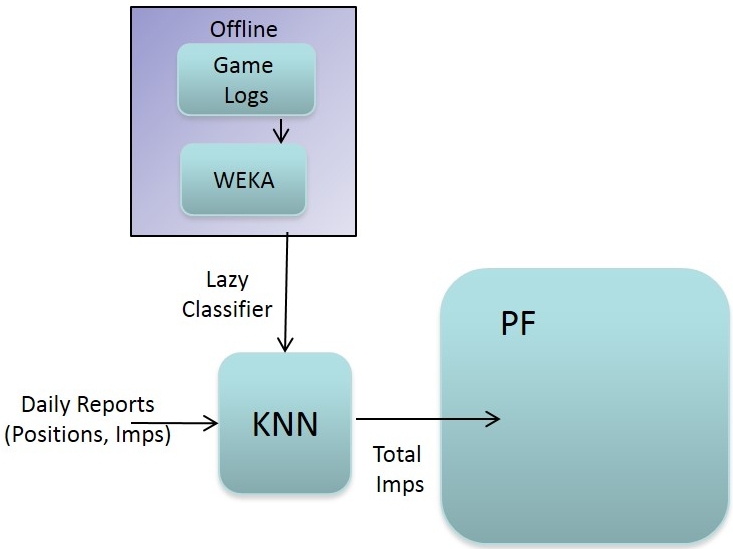


Figure 3.4: A K-Nearest Neighbor estimator trained off-line to provide the input observation (total number of impressions) to the particle filter.

3.4.3.3 Using KNN for Total Impressions Estimation

The input (observation) of the particle filter is the total number of level 2 queries impressions during a simulated day. This quantity is not directly provided to the agent, however. Therefore, a dedicated algorithm (presented by the TacTex team in [99]) may be used to find a value that is consistent with the reported data.

Instead of using TacTex's algorithm, we use a Nearest Neighbor estimator that uses training samples from past games logs (associating reported average position for each of the competitors and the position and number of impressions of the agent, to the sought after total number of level 2 impressions) to train a weka-based [64] K-Nearest-Neighbor estimator that is then used to provide (on-line, given the reported data) an estimation of the total number of impressions to the particle filter. This scheme is

illustrated in Figure 3.4:

Using 200000 samples with $K = 3$ resulted in an average relative error of 30% (compared to the 'exact' computation of the total number of impressions - see Figure 3.5). However, as indicated below, this minimal influence on the overall performance of the particle filter (and subsequently, no significant effect on the overall agent's performance).

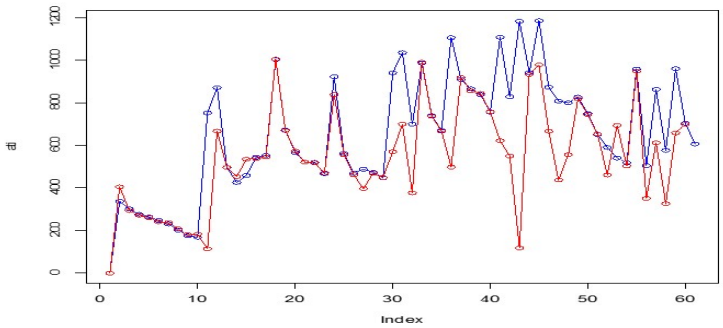


Figure 3.5: The estimated total number of impressions using K-NN (red line), compared to the actual number (blue line) throughout the 60 simulated days of a game.

3.4.3.4 Reweighting

Given an observation of T estimated total impression, the weight $w(P|T)$ assigned to a particle representing a users distribution

$$P = (N_{NS}, N_{IS}, N_{F0}, N_{F1}, N_{F2}, N_{TR})$$

is computed as the probability of a total of $T - N_{F2}$ successes in N_{IS} binomial experiments, each with success probability $\frac{1}{3}$ (this is because each of the N_{F2} users results in an impression for the related L2 query, and with probability $\frac{1}{3}$ each of the N_{IS} users results in an impression for the related L2 query - see Figure 3.6).

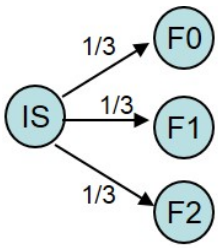


Figure 3.6: Expected number of F2 queries is $\frac{N_{IS}}{3} + N_{F2}$

In practice, we use the normal probability approximation for the distribution with expected value $\frac{N_{IS}}{3}$ and variance $\frac{2N_{IS}}{9}$ and set

$$w(P|T) \triangleq Pr(T|P) = \phi(\frac{3(T - N_{F2}) - N_{IS}}{\sqrt{2N_{IS}}}) \ , \tag{3.5}$$

where $\phi(\cdot)$ is the normal probability density. Upon re-weighting, the weights are normalized such that they sum to 1.

In some situations it may happen that the probability (3.5) is negligible for all particles. This may be caused by estimation errors of the total number of impressions (the particle filter *observation*) or by competitors behavior: if all advertisers reach their spending limit within the day then the total number of impressions is no longer equal to the total number of users in searching states, a condition that violates a fundamental assumption of the derivation of (3.5). A naive computation of (3.5) in such cases results in zeroing of the weights of all the particles, a situation that should be avoided to enable the subsequent resampling (that depends on the weights summing to 1). Therefore, the reweighing algorithm avoids a direct computation of (3.5) and instead computes for each particle the ratio of its probability to the maximal particle's probability (and subsequently normalize such that the weights sum to 1).

3.4.3.5 Resampling

Given a re-weighted particle set (the *baseline* set), re-sampling involves creating a new particle set in which the number of times each particle appears in the new set (the *re-sampled* set) is relative to its weight in the baseline set. Once re sampled, the weights are discarded and weighted averages over the baseline set are equivalent to uniformly averaging over the re-sampled set. We implement selective re-sampling (only reweigh and re-sample a randomly chosen portion of the particles[11], leaving the rest of the particles unchanged regardless of the observation). This allows for quick readjustment in case of estimation errors.

3.4.3.6 Advancing Particles

Advancing a set of particles consists of advancing each particle of the set, simulating the daily change of state of the users. A particle representing a users distribution

[11]The portion of the particles that is kept unchanged depends on our level of confidence in the observation

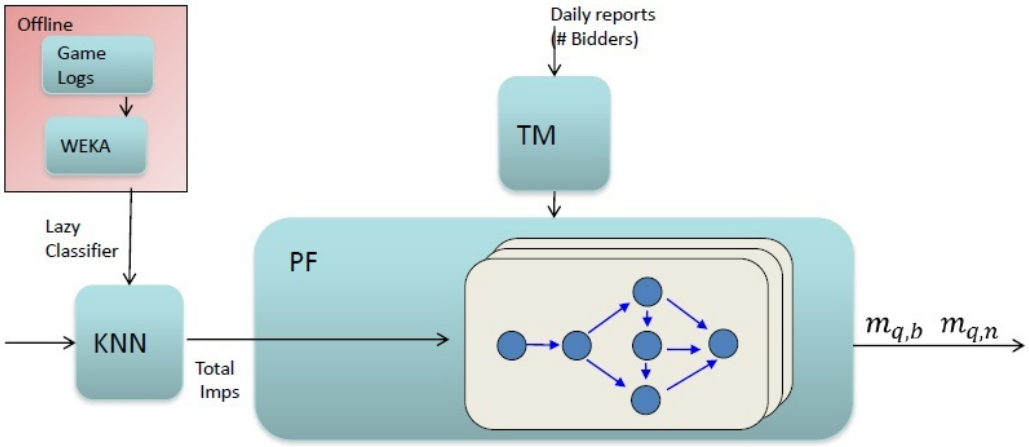


Figure 3.7: Modeler's Particle Filter Architecture.

$P = (N_{NS}, N_{IS}, N_{F0}, N_{F1}, N_{F2}, N_{TR})$ is advanced by applying a transition model (which defines the state transition probabilities) to the represented user population, resulting in an *advanced* users distribution $P^a = (N^a_{NS}, N^a_{IS}, N^a_{F0}, N^a_{F1}, N^a_{F2}, N^a_{TR})$. The transition model is given as part of the game specification, and is constant except for the transitions from focused searching states to the transacted state (which depend on capacity usage levels, and therefore is no longer Markovian). Furthermore, the transition probabilities depend on the presence of a burst (effecting the probability of transitioning from *NS* to *IS*, an effect may last for a few simulated days) and therefore each particle also maintains a burst-status which is used to select the appropriate transition model to use.

The particle advance algorithm is the following: First the appropriate transition model (*burst* or *regular*) is selected - this is a random choice (the probability of a burst depends on the current burst status). Second, the users N_S of each state S are transitioned according to the transition model (we use successive binomial sampling to implement a multinomial random generator, and we compute the conversion probability based on estimates of the number of bidders for each query). Finally, N^a_S (the advanced population of state S) is set as the aggregation of all users transitioned to state S. The overall Particle Filter part of the modeler's architecture is illustrated in Figure 3.7: Modeler architecture: particles are advanced using the transition model (TM) which is regularly updated with the recent estimates of the number of bidders for each query. Also, a K-Nearest Neighbor estimator that was trained off-line is used to provide the

input (total number of impressions) to the particle filter. The estimates of users at states are used to assess for each query q the total potential queries $m_{q,b}$, $m_{q,n}$ in converting states and non-converting state (respectively).

3.4.3.7 Performance

Empirical evidence (that is, several competitions in which both methods of estimating the total impressions for the particle filtering input were employed) suggests that using the Nearest Neighbor method for Particle Filter input (instead of exact computation based on game specification) has negligible effect on the agents performance overall.[12] The estimates of the particle filter regarding the number of users in two of the states are illustrated in Figure 3.8, where the red lines are the estimates based on the KNN-estimated inputs and the blue lines are the true figures.

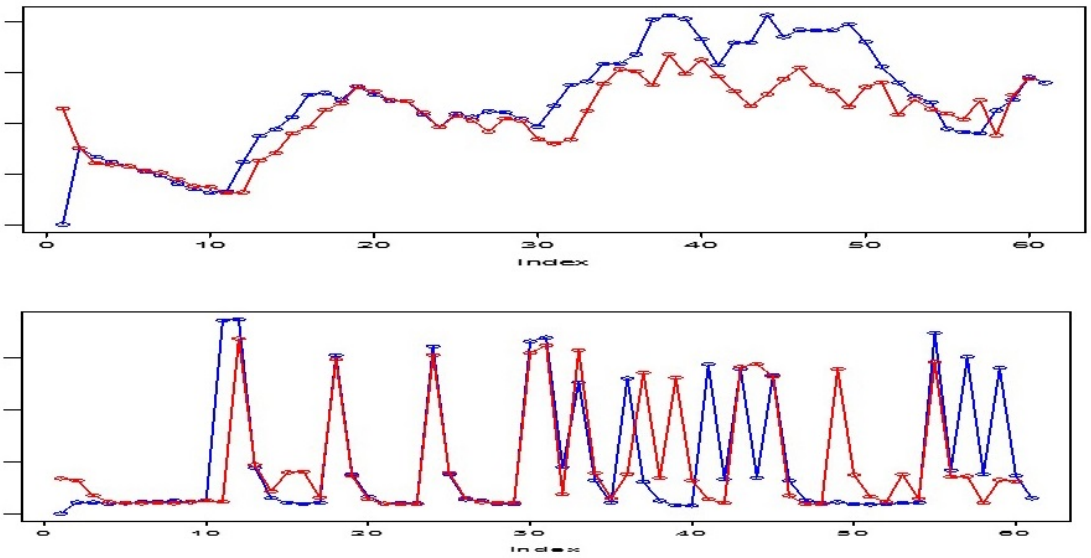


Figure 3.8: Particle Filters' estimates for the F2 state (top) and IS state (bottom). The horizontal axis is the game time (logical day), and the vertical axis is the number of users in the state.

3.4.4 Optimizer's Query Allocator

Consider the relation (3.2) of the total utility $U(m)$ resulting from selling m units through some fixed query q[13]. First note that not every value of m is sale-able. This

[12] This is based on scores variation similar to those resulting from the stochastic nature of the setting. Statistical significance tests were not performed, however.

[13] To simplify the notation, the query is omitted when not essential.

is because the number of units sold depends on the resulting position in the auction and there are only 5 positions possible (pertaining to 5 bid ranges and resulting in only 5 possible values of m). Nevertheless, since the estimated relation of bid to position is inherently inaccurate (see Section 3.4.1), we relax the discretization and assume a monotone continuous piecewise linear relation $b(m)$ and its inverse $m(b)$. Similarly, $CPC(b(m))$ (the cost per click when selling m items) is assumed to be piecewise linear m and we conclude with an approximated shape of $U(m)$, which is schematically illustrated in the left part of Figure 3.9:

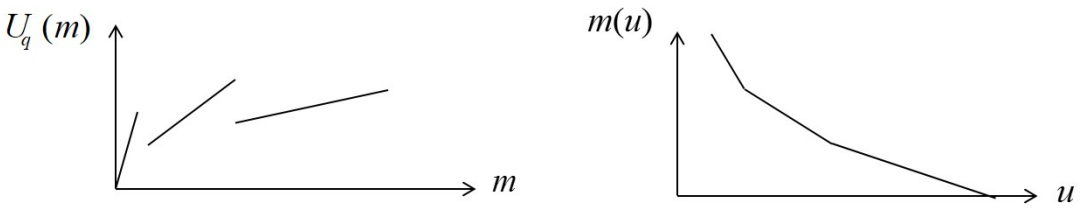


Figure 3.9: Schematic forms of $U(m)$ and $m(u)$.

For low values of m the cost of selling is low and therefore the marginal utility (derived from (3.2))

$$u(m) = R - \frac{CPC(b(m))}{CVR}(1 + \frac{m_n}{m_b}) \tag{3.6}$$

is high. Upon crossing a threshold, the marginal utility drops and therefore the total utility may drop (but increase with m thereafter). Note the somewhat convex shape (as m grows the marginal utility might become negative). Again, $u(m)$ may be approximated as a piecewise linear function of m with increasing slopes (since $b(m)$ is monotone in m). Its inverse $m(u)$ may be therefore approximated to have a somewhat concave shape[14], as illustrated in the right part of Figure 3.9.

Now, the optimizer's Query Allocator's task is to find at every day t an optimal (highest profits) bid bundle for day $t+1$ that achieves the daily allocation $\tilde{Q}_{t+1}$ of (3.1). Using (3.6) the optimizer may derive the result set (specifically the bid b and number of conversions m) pertaining to a target marginal utility u. This may be done by first solving (3.6) to recover the CPC (the rest of the variables are known or estimated by the modeler), then using the monotone relations maintained by the modeler to find the associated bid and target position, and finally using the game parameters maintained

[14]Note that a decrease in the utility per unit sold occurs when the related cost per unit sold increases (reflecting a better ad position for the query), which leads to a higher number of conversions. Therefore $m(u)$ decreases with u.

by the modeler to get an estimate of the total impressions, clicks, and conversions[15].

We can now formalize the optimizer problem of finding the optimal bid bundle subject to the capacity constraint as the following program:

$$\max_{\{b_q\}} \sum_q U_q(m_q(b_q)) \ , \quad \text{subject to} \sum_q m_q(b_q) \leq \tilde{Q} \ , \tag{3.7}$$

where $m_q(b_q)$ is the assumed number of units to be sold when bidding b_q on query q, and $\tilde{Q}$ is the daily sales quota as set by (3.1). Now, making the simplifying assumption that $U_q(\cdot)$ are concave[16] for every q, it is easy to see that for an optimal solution $\{m_q^*\}$ of (3.7) all the marginal utilities $u(m_q^*)$ are equal.

Therefore, the optimization problem (3.7) is solved by a simple linear search for the maximal marginal utility u^* for which the resulting total sales achieve the quota:

$$\{(m_q(u*)\} \ \text{ such that } \ u^* = \max_{\sum m_q(u) \geq \tilde{Q}} u \ ,$$

Our simple search for the optimal utility[17] starts at a predefined constant high utility level u_h and decreases it repeatedly (in Δ sized steps) until our total estimated sales reaches the target daily allocated quota (or until a predefined constant low utility u_l is reached - this is to avoid the risk of negative utility, and to ensure that our algorithm stops even in cases where the total potential sales are lower than the allocation).

The optimization's algorithm pseudocode alongside an illustration of this simple search method follows:

[15] Using the modeler assessment of total impressions, the estimation of the continuation probability, and the estimates of click and conversion probabilities to calculate (respectively) the effective number of impressions, total clicks, and the number of conversions when bidding for the target position.

[16] As discussed before, $U_q(m)$ is of course not concave, since $U_q(m)$ has a discontinuity precisely at the discontinuity points of $b(m)$.

[17] This could also be done faster using binary search.

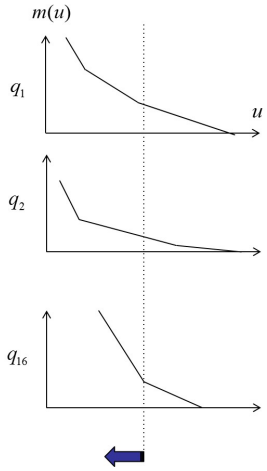
$u \leftarrow u_h$

while $(\sum_q m_q(u) \leq \tilde{Q})$ AND $(u \geq u_l)$ **do**

$u \leftarrow u - \Delta$

end while

return $\{b_q(u)\}$



Algorithm 1: Optimize $(\tilde{Q}, \{m_q(\cdot)\}, \{b_q(\cdot)\})$

Finally, regularization was added to preclude quota underutilization:[18] The regularization is based on the perplexity of the allocation.[19] Denote by $\overline{m}(u) \in \Delta_{16}$ the normalized vector of units allocated to the queries. The perplexity of $\overline{m}$ increases with the number of queries taking part in the allocation represented by $\overline{m}$ (and the other way around). Now, in the regularized optimization algorithm we discount the estimated sales for a given utility level using a logistic function of the allocation perplexity. Specifically, by a factor $\frac{1}{1+\beta \cdot e^{-p(\overline{m})}}$. We used a parameter β to tune the regularization degree, where a high β represents a high preference to spreading the allocation across queries. Now, as u decreases, $m_q(u)$, the perplexity $p(\overline{m}(u))$, and the regularization factor increases. Therefore, our regularized algorithm (which is identical to Algorithm 1, but with $(1 + \beta \cdot e^{-p(\overline{m}(u))})$ multiplying $\tilde{Q}$ in the while condition) will result in the highest uniform utility level that achieves a regularized quota (higher - since the regularization factor is smaller than 1).

3.4.5 Results

The additions to our **tau** agent proved effective. Our agent reached 3rd place in the 2011 TAC-AA competition finals, scoring within 2% below the winning agent (*TacTex* winning again, although in 2010 the winner's margin was 10%, significantly higher). The *tau* agent for the 2012 TAC-AA competition included minor enhancements, mainly tuning some of the agent's parameters based on logs from the 2011 competition. The agent did very well in the semifinal rounds, reaching the top position. However, in the final rounds (where only the best 8 scoring agents from the semifinal rounds participate)

[18] resulting from a combination of sales estimation errors and allocation to very few queries

[19] The perplexity of a probability distribution $\overline{d} \in \Delta_n$ is defined as $p(\overline{d}) = 2^{H(\overline{d})}$, where $H(\cdot)$ is the entropy

it reached 3rd place (again, scoring within 2% below the winning agent, this time *Mertacor*) as in the 2011 competition. The significant difference in ranking between the semifinals and finals is typical of such settings in which the quality of a strategy depends on the strategies of the other competitors. In the case of the TAC-AA competition, an agent that did not make it to the final rounds might be influencing more one agent than another, and once removed the ranking of scores of the remaining agents could significantly change. Therefore, the results of the TAC-AA 2012 competition further reinforces the game nature of the setting and the notion of a winning strategy which is only relevant with respect to the strategies of competing agents. This conclusion is further supported by the results of the TAC-AA 2013 competition, in which (assuming minor modifications to the other competing agents, *tau* was again minimally tuned) the top three positions were shared by the same top three agents from the 2012 finals, this time with our *tau* agent finally winning. Indeed, since the scores of the top performing agents (both in 2012 and in 2013) were so close, it might very well be that minor tuning and readjustments (although not resulting in significant score increase) made a difference with respect to the rankings, resulting in very similarly-performing agents switching the top position in consecutive years.

3.5 Limitations of Machine Learning Models

TAC-AA agents employ different Machine Learning techniques trying to estimate different game parameters and states. With the goal of assessing the benefit that agents can obtain from improving their Machine Learning components, we modified the game server so it will send to one of the agents some of the unobservable parameters the agents try to learn, so this agent has a perfect knowledge of them. Our results indicate that even though Machine Learning models are inherently inaccurate, eliminating their error rates completely has only a minor effect on the performance of the agent. Therefore, we can speculate that improving these models is not likely to increase the agents' score. We can deduce that the global performance of a learning system might not improve significantly even if all errors are eliminated. This is an excellent example where even significant improvements in the accuracy of the ML components would have a diminishing effect on the overall performance of the system.

Using our top performing **tau** agent, we modified the game server to send two types

of information to our agent:

1. Parameters - The exact values of several unobservable parameters: advertiser's effect (e_a^q), continuation parameter and reserve prices.
2. Users distribution - The number of users in each state, for each query type.

We tested the effect this information had on our agent's performance in a competition against the agents mentioned in Section 4.2. This information obviates the need for certain ML models, and in fact simulates the use of perfect models. Hence, these tests enable us to assess the additional profit one could hope to gain by improving these ML models.

In the first experiment we modified our agent to receive only the parameters from the server. These parameters enable it to make better assessment of Cost Per Click (CPC), Click-Through Rates (CTR) and effective number of impressions.

In our original agent, these parameters were estimated using iterative maximization methods, with an error rate of about 15%. Therefore, this information is a major estimation improvement. However, the average score of this agent was improved only by 1%, an improvement which is not statistically significant.

In the second experiment we modified our agent to receive only the users distribution from the server. Thus, the modified agent had perfect knowledge of the users' state in each time. In our original agent, this task was carried out using K-Nearest Neighbor approach, which was the input to a Particle Filtering model. The error rate of the K-NN model was 25%.

The score improved by 2%, which is not statistically significant ($p = 0.2$). This result implies that despite the high error rate of the K-NN model, the particle filter model creates good enough estimations of the users' state, which cannot be improved drastically.

Finally we modified our agent to receive both types of information. This agent had all of the information it needs in the game, except for the bid bundles of its opponents.

In this setting, the agent's average score was improved by 5%, an improvement which is statistically significant ($p < 0.05$). This result is somewhat surprising when considering the minor effect that each set of information had by itself. It seems that improving only one model has little contribution, but having perfect models in both domains is more significant.

The results of these three experiments suggest that improving the accuracy of the

ML models employed in TAC-AA has a limited contribution to the final result. Presumably, the high error rate of the ML models does not lead to impaired performance, and this error rate is somehow overcome during the optimization process of the agent.

3.6 Conclusion

Using a very simple model-light approach to implement our first TAC-AA agent (that is, minimal modeling of the game actual parameters and competitors behavior) resulted in a relatively high performing agent. Furthermore, due to its simplicity, the agent was implemented in a very short time and required minimal fixes and debugging. This agent used a very limited action space (only bidding to win the first position in a favorable subset of the available queries). To reach the top-scores, however, a key component (and state) of the game - the distribution of simulated users across searching states - had to be modeled. The user's distribution was modeled in our *tau* agent (as in other competing agents) using a particle filter method. Our agent, however, managed to implement the particle filter while avoiding a tailored reverse-engineering of the game specification and using instead the model free nearest neighbors learning algorithm for the estimation of the particle's filter inputs. Our top performing agent also avoided modeling the competitors by simple estimators for the cost and resulting position upon a certain bidding level. Our *tau* agent eventually won the 2013 TAC-AA competition after reaching third position in both the 2011 and 2012 TAC-AA competitions. We further showed, by modifying appropriately the TAC-AA game server, that no real benefit is expected from improving our simple learning models. This is probably due to the inherent unpredictability of the TAC-AA setting.

Chapter 4

An Empirical Study of Agent Robustness

We study the empirical behavior of trading agents participating in the Ad-Auction game of the Trading Agent Competition (TAC-AA). Aiming to understand the applicability of optimal trading strategies in synthesized environments to real-life settings, we investigate the robustness of the agents to deviations from the game's specified environment. Our results indicate that most agents, especially the top-scoring ones, are surprisingly robust. In addition, using the game logs, we derive for each agent a *strategic fingerprint* and show that it almost uniquely identifies it. An extended abstract of this chapter appeared in [66].

4.1 Introduction

As mentioned in Section 3.1, online advertising through sponsored search results has become a multibillion dollar business in the past years (see also [53, 77, 118]). In this form of advertising, query specific advertisements are placed alongside organic search-engine results. Sponsored search has been the object of a considerable amount of research, both from the publisher and the advertiser's perspectives.

The Ad-Auction (AA) game in the yearly Trading Agent Competition (TAC), as described in Section 2.1, presents a sponsored search scenario that employs an ad auction mechanism and a structured model of users [70]. Competitors in this game implement retailers that aim to maximize their profit through the use of sponsored search advertising. This setting facilitates research of agent strategies in a multi-agent

competitive environment. Furthermore, it can be used to draw more general conclusions about ad-auction mechanisms and sponsored search [69]. Ad Auctions games have been held since 2009, and in the course of time the agents improved their performance by employing complex techniques and strategies [27, 98, 101].

The main goal of the work reported in this chapter was to understand the applicability of TAC-AA to real world settings. The game specifies a synthetic environment, and the agents are developed to take advantages of various features of this environment. It is reasonable to expect, in that case, deteriorated agents' performance when confronted (unaware) with a different environment. Moreover, if agents strategies are over-fitted to TAC-AA, then the agents with higher TAC-AA performance are expected to suffer a bigger degradation. To be successful in real settings, trading agents should tolerate higher levels of uncertainty (compared to the synthetic and simplified TAC-AA scenario). Therefore, robustness of TAC-AA's top-performing agents to game variations serves as evidence for the applicability of TAC-AA agent's strategies to real environments.

Consequently, our goal was to test whether the agents (especially the top-performing ones) can adapt to a different environment and still perform well, as expected from agents in a real world. To that end, we modified the game parameters and tested the effect of this modification on the performance of some recent TAC-AA agents. Although (as expected) most of the agents are tailored to the specific game parameters, we show that the top performing agents perform well even when the parameters are changed and exhibit robustness. This result suggests that TAC-AA may indeed serve as a test-bed for addressing real-life scenarios, and that techniques used by top performing agents may potentially be applied in the real world.

Another objective of our research is to define a *strategic fingerprint* of a TAC-AA agent and characterize its behavior. To achieve this objective, we define several observable attributes that are calculated from the game logs for each agent in each game, and we incorporate them into an attribute vector we call a strategic fingerprint. We show that this strategic fingerprint identifies agents, and is also well correlated with their profit. Therefore, this fingerprint can be used to design better TAC-AA agents. In addition, it reflects the vulnerability of simple log anonymization, and demonstrates that it can be overcome using simple ML tools.

Table 4.1: The results of the benchmark competition and experiments 1 - 4 from Section 4.2. The numbers next to the agent name indicate the year in which this agent participated in the TAC-AA finals.

Agent	Benchmark	Ex. 1.1	Ex. 1.2	Ex. 2.1	Ex. 2.2	Ex. 3.1	Ex. 3.2	Ex. 4
TacTex10	58,146	67,627	61,294	50,903	62,544	53,578	61,866	36,737
tau11	58,124	64,187	61,107	52,175	61,983	54,406	61,013	49,339
TacTex(2)10	57,929	67,078	61,369	49,639	63,164	54,063	62,656	37,880
Mertacor11	55,716	40,710	53,576	44,349	51,653	51,930	51,546	54,033
Schlemazl10	55,413	62,766	60,952	51,323	59,553	53,145	59,246	47,139
Crocodile11	50,735	51,456	50,521	44,682	54,700	45,593	53,369	40,386
tau10	49,699	49,381	49,145	43,330	52,735	44,271	50,617	39,292
EpflAgent10	45,886	34,648	47,564	38,933	51,042	41,565	49,330	40,836
Median	55,565	57,111	56,032	47,161	57,127	52,538	56,308	40,611

4.2 Robustness of TAC-AA Agents

In order to assess the robustness of TAC-AA agents, we ran several experiments in which we varied some of the game hidden parameters (i.e., parameters which are not revealed to the agents at the beginning of each game), and ran a standard 48-game competition in each new setting. The agents tested are agents from the TAC repository, who competed in the Ad Auction finals in 2010 and 2011. To complete the set to the required eight agents, we used two copies of one agent, TacTex (this also enabled us to estimate the effect of the game randomness on an agent's profit).

Since this combination of agents never participated in a public TAC-AA competition, a benchmark competition was first held. The results of this competition as well as the results of our first four experiments are detailed in Table 4.1.

For each experiment, we compared the score of each agent to its score in the benchmark competition, and noted the difference in the agent's position. We ran t-tests with 0.95 confidence level to find the statistical significance of this difference. We also compared the median score in each experiment to the median score of the benchmark competition, in order to understand the general effect of the changes we made.

Our results show that most agents, especially the top performing ones, are robust to changes in the game parameters, although they overfit to TAC-AA parameters to some extent.

4.2.1 Experiment 1 - Users Model

In the following experiments we modified the users model, i.e., we changed the transition probabilities in the users state machine, in order to differentiate between agents who rely heavily on the exact game parameters and agents that do not.

Ex. 1.1: We increased the transition probability from Non Searching mode (NS) to Informational Searching mode (IS) by a factor of 5 (from 0.01 to 0.05).

We hypothesized that this change will have a strong general effect, i.e., that it will increase the median score of the competition, since it increases the number of users which see ads, click on them and convert. We also expected that it will affect all agents similarly, and expected only a mild change in the relative positioning of the agents.

The median score of this competition was 57,111, which is significantly higher than the median of the benchmark competition. In addition, this change had a very different effect on different agents: while it increased the score of the top-performing agents, TacTex, tau11 and Schlemazl by 10-15%, it decreased the score of EpflAgent and Mertacor by about 25%. These differences were found statistically significant ($p < 0.01$). The other two agents maintained their old score - tau10 and Crocodile.

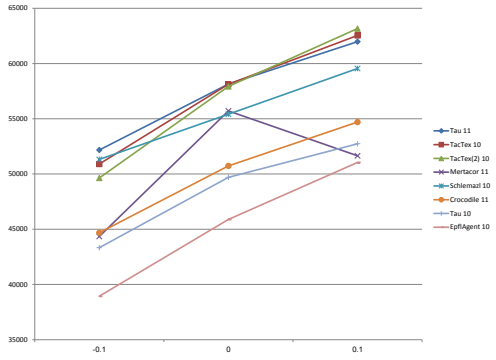


Figure 4.1: The results of experiment 2, where the advertiser effect is modified by ±0.1 .

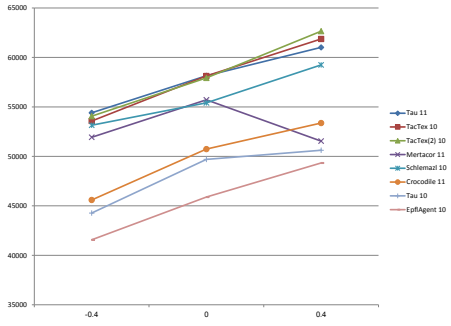


Figure 4.2: The results of experiment 3, where the conversion rate is modified by ±0.04 .

Ex. 1.2: We slightly modified the users' transition matrix, but this time we changed the transition probability between the different focus level searching modes. Hence, the overall number of searching users did not drastically change, but the distribution of the users between the different focus levels changed.

We hypothesized that this change will differentiate between agents who heavily rely

on the TAC-AA parameters and agents whose users models are more adaptive.

The median score of this competition was slightly higher than the median of the benchmark competition. This experiment had a milder effect on most agents. It significantly increased only the score of Schlemazl (by 10%, $p = 0.04$), while the scores of TacTex, tau11 and EpflAgent increased by about 5%, which was not found statistically significant. It also slightly decreased the score of Mertacor (by 4%, $p > 0.1$), as well as the scores of tau10 and Crocodile.

Combining the results of the two experiments, we can conclude that most agents are quite immune to changes in the users model. Specifically, TacTex, tau11 and Schlemazl seem less dependent on the exact values of the user transition probabilities, while Mertacor seems dependent on these values.

4.2.2 Experiment 2 - Click-Through Rate

In order to change the users' click-through rate, we modified the range from which the advertisers' effect (e_q^a) is drawn. This parameter is the baseline probability that a user will click a generic ad of advertiser a, shown in query q. This probability can be modified by a targeting factor, for a targeted ad, and a promotion factor, for a promoted advertisement slot.

We ran two experiments - one in which we increased the expectation of e_q^a by 0.1 for all focus levels[1] (Ex. 2.1), and another in which we decreased it by the same amount (Ex. 2.2). We expected that increasing this parameter will increase the overall revenues and vice-versa, since increased advertiser effect will result in more clicks and therefore more conversions.

The results of these experiments are shown in Table 4.1, as well as in Figure 4.1. As expected, decreasing the advertisers' effect reduced the median score (by 15%), while increasing this effect raised the median score (by 3%). This effect was similar for most agents, except for Mertacor. The effect on all agents was found statistically significant ($p < 0.001$).

When examining the graph in Figure 4.1, we can clearly see that most agents exhibit a similar performance in some sense - their score in the benchmark competition

[1] The values of these parameters are originally drawn uniformly at random from the ranges $[0.2, 0.3]$, $[0.3, 0.4]$ and $[0.4, 0.5]$ for focus levels $F0$, $F1$ and $F2$, respectively.

exceeds the average of the low and high CTR experiments. This excess is due to overfitting the agents to the game parameters. As expected, we can see that almost all agents are optimized to the game parameters. However, the degree of this optimization varies drastically between agents. The most overfit agent is Mertacor - its score in the benchmark competition is higher by 14% than the its average score in the two CTR experiments. TacTex and tau11 show some overfitting (an increase of about 2%), and the other agents show very little overfitting.

Thus, we can conclude that most agents are robust against changes in the click-through rate, despite a slight overfitting to the game parameters. This result is not so surprising when keeping in mind that this rate is not known to the agents and that they estimate it during the game.

4.2.3 Experiment 3 - Conversion Rate

In these experiments we modified directly the Conversion Rate by 0.04,[2] in both directions (Ex. 3.1 & Ex. 3.2). The original parameter is known in advance to the agents, so we can assume that they all rely on it in their optimization.

We expected that changing this parameter will have a direct and similar effect on all agents, i.e., that an increased conversion rate will lead to higher scores.

The results of these experiments are shown in Table 4.1 and in Figure 4.2. As expected, decreasing the conversion rate reduced the median score (by 5.5%), while increasing it raised the median score (by 1.5%). This effect was similar for most agents, except for Mertacor whose score dropped in both scenarios by about 7%, an effect which was found statistically significant ($p < 0.01$).

As in the previous experiment, this setting also allows us to measure the overfitting of the agents to the exact value of CVR, by comparing the agent's benchmark score to the average of its scores in the two experiments.

In this experiment we see that most agents do not exhibit overfitting to the exact CVR. The only agents whose score exceeds the average significantly are Mertacor (by 7%) and tau10 (by 4.5%). This result is surprising since in TAC-AA the exact value of the CVR is a constant known to the agents, and so we expected that agents will be optimized to it.

[2] The original conversion rates are 0.11, 0.23 and 0.36 for focus levels $F0$, $F1$ and $F2$, respectively.

4.2.4 Experiment 4 - Single Ad

In this experiment we reduced the number of advertising slots from 5 to 1, to simulate a banner-based advertising. This change is rather dramatic, and we expected that it will reduce drastically the median score. We also hypothesized that all agents will be affected in a similar way, since they all are optimized to multi-ad setting, where an agent can manipulate its position in order to optimize its sales and CPC. In the banner setting this flexibility is considerably diminished.

As we expected, the scores of all the agents dropped, and the median score was lower by 27%. However, the agents were not similarly affected - while the score of Mertacor was reduced only by 3% ($p > 0.2$), the score of other agents (tau10, tau11, Schlemazl and Crocodile) dropped by about 15% and the score of TacTex dropped by 35%. The latter changes were found statistically significant ($p < 0.01$).

This experiment differentiates between agents who aim at lower positions and agents who aim at higher positions. The former agents are more affected by the elimination of these spots. Unlike the previous experiments, Mertacor was the most robust to this change, presumably since it aims at higher positions at all settings.

4.2.5 Experiment 5 - Population Size

In this set of experiments we varied the number of users in each product population, from 2000 to 20000. The population size in the original competition is 10,000 and is known to be fixed. Many agents employ Particle Filtering in order to estimate the users distribution, and knowing the exact population size is a precondition for this technique.

The results of these experiments are shown in Figure 4.3. Due to the capacity limit in TAC-AA, increasing the number of users does not increase the score significantly. However, reducing the number of users damages the performance of all agents drastically. The median score for increasing the number of users was approximately unchanged, while for decreasing the median score deteriorated quickly from -15% for 8000 users until a decrease of more than 80% for 2000 users. These decreases were found statistically significant ($p < 0.04$).

It is no surprise that the TAC-AA agents are not robust against changes in the population size. This size is known in advance and it's an important factor in the optimization process. Furthermore, it is reasonable to assume that most agents try to estimate the number of users in each state, and this estimation is based on a constant

population size.

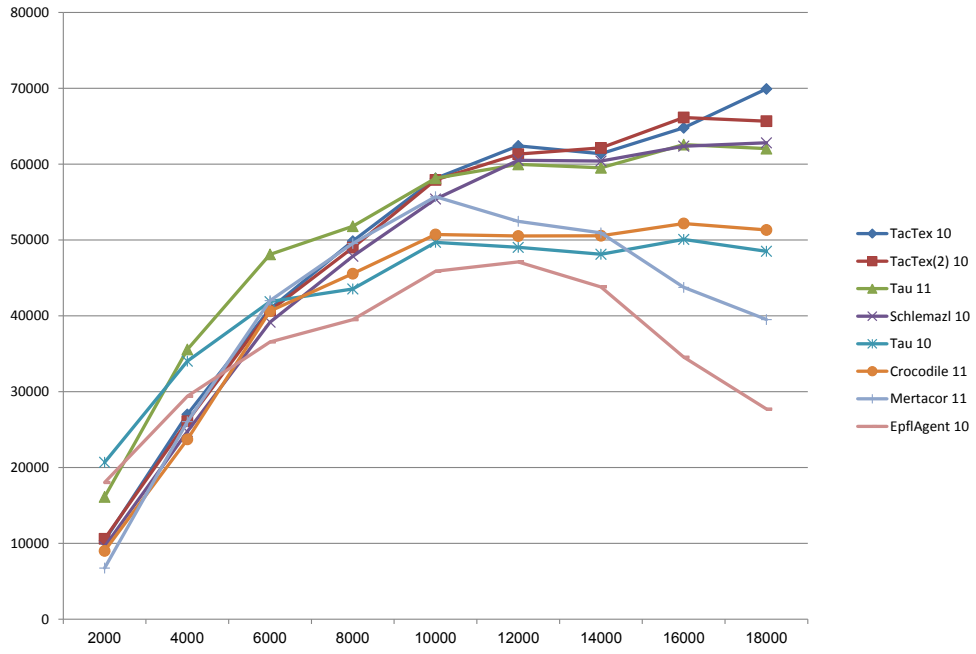


Figure 4.3: The results of experiment 5, where the users population size varies from 2000 to 20000.

4.2.6 Conclusion

Our experiments show that most of the TAC-AA agents adapt well to different settings, despite being optimized to the exact game parameters. The top performing agents of TAC-AA - TacTex, Schlemazl and tau11 - are rather robust to most changes, but when the setting is changed drastically, as in experiments 4 and 5, their performance deteriorates.

This robustness result is somewhat surprising, since one could expect that the top performing agents in the TAC-AA would be more optimized to the exact game parameters and thus will be more affected by changes in these parameters (as is the case of Mertacor). However, the experiments show that most agents are less over-fit to the game parameters than expected.

4.3 Agents Behavioral Identification

Using machine learning methods, we show that carefully chosen behavioral features may be used to identify a competing agent and to predict its profit.

4.3.1 Strategic Fingerprint

In order to characterize an agent's strategic behavior, we use several attributes extracted from the games' logs, to form *strategic fingerprint* vectors. These vectors identify the agents, as each agent's strategic fingerprint vectors are in a different region in space. In addition, these strategic fingerprints are a good predictor of an agent's profit in a game.

Queries are naturally grouped into the three focus levels, and we further split them into specialty and non-specialty groups. Therefore, we use 5 distinct groups: $F2$ & $F1$ specialty queries, and $F2$, $F1$ & $F0$ non-specialty queries. The attributes we use are:

1. **Query distribution of impressions, clicks and conversions:** We average across all days the relative part of impressions, clicks and conversions that the agent got from each of the abovementioned focus-specialty groups. Namely, for each focus-specialty group g, we compute:

$$imps\_percent_g = \frac{1}{60} \cdot \sum_{d=1}^{60} \frac{\sum_{q \in g} imps_q^d}{|g| \sum_q imps_q^d}$$

$$clicks\_percent_g = \frac{1}{60} \cdot \sum_{d=1}^{60} \frac{\sum_{q \in g} clicks_q^d}{|g| \sum_q clicks_q^d}$$

$$convs\_percent_g = \frac{1}{60} \cdot \sum_{d=1}^{60} \frac{\sum_{q \in g} convs_q^d}{|g| \sum_q convs_q^d}$$

Since we measure only the percentage of impressions, clicks and conversions for each focus-specialty group, this attribute reflects only the way an agent distributes its budget across queries, and not the actual number of impressions, clicks and conversions. For example, using this attribute we can see whether an agent places ads only on his specialty products, or also on other, less profitable ads.

2. **Average ad position:** The average position of the agent's ads within each focus-specialty group, only for the days in which the agent's ad was shown.

For each query q in a focus-specialty group g the set D_q holds the days in which the

agent's ad was shown in response to the query. We compute:

$$pos_g = \sum_{q \in g} \frac{\sum_{d \in D_q} pos_q^d}{5 \cdot |g| \cdot |D_q|}$$

This attribute tells us if the agent aims at higher or lower positions, and if it aims at different positions for different focus-specialty groups. For example, we can observe agents who try to get their ad shown first for their specialty products, but try to get lower positions for other queries.

3. **Proportion of active days:** The number of days in which the agent's ad was shown for each query, and then average within each focus-specialty group. For each focus-specialty group g (and D_q as defined above) we compute:

$$days_g = \frac{\sum_{q \in g} |D_q|}{|g| \cdot 60}$$

This attribute can be combined with the above-mentioned attributes to deduce the emphasis an agent puts on a certain focus-specialty group.

4. **The standard deviation of the agent's daily profit:** This attribute is oblivious to the relative part of the profit that comes from each focus-specialty group, but rather looks at the daily total revenues and total costs to compute a single attribute. We scale this attribute by the empirical maximal standard deviation observed across all games in order to normalize. Hence, this attribute is not directly related to the agent's profit, but rather to its stability.

Apparently, the strategic fingerprint vectors of each agent in various games lie in a typical range. To find these ranges we analyzed the logs of the TAC-AA 2011 finals and created a strategic fingerprint vector for each agent in each game.

To visually illustrate the strategic ranges of different agents, we used Principal Components Analysis (PCA) and projected the agents' vectors on the first 2 components. The result is shown in Figure 4.4, and we can see that each agent maintains a slightly different zone. However, there is no clear distinction between different capacities for each agent. We can conclude that the agents maintain similar behavior for all capacity values, and therefore the strategic fingerprint does not hold information about the agent's capacity.

To demonstrate agent identification using its strategic fingerprint, we used a simple

3-Nearest Neighbor model that classifies agents based on their strategic fingerprints. The error rate of this model was 5.9%. Most of the errors of the model are due to a specific agent, Mertacor, whose strategic range is rather wide, while the other agents are more accurately classified.

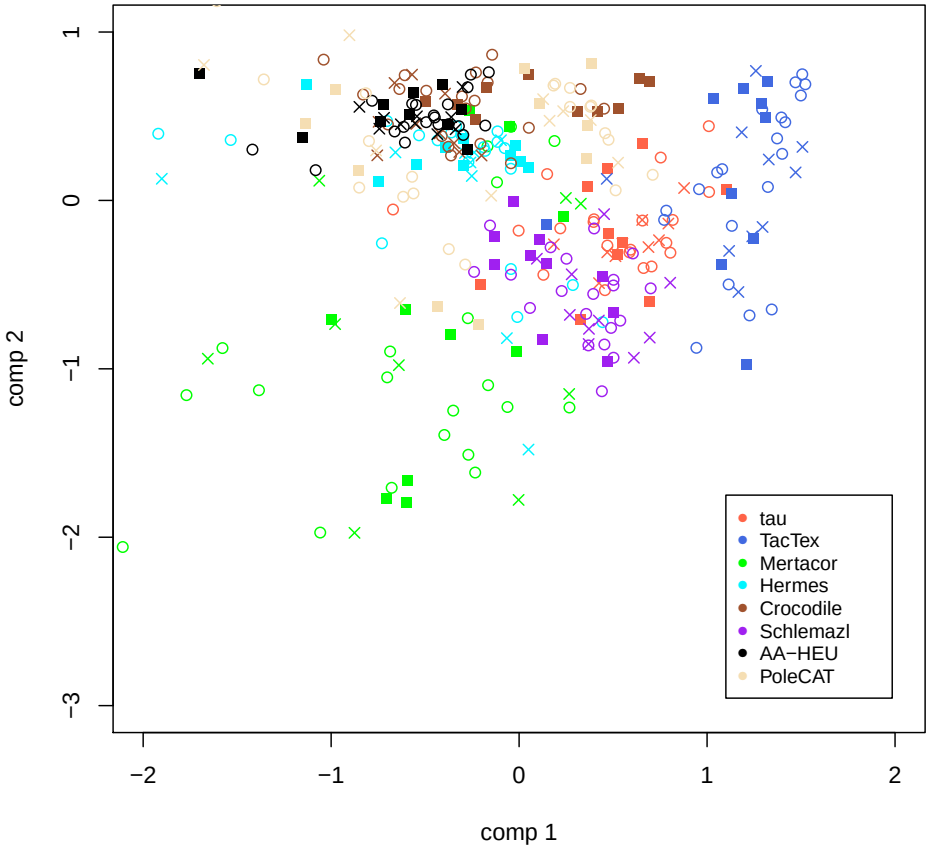


Figure 4.4: The Strategic Fingerprints of TAC-AA 11 finalists, projected on the 2 most principal components. The different point styles correspond to different capacities: high capacity is marked with full squares, medium capacity is marked with empty circles and low capacity is marked with crosses.

4.3.2 Relation to Profit

We also used the strategic fingerprints of the TAC-AA 2011 Finals in order to find the relation between strategic fingerprints and agents' performance. The measure we used to evaluate our prediction is an average relative error rate, computed as: $average(|\frac{actual\_profit - prediction}{actual\_profit}|)$.

The benchmarks to which we compare our results are two simple predictors - one that always outputs the agent's average score (has 19% relative error rate), and one that given an agent name and its capacity predicts the corresponding average (has 10.8% relative error rate). It should be noted that our model is oblivious to both the agent name and its capacity. In addition, it has no knowledge of the actual number of impressions, clicks and conversions that the agent was subjected to, nor its actual profit. It only has information about the emphasis it puts on different query groups and about its stability.

A simple 3-Nearest Neighbor model to predict an agent's profit from its strategic fingerprint had relative error rate of 14.7%, while a linear model had 12.5% relative error rate. Using Boosting with regression trees the relative error rate was reduced to 9.7%.

4.3.3 A Short Discussion

We can conclude that our strategic fingerprints model well the behavior of TAC-AA agents. Each agent has a typical range of strategic fingerprint vectors, and these vectors are well related to its performance in the game. The strategic fingerprint vectors can also be used to identify agents with high precision.

During the game, the exact values of other agents' strategic fingerprint are hidden from each agent. However, an agent can try to deduce them (e.g., by a k-Nearest Neighbors approach) in order to identify its opponents. This technique might also be applied in retrospect to identify agents from the game logs. This identification may be used in the real world, to overcome log anonymization.

Related methods of using behavioral features to differentiate TAC-AA agents were presented in [69] where the features (some very similar to the ones used here) and related distance metrics serve to cluster the agents and reason regarding the agent's profits and overall performance. Such methods, however, only differentiate groups of agents (using hierarchical clustering), and the methods presented here may therefore be viewed as complementing them in a sense.

Finally, since the strategic fingerprint reflects the agent's performance in the game, it could possibly be used for agent optimization, e.g., in a gradient descent method. However, further research is needed in order to assess the contribution of such optimization to the agent's performance.

4.4 Conclusion

We show that most of the TAC-AA agents are robust to environmental changes, despite their optimization to the exact TAC-AA setting. This robustness is "good news", and it is a very important ingredient if one wishes to relate the agents back to a real world scenario.

In addition, we present a behavioral model of TAC-AA agents that can be used to identify an agent with high probability and to predict its profit in a game. Future research could investigate the connection between this characterization and the agent's robustness. In addition, this model could possibly be used in an optimization process of a TAC-AA agent.

Chapter 5

AdX - A New TAC Game

A new game, TAC-AdX, is presented. The game simulates elements of the Ad Exchange scenario, and competing agents implement the strategies of advertisers tasked with acquiring and executing advertising campaigns. As other TAC games, this multi-agent platform serves as a controlled environment to assess agents' strategies and the mechanisms employed by the different elements in the setting. The AdX scenario is presented first, followed by a detailed description of the game and the elements of a competing agent's strategy. Thereafter, key architectural and configuration aspects of the game implementation are discussed. The chapter concludes with an account of the first few AdX competitions held during 2014 and related insights.

5.1 Motivation, The AdX Scenario

Similarly to traditional communication platforms such as radio and television, online advertising is the most significant business paradigm of the Internet. Most business models for Internet-based services depend on online advertising revenues to enable the huge investments that are needed in order to provide their services at attractive (or no) cost to users.

The Internet as an advertising platform is used by advertisers during the different stages of the purchase funnel: Display ads (the ads displayed alongside web content) are mostly used to strengthen brands by creating awareness and interest, while sponsored search ads (the ads displayed alongside search results) are mainly used to directly induce sales of products or services. This difference also results in different pricing schemes for the ads: while advertisers pay a *cost per click* (CPC) for sponsored search, the

display ads are usually priced per thousand impressions - *Cost Per Mille* (CPM). The effectiveness of both schemes however (from the advertiser's perspective) relies on the ability to target the right audience.

While the effectiveness of sponsored search advertising is straightforward to measure (direct effect on sales), the situation is more challenging for brand advertising where brand awareness and purchase intentions may only be indirectly deduced. Nevertheless, brand advertising accounts for a significant portion of the Internet advertising activity. It is therefore not surprising that with the advent of some key enabling technologies[1], the ecosystem has evolved from direct advertiser-publisher interaction for setting up-front the price of the impressions inventory, to an interconnected network of entities in which the inventory prices are dynamically set. Many of those entities, schematically partitioned to the Ad Exchange, the publisher's supply side platforms, and the advertiser's demand side platforms (all introduced below, adding value to the advertisers, publishers, or both) are essential to our AdX setting modeled. Some other entities, such as ad delivery servers and content distribution systems, take part in the display ad ecosystem but are less relevant to our AdX setting modeled, and are therefore omitted from the game.

As the number of interactions between advertisers and publishers increased, Supply Side Platforms (SSPs) were introduced to assist the publishers to optimize their inventory allocation decisions (e.g., by dynamically assigning each ad impression opportunity to one of several contracted advertisers). Ad Exchanges were introduced in turn to increase ad value to publishers, offering liquidity of inventory (e.g., impression opportunities that did not fit any ongoing pre-contracted campaign) and value discovery (i.e., impressions that may be sold for higher value than the contracted price) through a platform that enabled interested advertisers (or ad networks and agencies acting on their behalf) to bid for impression opportunities. Similarly to SSPs and Ad Exchanges, Demand Side Platforms (DSPs) were introduced to assist the ad agencies and networks in optimizing their decisions (e.g., budget allocation of the advertising campaigns across publishers and ad exchanges, and impression opportunities bid levels) such that market targeting goals are met. Finally, audience classification is key both for publishers and advertisers (the former may get higher prices for impressions in which the audience attributes are specified, the latter uses the audience attributes to ensure

[1] Mainly user classification services and real-time bidding.

proper targeting). Therefore, user classification services are also provided by dedicated entities based on cookie matching technologies. The resulting setting is schematically illustrated in Figure 5.1.

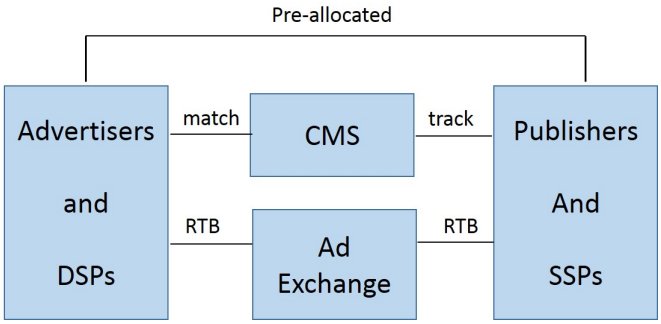


Figure 5.1: The AdX Setting: Publishers (potentially using SSPs) may allocate user impressions to advertisers according to pre-agreed contracts or through the Ad Exchange in real-time (RTB). Advertisers respond to real-time impression opportunities (potentially using DSPs) with a bid that may depend on the related user, whose attributes are provided by a Cookie Matching Service (CMS) provider that trackes users across publishers.

As noted, the Ad Exchange (AdX) is a pivotal entity in the display ad ecosystem. It interacts with most interested entities (sometimes including cookie matching services), provides added value both to the publishers and the advertising agencies, and is best positioned to extract value from the aggregated information that flows through it as bidding takes place (e.g., the true value of ad impressions to different advertisers, the orientation of the audience of different publishers, etc.).

Naturally, this has spawned research activity aimed at analyzing and establishing the methods used by the different entities involved (as surveyed in [91]): e.g, the auction mechanism at the AdX [83], the reserve price decision by the publisher (or more generally, the decision whether to submit an impression opportunity to an AdX or consume a prepaid inventory) [19], and, in a somewhat different setting, the bid price decision by the ad network [56].

The purpose of designing the AdX game is multi-fold. Being a multi-agent system and as with the other TAC games, the AdX game is a platform for evaluating the effects of the implementation choices of different mechanisms on the situation dynamics. This mainly includes (among many others, to be detailed later on) the method in which advertising campaigns are allocated and the payments (to the chosen agencies that execute the campaigns) are determined, the information available to agencies (and related cost) as they calculate their bids upon impression opportunities, and the

mechanism implemented by the Ad Exchange for calculating the winner, cost and the method to set reserve prices by the publisher upon impression opportunities.

Crucially, the AdX game is designed to bring forward the question of *the value of information*. Specifically, the amount agents implementing the ad network strategy should pay for information regarding the attributes of the user related to an impression opportunity. To that end, the AdX game includes a *User Classification Service* that competing agents periodically bid for. Determining the value of such information (in practice, say for bidding purposes) is challenging, mainly since the instantaneous value of such information depends on the profile of active campaigns to be executed (fulfillment level, remaining time and number of impressions, scarcity of targeted population, to name a few) while the information service level and cost is set for the aggregated number of impression opportunities during a whole period and for all campaigns.

Ultimately, the AdX game provides a controlled environment for evaluating Ad Network strategies through competitions. Such a simplified environment, having significantly fewer parameters (compared to *reality*) yet capturing the essence of challenges faced by Ad Networks, constitutes a test-bed for designing Ad Network strategies. Moreover, by the nature of the setting being a multiagent game, the performance of agent strategies depend on the strategies of the other competing agents and as such the AdX game is essentially a platform for evaluating performance sensitivity to changes in strategy (both of the agent being designed and its competitors) without the costs associated to such task in reality. Nevertheless, to maintain relevance and make the game as realistic as possible, many game parameters are based on *real* data, combining information from U.S Census Bureau [38], web traffic data provider Alexa, and audience measurement company Quantcast.

All in all, taking the ad network perspective, the competing agents in the AdX game implement the strategies of Ad Networks. They periodically bid to win advertising campaigns and execute the campaigns by bidding for impression opportunities at the Ad Exchange. The advertising campaigns are characterized by duration, target user population, and *reach* and each campaign is allocated to the agent offering to execute it at the lowest effective budget (the *reach* of an advertising campaign is the number of unique users of the required market segment that are exposed to the campaign, and the effective budget is a score that considers the offered budget and a quality rating that is updated based on the ability of the agent to successfully execute campaigns). This gives

rise to a fundamental conflict faced by ad networks which are required to balance the long term profitability goal (attracting advertisers by providing sustainable high quality targeting) with the short term campaign profitability goal (which depends on its ability to win properly targeted impression at low cost, compared to the agreed upon budget). Furthermore, since an ad network may conduct several campaigns simultaneously, a key challenge for the ad network in this setting is the choice of advertising campaign to serve for each impression opportunity. The AdX game is designed around those conflicts and challenges, while simulating many of the methods and mechanisms of the other entities involved, mainly the reserve price optimization by publishers (a too high reserve price might result in unsold impressions and therefore unrealized potential profits), an approximation of the the real-time bidding at the Ad Exchange, and an auction for user classification service level as a way to reveal the actual value of such information.

5.2 Game Overview

A high level description of the game model, elements, and flow is provided. Those are described in more detail in subsequent sections.

5.2.1 A Brief Description

In the AdX game each competitor implements a software agent that performs the bidding strategy of an Ad Network (AdNet), while a game server simulates the behavior of users, web sites, advertisers, and an Ad Exchange. Advertising campaigns are created by advertisers to promote their brands, and the AdNet's role is to carry out such advertising campaigns. Each campaign targets a designated subset (*Market Segment*) of the Internet user's population and has a predefined required number of impressions (*Reach*) and duration. Each campaign is auctioned among the competing AdNets, and is allocated to the AdNet that bids to execute the campaign at the lowest cost[2] to the advertiser (*Budget*).

An AdNet carries out a campaign by bidding for impression opportunities at the Ad Exchange (*AdX*). Each impression opportunity is the result of an Internet User (*User*) visiting a Web Site (*Publisher*), and is allocated by the AdX to the highest bidding

[2]With some restrictions, to be detailed later.

AdNet. Upon the termination of a campaign, the AdNet gets paid by the Advertiser an amount that depends on the Budget and the actual Reach achieved. The *Quality Rating* of the AdNet - its ability to execute a campaign as contracted - is also updated and used in the campaign allocation auction. Deducting from the amount paid by the advertiser the price paid through the AdX for the user's impressions results in the AdNet's net income related to the campaign.

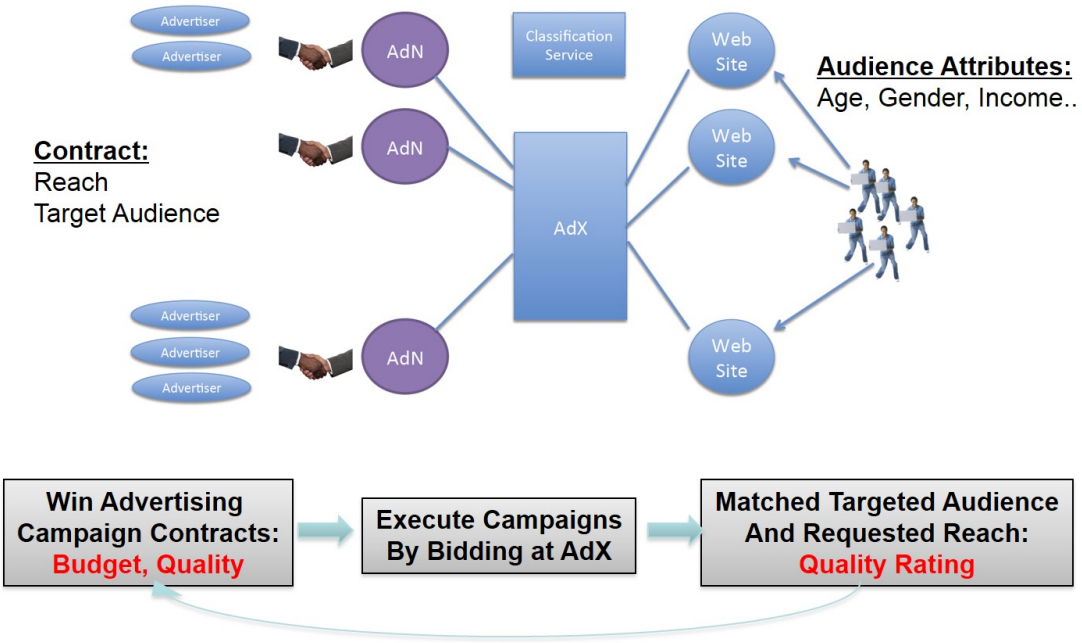


Figure 5.2: AdX game setting: From the *supply* side, visits of *users* (characterized by their age, gender, and income level) to *publisher's* web sites result in impression opportunities that are auctioned by the *Ad Exchange*. From the *demand* side, the *Ad Networks* bid daily to win advertising campaigns (characterized by their reach and targeted audience). The Ad Networks also bid daily for the cost and level of a *user classification service* that determines the ability of the Ad Network to identify the market segment of the potential user to be impressed. Bids for the impression opportunities are submitted by competing *Ad Networks* in order to execute their contracted *Advertising Campaigns*.

The game server simulates up to 60 days.[3] A new campaign is announced and auctioned every day (therefore, a competing AdNet may be executing several campaigns simultaneously!). Each day, every user visits one or more Publisher's web sites (The sites visited are randomly chosen according to the user's attributes and the Publishers' predefined orientation), and the resulting impression opportunities are handled by the AdX and assigned to AdNet's campaigns. The ability of an AdNet to access the user attributes related to an impression opportunity (such attributes highly influence the

[3]This figure, and many other game parameters are configurable through a configuration file

relevance of the impression to a campaign and as a result its value to the AdNet and the related bid) is determined by the current User Classification Service (*UCS*) level of the AdNet. The AdNets bid daily for the UCS level. Upon game termination, the total score of each competing AdNet is the sum of campaign related net income deducted by the accumulated UCS cost. The game entities and relations are illustrated in Figure 5.2 which also illustrates the key challenge faced by a competing Ad Network: In the short-term the Ad Network wishes to make a high net profit by winning advertising campaigns and executing them at an AdX cost that is significantly lower (to allow for UCS costs that are shared among all campaigns) than the related campaign budget. In the long-term, however, AdX execution costs may be high (due to competition over certain user populations with other Ad Networks executing other campaigns) in order to maintain a high quality rating that is essential to win future advertising campaigns.

5.2.2 Game Entities and Flow

As typical in the Trading Agent Competition[4] and architecturally similar to the TAC Ad Auctions[5] (TAC-AA) game [70], the game consists of a sequence of periods (each lasting one day) in which the competing Ad Networks aim to win user impressions in order to fulfill their contracted advertising campaigns. Every simulated day the agent bids to win advertising campaign contracts and submits a bidding strategy to the Ad Exchange. The Game server simulates the daily activity of a population of users who visit web sites, each visit resulting in an impression opportunity announced to the Ad Exchange. Upon every impression opportunity the game server (simulating the Ad Exchange functionality) conducts an auction based on the agent's submitted bidding strategies and the impression is allocated accordingly.

At the beginning of the game each competing Ad Network is assigned an advertising campaign, and additional advertising campaigns are auctioned daily among the Ad Networks. Each advertising campaign auctioned results in a contract in which the winning Ad Network commits to win a fixed number of targeted user impressions at a price per impression (the amount to be earned by the Ad Network) that is set through the daily auction. The actual bid for a user impression may also depend on the access device used by the user to access the web sites (desktop or mobile) and the type of ad

[4] See www.sics.se/tac.
[5] See aa.tradingagents.org/

chosen by the publisher (video or text).

A performance rating is maintained for each Ad Network. The performance rating is taken into account in the daily advertising campaign auction (it influences the ability to win new advertising campaigns and the associated revenue) and is updated upon the expiration of each campaign based on the success level of the Ad Network in fulfilling the contract. Therefore, in order to maximize its profits (the ultimate goal of the game), it is key for the Ad Network to balance the performance rating and the actual costs of bidding for impression opportunities at the Ad Exchange. The game setting is illustrated in Figure 5.3 and further detailed:

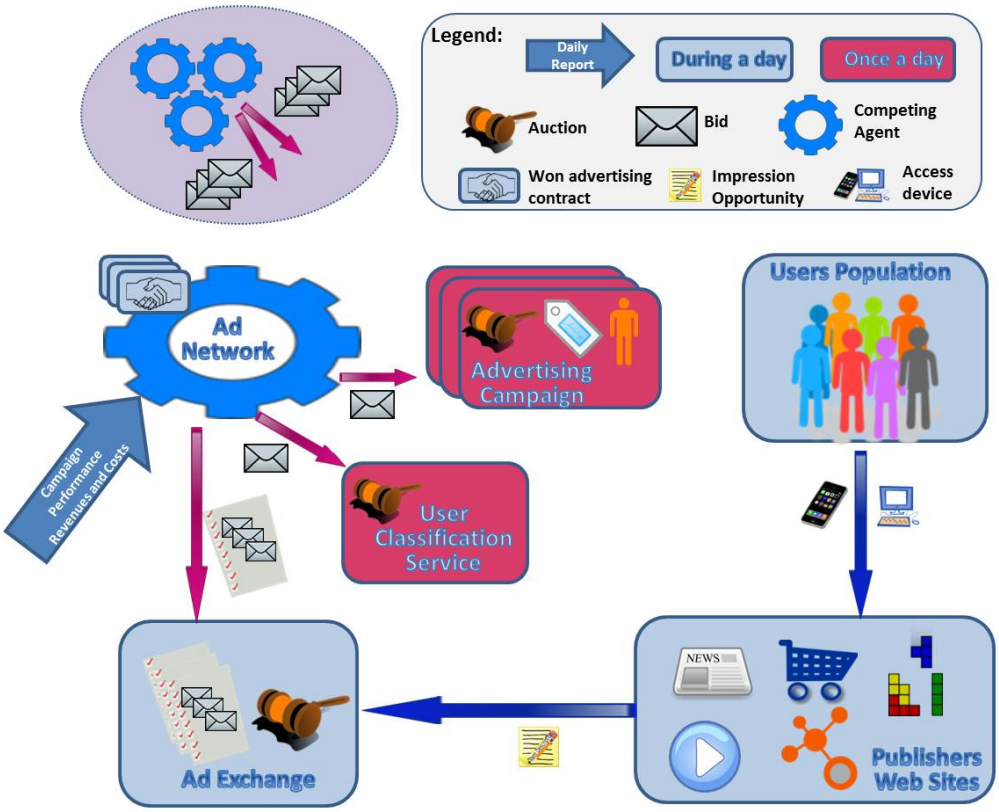


Figure 5.3: AdX game entities and flow: *Users* visits to *Publisher's* web sites result in impression opportunities that are auctioned by the *Ad Exchange*. Bids for the impression opportunities are submitted by competing *Ad Networks* in order to execute their contracted *Advertising Campaigns*. The Ad Networks also bid daily to win advertising campaigns and for the cost and level of a *user classification service* that determines the ability of the Ad Network to identify the market segment of the potential user to be impressed. The competing agents base their bids on daily reports detailing their specific contract execution figures and overall user and web sites statistics.

- *Audience: Users and Market Segments*: The user population visiting the publishers' web sites is based on Age, Gender, and Income, where each attribute has a

small set of possible values (e.g., *male* and *female* for Gender, *25-34*, *35-44*, ... for Age). Furthermore, each user's attribute value belongs to one of two ranges (e.g., *Young* and *Old* for the Age attribute, as illustrated in Figure 5.4) and a *Market Segment* is a subset of the population that belong to specific ranges for one two or all the three attributes (e.g., *Young* users of *High* income, denoted YH). Each day every user may visit one or more web sites: After each visit of a user to a web site, a continuation parameter determines whether the user continues to visit web sites or stops until the next day.

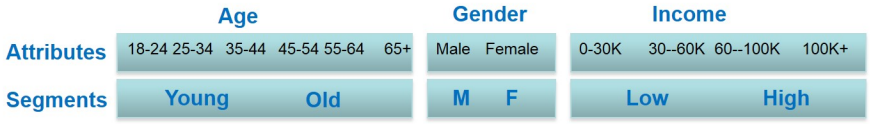


Figure 5.4: Users attributes and market segments

- *Publishers*: The web sites submitting impression opportunities to the Ad Exchange upon users' visits differ by the service they provide to the users: News, Shopping, Social interaction, Media consumption (e.g., music, video, books, etc.), and Games. Accordingly, each web site has a predefined orientation level toward the audience attributes, which is reflected in the probability of a user with certain attributes visiting each web site.

Figure 5.5: Publishers Orientation: A publisher's web site is characterized by the distribution of visiting user's attribute values (*Age* is illustrated above)

With every user visit, the *publisher* submits one or more Ad Requests (each reflecting an impression opportunity) to the AdX, accompanied with a user identification reference and a reserve price (the requested minimal price to be paid by a winning Ad Network).

- *Ad Exchange*: Upon an Ad Request from a publisher, the AdX solicits the competing Ad Networks to bid for the potential impression. Together with the Bid

Request indication, the AdX passes the related publisher and user details. The amount of user details disclosed to each Ad Network depends on the Ad Network's *User Classification Service* (UCS) level, as determined through a dedicated daily auction. The AdX implements the mechanism for selecting the winning bid and related price and facilitates the display of the ad from the allocated campaign. To allow for efficient implementation of the *Real Time Bidding* that takes place in reality (that is, the AdX announcing each impression opportunity to the Ad Networks), the competing Ad Networks communicate to the AdX a bidding map (called a *Bid Bundle*) ahead of time. The Bid Bundle maps the potential context of an impression opportunity (the user's market segment, the publisher, the access device and ad type) to a bid amount and a distribution over the Ad Network's active advertising campaigns. During each simulated day, impression opportunities are auctioned and allocated to the winning Ad-Networks' campaigns according to the submitted bidding strategies.

- *User Classification Service*: Using cookie matching technologies, the user classification service provider allows the ad networks to target the required audience for their contracted advertising campaigns. The price of the service and its accuracy are set by a dedicated daily auction.

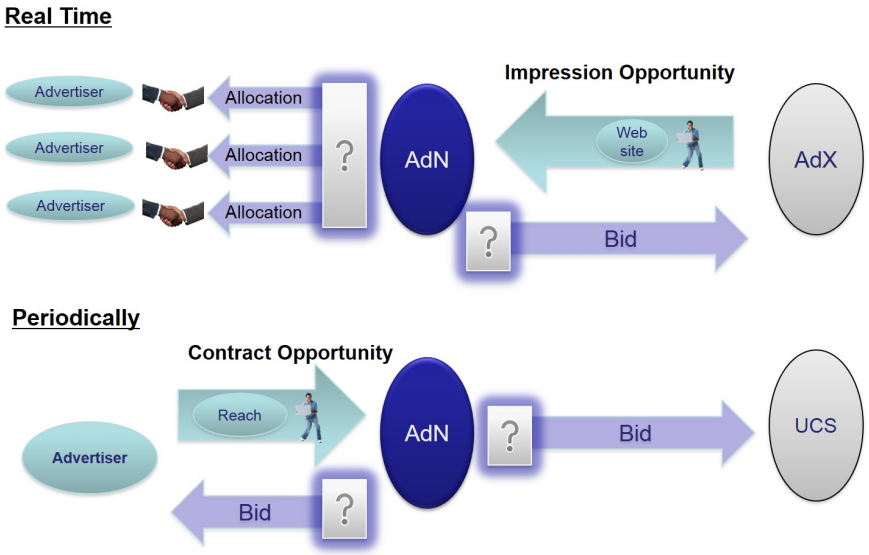


Figure 5.6: Ad Network Decisions in the AdX game: In real-time (approximated by the Bid Bundle scheme) every impression opportunity is mapped to a bid amount and a campaign allocation. Once a day, a bid for the UCS level and the outstanding auctioned advertising campaign.

Finally, *Ad Networks* are implemented (each) by a competing agent. The competing agents bid daily for new advertising campaign's budget and for user classification service level. The Ad Networks may update their daily submitted bid bundles to the AdX based on daily reports that include web sites' popularity statistics and campaigns' revenues and costs. Figure 5.6 illustrates the decisions made by an Ad Network.

This concludes the review of the AdX game entities and related mechanisms. See Figure 5.7 for an illustration of the different mechanisms as applicable to the game entities.

In later sections complete details of the mechanisms is provided, covering the many details that were omitted in this short overview: The method in which the Ad Networks' Quality Rating is updated upon campaign termination, the effect of the quality rating on the campaign allocation auction, minimum and maximum budgets in the campaign allocation auction, the way user classification service levels and related prices are set, the daily impression and budget spending limits used in the AdX impression opportunity auctions, the algorithm implemented by publishers for setting reserve prices on impression opportunities and ad types, and the way real web sites statistics are used in the simulation of the user's choice of access device and visits to publishers web sites.

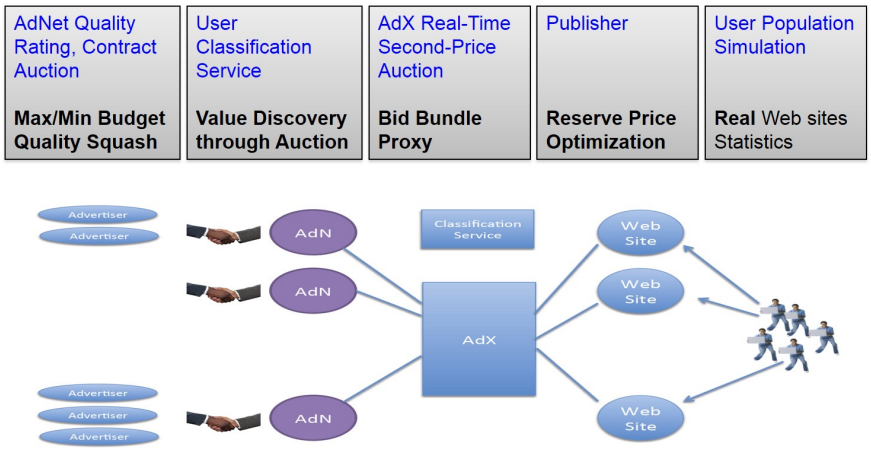


Figure 5.7: AdX Game Main Mechanisms.

Finally, the game's daily flow (illustrated in Figure 5.8) is the following: The first message received on day 0 is a campaign-allocation message - each agent gets allocated a random campaign (of random targeted audience and reach, scheduled to start on day 1) to carry out. The first message received by agents on a typical day n (where $n > 1$) is a report regarding their allocated campaigns (accumulated statistics - up to and including

day $n-1$ of achieved impressions and related costs). A campaign-opportunity message follows with details regarding the targeted audience, reach, and duration of a campaign that is scheduled to start on day $n+2$. An agent may respond with a bid-message that includes both the agent's bid regarding the budget of the campaign announced and the agent's bid with respect to the UCS. The results of the campaign and UCS auctions and the updated quality score (those to be in effect starting day $n+1$) are reported on a typical day $n > 0$ by the game server to the AdNets in a daily-notification-message that is sent before the campaign-opportunity message. Finally, after an additional set of reports sent by the server to the AdNets (a bank-status message, a publisher-report with web-site statistics, and an AdNet report with AdX bidding statistics, both regarding day $n-1$) the server simulates the users behavior of day n and during that time the agents may calculate their bid bundles to the AdX (the bid bundle includes the campaign allocation probability and bid amount to be used upon an impression opportunity, as a function of the impression attributes: the market segment the user may belong to, the access device used - mobile or desktop - and the ad type - video or text). The bid-bundle message is then sent by each AdNet to the game server upon request (a simulation-status-message).

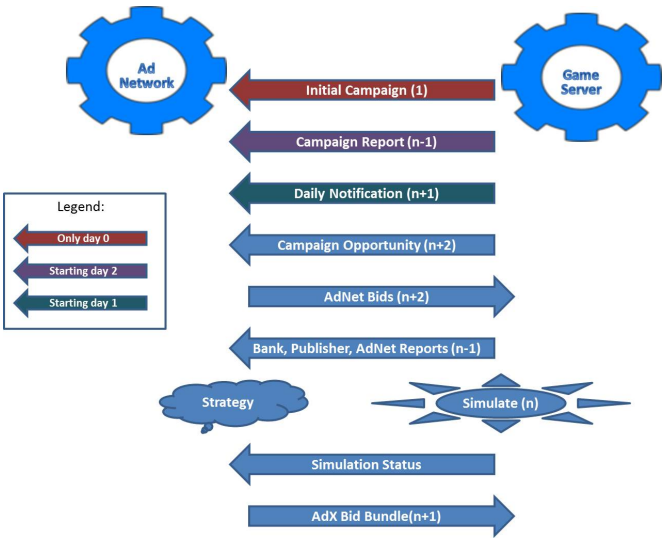


Figure 5.8: Message flow in the AdX game. The number in parenthesis indicates the day referenced in the message content. For example, the content of the Daily Notification message sent on day n announces the winning AdNetwork for the campaign to begin on day $n+1$, and the UCS level and Quality Rating to be in effect for the AdNet during day $n+1$. Note also that sending the AdNet Bids message may be deferred until after the reception of the Simulation Status message.

5.3 The Model

The game model and related mechanisms and entities are now described in detail: Those simulated by the game server (users, publishers, ad exchange, and advertising campaigns) and those to be implemented by the competing agents.

5.3.1 Users and Market Segments

In reality, the Internet economy is based on showing ads to a population of users that browse the Internet. As evident from interaction protocols such as [68], ad targeting algorithms use the users' reported attributes for their decisions (mainly since a key requirement faced by algorithms for executing advertising campaigns is to reach a designated *target market segment*). Therefore, to model such a population in a realistic manner we are required first to choose a representative subset of users features and then to generate a population sample that statistically resembles the real users population. For the first, three attributes are chosen to characterize an AdX game *user* - *Age* (One of six possible ranges: 18-24, 25-34, 35-44, 45-54,55-64,65+), *Gender* (One of the two values: Male, Female), and *Income* (One of four ranges: \$0-\$30K, \$30-\$60K, \$60-\$100K, \$100K+), and for the second U.S. Census Bureau data ([38]) is used. A population of 10000 users (the total audience) is created at the beginning of each game by sampling according to publicly known probabilities that are detailed in a table in the game specification. The table consists of an entry for each of the 48 user types, indicating the number of users - out of the default total 10000 - of that type.

Market segmentation (the structure used to designate a subset of the population) is essential for efficiently specifying the targeting of advertising campaigns (see for example [57], which uses user demographics, but also interests, activities, and the social graph). For the AdX game, a set of partitions of the users population is defined by partitioning each attribute range to two (that is, the user's Age range to $Younger = \{18-24, 25-34, 35-44\}$ and $Older = \{45-54, 55-64, 65+\}$ and the user's income range to $Low = \{\$0-\$30K, \$30-\$60K\}$ and $High = \{\$60-\$100K, \$100K+\}$). The partitions are illustrated in Figure 5.4. Now, a market segment is any intersection of partitions (at most one partition per attribute). For example, if we designate each partition by its initial letter (e.g., Female by F and Younger by Y) we get the following 12 market segments of double partitions: FY, FO, MY, MO, YL, YH, OL, OH, FL,

FH, ML, MH. Market segments of single partitions (e.g., M) or triple partitions (e.g., MYH) are also valid (and there are 6 and 8 such segments, respectively). Note that in general (and specifically in this case) the segments may overlap (i.e., a user may belong to multiple segments).

5.3.2 Publishers' Web Sites

Publishers' web sites differ by the demographic statistics of their visiting users and the governing statistics regarding the access devices used by those users and the type of ads displayed. This section details the model used to implement the different publishers, the mechanism used to set the reserve prices, and the way those are combined to simulate the users' visits and resulting impression opportunities announced to the ad exchange.

5.3.2.1 Web Site Characteristics

To model user's visits to web sites we use real data from Alexa [3] and Quantcast [6]. Using Alexa, we set age, gender, and income statistics for each of six leading web sites in *News*, *Shopping*, and *Information* business categories.[6] This demographic distribution of user attributes at a web site is interpreted as the conditional probability of a user to have a certain attribute value given a visit to the web site. Specifically, we denote the *user orientation* characteristics of web site w as $P_{\text{Age}}(\cdot|w)$, $P_{\text{Gender}}(\cdot|w)$, and $P_{\text{Income}}(\cdot|w)$. Using QuantCast data (merely the user visiting rate to web sites) we set the relative popularity $P_W(w)$ of each web site w.

Two more modeling aspects of the publisher's web sites are the user's choice of access device (whether using a desktop computer or a mobile device to browse the internet), and the publisher's choice of the ad type (that is, a static/text banner or a video clip). The value to an advertiser of impression a user through video (vs. e.g., text), may be significantly higher (and is modeled into the advertising campaign characteristics, as detailed in Section 5.3.4) although some restrictions (e.g., availability, bandwidth, user attention, etc.) might preclude the constant usage of one method over the other. Similarly, advertising campaigns may indicate higher value to impressions on a user browsing through a mobile device (vs. the desktop). Therefore, each web site w is further characterized by the *Access Device probability* $P_{\text{Device}}(\cdot|w)$ (over the

[6] We use different business categories assuming that the underlying demographic statistics are essentially different in different categories.

set {*Desktop*, *Mobile*} of access device types used by visiting users) and by the *Ad Type probability* $P_{\text{Adtype}}(\cdot|w)$ (over the set {*Video* or *Text*} of ad types that may be presented to visiting users). All the web sites characteristics are disclosed to the competing agents upfront (through tables in the game specification and dedicated messages at game start) except the Ad-Type probability - The only one under full control of the web site.[7]

5.3.2.2 Reserve Prices

A reserve price in an auction is an amount set by the auctioneer such that bids below it are ignored. In a second price auction, the reserve price also serves as the price to be paid by the winner in case the winner's bid is the only one above it. The use of reserve prices to optimize the auctioneer's revenue is justified empirically [95] and theoretically [92]. Naturally, the impact of the reserve price on the auctioneer's revenue (and the growing size of the related industries) spawned extensive research in the area of reserve price optimization, resulting in several published (e.g., [39, 89]) and surely many privately kept methods. For the AdX game, a simple adaptive gradient ascent algorithms is implemented (as detailed below), and the effect of more sophisticated algorithms on agents strategy and behavior is left for future versions of the game.

A reserve price is set by each publisher for each impression opportunity using an adaptive method that maintains during day t an *average reserve price* $b_t(u, a)$ for each user type u and impression type a (where a is one of the four combinations of Mobile / Desktop and Video / Text). The actual reserve price $r_t^i(u, a)$ of an impression i during simulation day t of user of type u and ad type a is randomly set

$$r_t^i(u, a) = b_t(u, a) + \epsilon_i \ , \tag{5.1}$$

where the perturbation ϵ_i is normally distributed with zero mean and predetermined (game parameter) variance.

Initially, for the first simulation day, $b_1(u, a)$ is chosen uniformly between 0 and a predefined game parameter. In subsequent days, the baseline average reserve price is adaptively set in the direction of the reserve price that maximized the average observed

[7] And therefore may be adaptively set by the web site during a game! In the current implementation of the game, however, the publisher simply uses a random selection.

profits $b_t^{\max}(u,a)$:[8]

$$b_{t+1}(u,a) = \eta b_t(u,a) + (1-\eta) b_t^{\max}(u,a) \ , \tag{5.2}$$

where the the learning rate η is a preset constant game parameter.

5.3.2.3 Simulation of User's Visits

At the beginning of each game, out of the total eighteen available web sites, six are randomly chosen (two from each category) to be used in the game simulation. Now, the assuming (by design) independent attribute values given a visit to a web site and applying the Bayes rule, we can formulate the probability of a user of attributes $Age = a$, $Gender = g$, and $Income = i$, to visit web site w as follows:

$$Pr([a,g,i] \text{ visits } w) \triangleq Pr(w|[a,g,i]) = \frac{Pr([a,g,i]|w)P_W(w)}{Pr([a,g,i]}) \tag{5.3}$$

$$\propto P_{\text{Age}}(\text{a}|w) P_{\text{Gender}}(\text{g}|w) P_{\text{Income}}(\text{i}|w) P_W(w) \ , \tag{5.4}$$

inducing for any user of type $[a,g,i]$ a distribution over web sites,. This induced distribution is used to simulate a web site visit (or more) for every user every simulated day. Now, every simulated day, upon visiting a web site, a user may continue visiting web sites that day with a predefined probability (a game parameter) up to a maximum of visits per day (yet another game parameter). Finally, Upon every visit of a user to a web site, one or more impression opportunities are announced to the Ad Exchange.[9]

5.3.3 Ad Exchange and User Classification

The Ad Exchange is a pivotal entity handling transactions at a very high rate. As such, the mechanisms implemented by the Ad Exchange are required to be fast (that is, efficiently implementable, usually implying simplicity). Furthermore, since the AdX mechanism determine the revenue of all other involved entities (and also it's own revenue, usually through commissions, which is not modeled in the AdX game) attention is also paid to the mechanism's effect on strategic behavior of the involved elements.

[8] Taking into account the estimated probability of at least one bid reaching or passing the reserve price. In case of no won impressions $b_t^{\max}(u,a)$ is set to 0.

[9] The number of impression opportunities is uniformly chosen from a range that is set upfront as a game parameter. Since Ad Networks are rewarded by *unique* impressions, the multiple appearance of the same ad in a web page is avoided by excluding the winning agent from subsequent auctions related to the same web page visit.

Design considerations for the AdX auction are researched in [56, 83, 91, 113], some also considering the more general case in which the bidding Ad Networks act as intermediaries (e.g., Ad Words). As common in the industry, we implement a second price auction for AdX announced impressions: for each announced impression opportunity, the Ad Exchange operates a second-price auction among the Ad Networks, where each Ad Network's reported bid (and associated campaign, in case the bid wins) depends on the context accompanying the bid request: the publisher, the ad type (either *Video* or *Text*}) and access device used (either *Mobile* or *Desktop*}), and the user (specifically, its market segment). Note that matching depends on the User Classification Service level in effect for the Ad Network. Actually, to approximate real-time bidding while keeping the messaging load of the game reasonable, the Ad Exchange consults the daily bidding strategy submitted by each competing agents instead of interacting with each agent upon every impression opportunity.

The mechanics of the second price auction are such that if the highest (that is, winning) bid is below the reserve price indicated by the publisher for the impression then the impression opportunity is lost. Otherwise, the impression is allocated to the highest bidder, at a cost being the second highest bidder or the reserve price, whichever is higher. Finally, the campaign to which a won impression is allocated (i.e., the specific ad to be displayed) is the one associated to the winning Ad Network's bid.

The Ad Network's knowledge of the context associated to an impression opportunity (especially the user's attributes) is of key importance regarding its ability to fulfill the targeting requirement of its advertising campaigns. Furthermore, publishers may prefer that bidders are aware of the user's attributes as such bidders will be willing to bid more (resulting in higher revenue to the publisher) for potential impressions that they are sure to match their targeted audience. This potential higher revenue however might be offset by a thin market phenomena (resulting from full information regarding the context of an impression) - where very few bidders compete over thin market segments (and ending up paying the reserve price, which might be significantly lower than the value to advertisers[10]). The tendency of publishers to facilitate the transfer of user information to the advertisers (e.g., through cookie matching technologies) is further hindered by the potential *information leakage* and further exploitation of the information by the advertisers to lower their costs at the expense of the publishers (e.g., the ability of the

[10]The value to advertisers is supposedly thier bid, with the second price auction being *Incentive Compatible*. This however does not necessarily hold for repeated auctions such as in the AdX scenario

advertiser to target the same user through a cheaper publisher). Such situations and concerns are analyzed in [59].

To address the question of the value to the advertiser of knowing the user information associated to a potential impression, we implement a periodic competition among Ad Networks for the services of a *User Classification Service* (UCS) entity. This reflects the situation in which the user reference (i.e., cookie) provided by the Ad Exchange has to be matched to a real user, using a dedicated third-party service provider. The competition is in the form of a *Generalized Second Price* (GSP) auction (as detailed below) and results in each Ad Network assigned a service level that is the probability of the user's attributes being available at the time of making the decision regarding the bid and campaign allocation for an impression opportunity. The auction for UCS is done daily (specifically, every day n the competing Ad Networks bid for the service level to be in effect during day $n+2$) since the value of such information to Ad Networks may depend on its related state (that is, its active campaigns, quality rating, etc.).

Specifically, in the GSP auction for UCS, the highest bidder will get revelation of user's attributes 100% of the time, the second a lower probability (a game parameter $P_{\text{UserRevelation}}$), the third $P^2_{\text{UserRevelation}}$, and so on. In the case where user attributes are not to be revealed, an "unknown attributes" value is used by the Ad Network as user's context to decide the bid and the campaign allocation. The outcome of the auction (quality and cost of the service to the ad networks) is determined in the following manner: Denote by $c_{i_1}, .., c_{i_m}$ the ordering of the bids of the ad networks from high to low. As a result of the auction, the ad network n in the k^{th} position will receive the true value of a user's attributes with probability $p_k = P^{k-1}_{\text{UserRevelation}}$, and will pay for the service the amount of,

$$K_n = p_k \cdot c_{i_{k+1}} \ . \tag{5.5}$$

The normalization by p_k above ensures that the actual price paid is the average price for correct user classification.

5.3.4 Advertising Campaigns

To be successful, Ad Networks first have to win advertising campaigns. Thereafter, Ad Networks have to execute their acquired campaigns according to the related characteristics (e.g., target audience and reach), in a manner that balances their long term

ability to win future campaigns (that is, maintain a high quality score by fully executing) and their short term earning (as determined by the cost of the ad impressions at the Ad Exchange). This section details the way campaigns are generated for competing Ad Networks to bid upon, how allocated to Ad Networks, and the effect of campaign execution results on the quality rating and earnings of the Ad Networks.

5.3.4.1 Campaign Generation

Naturally, due to their inherent financial impact, the characteristics of effective advertising campaigns were extensively researched (see e.g., [120]). There is no doubt however, that the defining the campaign's targeted audience over time (that is, the campaign length, size, and attributes of the targeted population) is of key importance. When it comes to digital media (that is, advertising through display ads in web sites and mobile apps) the impact of an impression may also be effected by the medium - the access device used and the ad type. We therefore characterize a marketing campaign in the AdeX game by the required *reach* C_R (the size of total audience impressed - the total effective number of daily-unique user impressions), the duration in days C_L, the targeted market segment C_S, an ad type preference factor C_V (A unique *Video* impression is counted as C_V effective impressions. A *Text* impression is counted as one effective impression), and access device preference factor C_M (A unique *Mobile* impression is counted as C_M effective impressions. A *Desktop* impression is counted as one effective impression[11]).

A marketing campaign's contract C is auctioned daily among the competing Ad Networks. Special care is taken to ensure the game balance between demand (the number and type of impressions required to fulfill the allocated campaigns) and supply (the number of potential impressions to browsing users). Therefore, the campaign parameters are set as follows: C_L is uniformly chosen to be one of $C_{\text{CampaignL1}}$, $C_{\text{CampaignL2}}$, $C_{\text{CampaignL3}}$. C_S is uniformly chosen over the 26 possible market segment combinations (see Section 5.3.1). Finally, a reach level C_{RL} is uniformly chosen over $C_{\text{CampaignLowReach}}$, $C_{\text{CampaignMediumReach}}$, $C_{\text{CampaignHighReach}}$ and C_R is set

$$C_R = C_{RL} \cdot |C_S| \cdot C_L \ ,$$

[11] A user that is impressed more than once during a period is counted according to the highest effective value of the impressions.

where $|C_S|$ is the average size of the chosen target segment C_S.

5.3.4.2 Campaign Allocation

Every simulated day, each Ad Network n optionally bids B_n for the total budget of the daily auctioned campaign. As detailed below, before applying a second price auction among the bidding Ad Networks each bid B_n is squashed using the outstanding quality rating of the Ad Network (resulting in an effective bid for each Ad Network) and only effective bids that are not too low (to preclude networks from submitting unrealistic offers) and not too high (essentially acting as a reserve price - the maximum an advertiser is willing to pay an Ad Network for executing a campaign) are qualified.[12] This squashing scheme might make it very hard to Ad Networks that were 'unlucky' in executing past campaigns (and as a result have a low quality rating) to win newly auctioned campaigns. Therefore, to allow a 'second chance' to such Ad Networks the allocation of a campaign may be randomly allocated (instead of using a second price auction - with some probability that is a game parameter that is set upfront and known to all) among the qualified Ad Networks. If not randomly allocated, the campaign is allocated to the Ad Network of highest quality per dollar ratio. That is, the campaign is allocated to the Ad Network with highest effective bid $e_n = \frac{Q_n}{B_n}$, where Q_n is the quality rating of Ad Network n as defined later on by (5.9).

As noted above, reserve maximal and minimal cost per impression (game parameters $R_{\text{CampaignMax}}$ and $R_{\text{CampaignMin}}$ respectively) are used in all campaign auctions, the former to bound the price to be paid by advertisers for campaigns of little demand and the latter to preclude Ad Networks from committing to execute campaigns at unreasonable loss while denying other Ad Networks a chance to execute the campaign. The quality rating is taken into account in the reserve prices by only considering bids that satisfy $B_n Q_n > C_R R_{\text{CampaignMin}}$ and $\frac{B_n}{Q_n} < C_R R_{\text{CampaignMax}}$ (thereby giving the advantage to Ad Networks of higher quality rating in both directions). The campaign's budget C_B is set to the maximum campaign budget the winning Ad Network could have had bid and still win: $C_B = \frac{Q_{\text{win}}}{b_{\text{second}}}$ where Q_{win} is the quality rating of the winning Ad Network and $b_{\text{second}} = \max\{e_{\text{second}}, \frac{1}{R_{\text{Campaign}} C_R}\}$ is the highest of the effective bid of the Ad Network reaching the auction's second place and the reserve effective value.

[12] Note that squashing is also performed in sponsored search ads auctions, for different reasons: merely to normalize the bids from 'pay per click' to 'pay per impression', which better reflects the auctioneer's revenue.

5.3.4.3 Campaign Execution

For an impression on user u, let $C(u)$ be the contract chosen by the ad network for the potential impression on u and let $D_C(u)$ be $C(u)_M$ if u is using a mobile device and 1 otherwise. Similarly, let $T_C(u)$ be $C(u)_V$ if u is being impressed by video and 1 otherwise.

The effective number of unique impressions w.r.t. contract C achieved by Ad Network n is

$$I_n(C) = \sum_{u:C(u)=C \text{ and } C_S \in s(u)} D_C(u) \cdot T_C(u) \ , \tag{5.6}$$

where $s(u)$ indicates the actual[13] set of segments to which user u belongs, and the sum is over all daily-unique impressions on users u that belong to segment s.

Now, to discourage Ad Networks from deviating from the required reach levels of the contract, a nonlinear relation is defined between the effective number of unique impressions $I_n(C)$ achieved and the related Ad Network's payment as follows: First, the effective reach ratio $\mathrm{ERR}_n(C)$ of contract C is set

$$\mathrm{ERR}_n(C) = \frac{2}{a}\big[\arctan(a\frac{I_n(C)}{C_R} - b) - \arctan(-b)\big] \ , \tag{5.7}$$

where a and b are set[14] such that when $I_n(C) = C_R$ we have $\mathrm{ERR}_n(C) = 1$ and the marginal effective reach per unique impression is $\frac{1}{C_R}$ (that is, when the effective number of impressions achieved is exactly the contracts' required reach we have an effective reach ration equaling 1 and the benefit of each additional impression is the campaign's average reach amount per impression). This monotone relation is illustrated in Figure 5.9.

Now, the payment $E_n(C)$ to Ad Network n for impressions on users allocated to contract C is set

$$E_n(C) = \mathrm{ERR}_n(C) \cdot C_B \ . \tag{5.8}$$

Note that by the form of (5.7), achieving a significantly lower number of impressions than required results in loss of un-proportional portion of the revenue to the Ad Net-

[13] Note that the ad network may not have complete information to exactly compute $I_n(C)$. The game server computes this value, mimicking a marketing survey that may take place in reality upon the conclusion of a campaign. After all, the actual segment to which the user belongs carries the true marketing value for the advertiser!

[14] For any nonzero k, take the unique b satisfying $\frac{\arctan(k)-\arctan(-b)}{1+b} = \frac{1}{1+k^2}$, and set $a = b + k$. We use $k = 1$ resulting in $a = 4.08577$ and $b = 3.08577$.

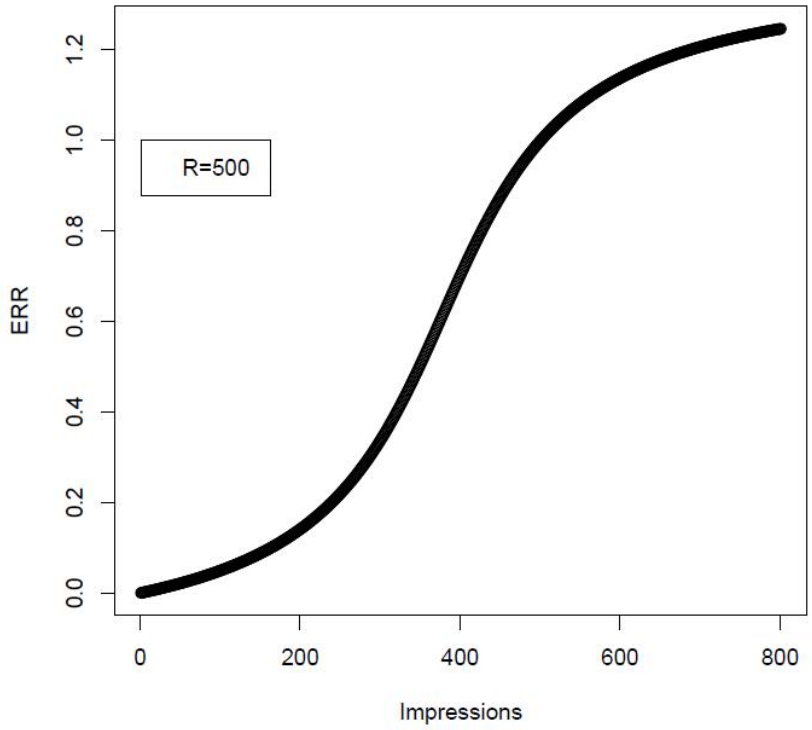


Figure 5.9: The Effective Reach Ratio (ERR) as a function of the effective number of unique impressions achieved by the ad network, for a contract requiring a reach $C_R = 500$.

work (reflecting unrelated fixed costs to the advertiser that are now wasted due to the inability of the Ad Network to execute). Similarly, achieving a significantly higher reach than requested results in un-proportionally lower portion of additional revenue to the Ad Network (since the advertiser might not have prepared to make use and benefit from the unplanned increased number of impressions, e.g., due to lack of inventory).

At the expiration of every contract, the quality rating Q_n of the relevant Ad Network is updated using $\eta = L_{\text{Rating}}$ learning rate:

$$Q_n^{\text{new}} = (1 - \eta)Q_n + \eta \text{ERR}_n(C) \ . \tag{5.9}$$

5.3.5 Ad Networks

The last modeled element in the AdX game is the Ad Network, whose strategy is implemented by the competing agents. As will become evident from the description below, the resulting strategy space is huge, making the problem of designing and analyzing such strategies both challenging and interesting.

5.3.5.1 Decisions

The Ad Networks, bid daily for advertising campaign contracts and for user classification service level. To overcome the intensive communication rate required to implement real-time bidding in our game, every Ad Network n submits to the AdX upfront a daily bid bundle. The bid bundle is used by the AdX throughout the day to map bid requests to ad networks' bid/contract pairs by indicating the following information for each market segment s,[15] site w, access device, and ad type combination:

- Contract weight p_n: The weight associated to the contract induces a probability distribution over all the contracts associated with entries that match a certain impression opportunity. An impression opportunity matches an entry if the user to be impressed belongs to the market segment of the entry. The "unknown" segment entry matches impressions for which the user classification service fails to recover the user attributes.

- Bid b_n: The bid level, upon assignment of the impression opportunity to contract C.

The way the game server simulates the daily real-time bidding on behalf of the Ad Networks is as follows: Upon a bid request as a result of user u visiting web site w, the matching Bid Bundle entries are set according to actual user attributes and the user classification service level in effect (multiple segments may apply - resulting in more than one matching bid-bundle entry). The contract C_n to use for Ad Network n is randomly selected according to probability induced by p_n and the bid amount is set according to the chosen entry's b_n. The daily bid bundle of an Ad Network may also indicate for each assigned campaign a budget constraint on the daily spending for impressions and an impressions-won constraint (and also similar total campaign constraints). Once the spending or impressions limit is reached for contract C no more bids are placed on behalf Ad Network n w.r.t. contract C.

Now, set $c_n(u)$ to be the price paid by Ad Network n for an impression won on user u (the outcome of the second-price auction conducted by the AdX). The net earnings

[15] an "unknown" segment is also included - to be used when the user classification service fails to reveal the visiting user's segment.

$N_n(C)$ of ad network n on contract C are therefore:

$$N_n(C) = E_n(C) - \sum_{u:C(u)=C} c_n(u) \ , \tag{5.10}$$

where $E_n(C)$, Ad Network's n income related to contract C, is according to (5.8).

Upon game termination each ad network score is its net accumulated profits (5.10) over all executed contracts, less the cost of the user classification service over all periods. The overall score of Ad Network n is therefore

$$N_n = \sum_C N_n(C) - \sum_d K_n(d) \ , \tag{5.11}$$

where $K_n(d)$ is the price paid by Ad Network n for the user classification service on day d as set by (5.5).

5.4 Game Flow

The game consists of at most T_{Gamedays}[16] simulated days in which up to 8 competing ad networks aim to maximize their total accumulated profits (5.11). Throughout the game each ad network executes one or more campaigns (the first one is allocated, the others have to be won), where its competitiveness in winning campaign contracts (reflected by a squashing value applied to its bids) depends on the targeting effectiveness achieved by executing its past campaigns (5.9).

5.4.1 Daily Sequence of Actions and Events

To achieve its goals, each Ad Network bids daily for users' impression opportunities and selects for each potentially won impression which of its contracts to serve. The ad networks base their decisions on daily reports. In what follows, the daiy sequence of actions and events is detailed for a typical day d (from the point of view of an agent implementing the Ad Network functionality). Note that a day in game-time is executed in $T_{\text{Dayseconds}}$ real-time seconds. The messaging sequence of Figure 5.8 is illustrated as part of the conceptual sequence of actions of Figure 5.10: Day d starts with the server sending to the Ad Network the reports (5.4.2.2) regarding results of simulating day $d-1$.

[16] T_{Gamedays} and other similarly named constants are game parameters that are fixed in advance and known to the competitors.

Notifications (5.4.2.3) about the campaign contract and user classification service level to be in effect during day $d + 1$ are sent next, followed by details (5.4.2.4) regarding the campaign to start on day $d + 2$. Based on the information, the Ad Network may conduct an analysis toward decisions regarding its bidding strategy during day $d + 1$ while the server simulates the impressions and related AdX auctions of day d. After the daily simulation ends, the server polls each of the competing agents for the daily decisions (5.4.2.5) and the cycle repeats.

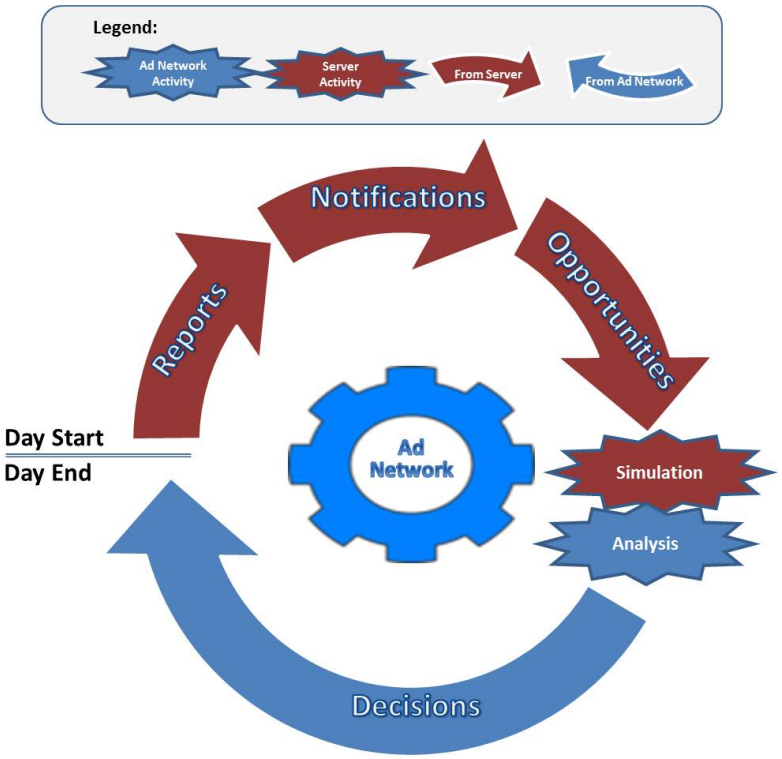


Figure 5.10: AdX game daily flow

5.4.2 Mesages

The game server and the competing agents implementing the Ad Network strategies run in potentially remote locations and communicate through dedicated game-specific messages. Such messages comprise of reports by the game server regarding the results of past days simulations, announces by the game server regarding campaign opportunities and notifications regarding the results of related auctions, and decisions by competing Ad Networks regarding their bids for announced campaigns and impression opportunities at the Ad Exchange. This section details the types and sequencing of

messages during the different phases of the game.

5.4.2.1 The First Days

During day 0 the agents are notified regarding their first campaign contract (each Ad Network is allocated one campaign) to start on day 1. All initial campaigns have $C_{\text{CampaignL2}}$ duration, $C_{\text{CampaignMediumReach}}$ reach level, and double partition target segment C_S. The Budget of each initial campaigns is set to 1\$ CPM. During day 1 the user classification service is provided to all agents at no cost and at accuracy level ($Z_{\text{UCSaccuracy}}$). As in subsequent days, during day 0 the campaign opportunity for day 2 is announced, and the competing agents respond with their related decisions (Bid Bundles for the simulation of day 1, bids for the campaign opportunity that was just announced, and UCS bids for the user classification service level to be in effect during day 2).

5.4.2.2 Reports

After the game server simulates the users visits to publisher's web sites during day $d-1$ and related AdX auctions, each agent n receives during day d reports summarizing the status as of the preceding day $d-1$.

- Publishers-Report: A public report that details user visiting statistics for each web site (popularity and orientation frequencies, out of total user visits during the day), and impressions statistics by campaign.
- AdNet-Report: A private report that details the Ad Network bidding totals for each bid-bundle entry[17]: bids, wins, and cost.
- Campaigns-Report: A private report that details accumulated statistics for each AdNet's campaign: targeted impressions, non-targeted impressions, and cost.
- Bank Status: A private report with the balance of the recipient.

5.4.2.3 Notifications

A Daily-Notification message is sent on day d to notify the Ad Networks regarding the results of the following developments of fay $d-1$:

[17] For convenience, the key includes the campaign id.

- Campaign-Auction-Result: The Ad Networks are announced regarding the winner of the campaign auctioned on day $d-1$. The campaign is scheduled to start on day $d+1$ and last until day $d+C_L$. The winner is also announced regarding the resulting budget C_B (this figure is not disclosed to non-winners).

- UCS-Auction-Result: The Ad Networks are notified (each) regarding their user classification service level and cost to be in effect during day $d+1$ as auctioned according to the bids submitted during day $d-1$.

- Quality-Rating-Update: Each Ad Network is notified regarding its updated Quality-Rating, as a result of campaigns ending during day $d-1$. This rating is in effect for the Campaign-Opportunity auctions to take place during day d.

5.4.2.4 Campaign Opportunity

The details of the advertising campaign to start on day $d+2$ are provided to the Ad Networks. The Ad Networks' bids (to be sent to the game server during day d as detailed below in Section 5.4.2.5) will be considered and the winner will be announced (as detailed above in Section 5.4.2.3) during day $d+1$.

As the last simulation day approaches, campaigns whose end day are beyond the last day are not announced. Whenever this happens an empty campaign is indicated in the Campaign-Opportunity message.

5.4.2.5 Ad Networks Decisions

After the Ad Networks consider the reports, notifications, and opportunities, each submits the related decisions to the game server: The bid for the advertising campaign opportunity just announced (see Section 5.4.2.4) together with the the bid user classification service level to be in effect during day $d+2$. The results of the user classification service level auction will be reported during day $d+1$ (as in Section 5.4.2.3). Finally, each Ad Network submits a Bid-Bundle message reflecting its bidding strategy to be used by the Ad Exchange upon impression opportunities resulting from users visits to web sites during day $d+1$. The strategy is conveyed in a bid bundle that maps the context of the impression opportunity to a bid level and a distribution over campaign contracts, as detailed in section 5.3.5.1.

5.5 Elements of Ad Network Strategy

The Ad Network daily decisions (as illustrated in Figure 5.6) reflect the consideration of many inter-related aspects, which (for convenience) are grouped by the decision most effected in the following sections:

5.5.1 Bidding for the User Classification Service level

The value of the user classification service to the Ad Network is a fundamental aspect of the game. The benefit of having a certain UCS level depends of course on the (number, attributes, and status of) active campaigns the competing Ad Network has. Foe example, no UCS is required when there are no active campaigns, and minimal UCS may be needed for a campaign targeting a large fraction of the whole population (in that case an 'unknown' match of a user is actually targeted with high probability). For campaigns targeting a scarce audience, on the other hand, and especially if the campaign is about to end while the contracted reach is yet to be achieved, the value of accurate user classification might be well above the direct benefit to the revenue from the campaign at hand since poor performance in executing the campaign results in reduced quality rating that can have detrimental effect on future earnings of the Ad Network (hence, this may be more important during earlier stages of the game, especially during the first few days while Ad Networks execute the pre-allocated campaigns). The choice of the Ad Network to try and achieve a certain UCS level is based also on the associated cost, which should be estimated by the Ad Network taking into account the state of its competitions and an assessment of their bidding strategies (as reflected by their past actions in the data available to the competing Ad Network through the daily reports).

5.5.2 Bidding for the Campaign Opportunity

Naturally, the decision whether to try and win a campaign opportunity depends on the prospect benefits from the campaign, which in turn is effected by the state of the other active campaigns (specifically their targeted populations and reach levels) which dictates the future competition (and therefore related cost) over user impressions. In this regard, longer campaigns targeting larger populations may be more attractive, and short campaigns might be harder to fulfill (and therefore probably less attractive due to potential effect on the quality rating). Furthermore, the overall benefit of winning

a campaign should consider the associated cost of the UCS - which is shared by all active campaigns and might be non-trivial to allocate. Finally, the actual bid for the campaign budget should take into account the Quality Rating levels and (as before) the bidding strategies of the competitors.

5.5.3 Ad Exchange Bid Bundle

The Bid Bundle associates the context of a user's impression opportunity to an Ad Network's bid and campaign assignment probability (over the active campaigns of the bidding Ad Network). The bid level (again) may depend on the state (allocated campaigns) and bidding strategies of the competitors, but also on the reserve price set by the publisher. Specifically, the Ad Network should probably consider the specified and reported publisher's orientation and users statistics to assess the competition levels over certain impressions and set the bids accordingly. The Ad Network may also set daily count and spending limits - those depend on campaign's reach levels but may also consider the longer-term effect of over or under fulfillment of the campaign.

5.5.3.1 The AdNet Problem

The AdX bidding strategy may be formulated in a simplified and abstract way as follows: An Ad-Network commits to a set of m contract with advertisers. A contract $i \in \{1, \dots m\}$ is characterized by a concave utility funtion that associates a financial benefit $u_i(x)$ to the event of allocating x matching impressions to the advertising campaign associated to contract i.

The n impression opportunities are announced to the Ad Network by the Ad-Exchange sequentially. Each impression opportunity $j \in \{1 \dots, n\}$ is characterized by a binary vector $\mathbf{a}_j = (a_{1j}, a_{2j} \dots a_{mj})^t \in \{0,1\}^m$ where a_{ij} is interpreted as the relevance of impression j to contract i. We further assume that $\mathbf{a_j}$ are generated i.i.d with respect to the time index j.

The advertiser decision for each impression opportunity $\mathbf{a_j}$ is an allocation $\mathbf{x_j} = (x_{1j}, x_{2j} \dots x_{mj})^t$, where x_{ij} is interpreted as the (eventual) probability of assigning impression j to advertising campaign i. We therefore have that the probability of winning impression j, is $p_j \triangleq \sum_{i=1}^{m} x_{ij}$.

The allocation decision directly translates to a bid level decision as follows: We assume a stationary probability of winning $F(b)$ as a function of the bid b. For now,

we assume that F is given. Assuming that F is strictly monotone in the range $[0, B]$ (hence, invertible) we can associate every bid $b \leq B$ with a probability of winning p such that $b = F^{-1}(p)$. Now, assuming a first price auction at the AdX, the average cost to the ad network when bidding with a probability of winning p is $c(p) \triangleq pF^{-1}(p)$. We assume (as customary in the literature) that $c(p)$ is convex.

Given the impressions $A = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n)$, the cost function $c(\cdot)$ and the utility functions $\{u_i(\cdot)\}_{i=1}^m$, the ad network problem is to find an optimal (maximal net income) allocation $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ of impressions to contracted campaigns:

$$\max_X \sum_{i=1}^{m} u_i(r_i) - \sum_{j=1}^{n} c(p_j) \tag{5.12}$$

subject to

$$\sum_{j=1}^{n} x_{ij} a_{ij} = r_i \qquad i \in \{1, \ldots, m\} \tag{5.13}$$

$$\sum_{i=1}^{m} x_{ij} = p_j \qquad ji \in \{1, \ldots, n] \tag{5.14}$$

$$p_j \leq 1 \qquad j \in \{1, \ldots, n\} \tag{5.15}$$

$$x_{ij} \geq 0 \qquad j \in \{1, \ldots, n\}, i \in \{1, \ldots, m\} \tag{5.16}$$

One might be tempted to formulate the ad network problem as an instance of the Adwords Problem [50] or the more general Online Stochastic Packing problem [55]. Those works describe a general method for solving an online allocation problem that is formulated as an LP. They do so by first learning the optimal strategy (solving an LP) based on a fraction of the samples, and then applying it to the rest of the samples. Our model deviates from their setting since the objective is no longer a linear function of the inputs and their method can't be directly applied. Moreover, in the hard-limit budget constraint of their models is generalized in ours by a concave utility function.

A more relevant model was introduced in [19] where the Publisher's Problem is considered: Similar to our setting, the publisher has a set of contracted advertisers and is committed to allocate to each advertiser a portion of its periodic impressions (still, each contract has a hard-limit budget constraint). The publisher's decision at time j is twofold: the Ad Exchange impression acceptance probability s_j (or equivalently, the

reserve price r_j - if another, unrelated, advertiser wins the auction then the publisher is paid r_j and no contracted advertiser gets the impression) and the eventual probabilities of allocating the impression to the pre-contracted advertisers in case the reserve price was set too high (i.e., all bids at the AdX were below the reserve price). Again, a probability of acceptance $1 - F(r)$ is assumed for a reserve price r, where $F(\cdot)$ is the c.d.f. of highest bids at the AdX.

Given the impressions relevance to advertisers $Q = (\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n) \in \Re^n$, contract capacities ratios $\rho = (\rho_1, \ldots, \rho_m)$ and the concave revenue function $r(\cdot)$, the Publisher's Problem is to find an optimal (maximal revenue and advertisers value) allocation of impressions to contracted advertisers, and reserve price acceptance probability s_j:

$$\max_{\mathbf{s},X} \sum_{j=1}^{n} r(s_j) + \mathbf{x}_j^t \mathbf{q}_j \tag{5.17}$$

subject to

$$\sum_{j=1}^{n} \mathbf{x}_j = N\rho \tag{5.18}$$

$$\sum_{i=1}^{m} x_{ij} + s_j \leq 1 \qquad j \in \{1, \ldots, n\} \tag{5.19}$$

$$s_j \geq 0 \qquad j \in \{1, \ldots, n\} \tag{5.20}$$

$$x_{ij} \geq 0 \qquad j \in \{1, \ldots, n\}, i \in \{1, \ldots, m\} \tag{5.21}$$

Where $r(s) = sF^{-1}(1 - s)$ is the expected revenue to the publisher from the AdX when setting the reserve price to be accepted with probability s. Note that in a sense this problem is more general than the AdNet problem since the relevance $\mathbf{q}$ of the impression is not restricted to binary values. In another way, this problem is simpler than the AdNet problem since the concave utilities in the AdNet problem are replaced here with linear functions.

5.6 Implementation

The competing AdNetwork agents communicate with the game server over the Internet. The server communication details (address, ports, and additional agent information

such as the agent name and password) are detailed in a dedicated configuration file used at agent runtime. Once an agent is registered at the server (using the server's web interface), it may join and take part in games (using the server's game interface). The game status may be observed in a dedicated GUI that is accessible through the web interface of the game server, which also enables access to game results, history, and related logs.

The implementation of the AdX game, available through the git repository [8], is based on the JAVA implementation of the TAC Ad Auctions game [7] which in turn mainly depends on the SICS infrastructure for TAC, a framework that was originally developed for the SCM game [11]. The AdX game reuses as much as possible of the Ad Auctions game mechanisms and high level architecture, many mechanisms (e.g., remote messaging, web based GUI for game and competition administration, and configuration scheme) minimally adapted (if at all).

5.6.1 Architecture

The key entity of the AdX game implementation is the *Simulation*. A simulation instance is created to execute a single game among a set of predefined competing agents (therefore, a competition consists of a sequence of instantiations of a simulation) and is terminated upon game completion. Figure 5.11 illustrates the internal structure of a simulation instance: A *Users* component that implements the users population, with users' attributes generated according to the distributions configured in the game server's configuration file, and (similarly configured) a *Publishers* component that implements the web sites and their related user orientation statistics. A simulation also includes an *Agent Messaging* component that acts as an interaction proxy to the competing Ad Network agents, an *Internal Message Bus* that enables local one-to-many communications by implementing a publisher-subscriber pattern, and an *Activity Scheduling* component that coordinates the activities of the two agents implementing the game - the *Demand Agent* and the *Supply and AdX Agent*. Both agents are described in detail in later sections.

Scheduling of activities in the simulation is done by a timed thread that wakes up every logical day (lasting a configurable period of 10 real seconds) and calls back to the callNextTimeUnit handler of the simulation instance. The simulation instance in turn calls back to the nextTimeUnit handler of each registered game agent instance

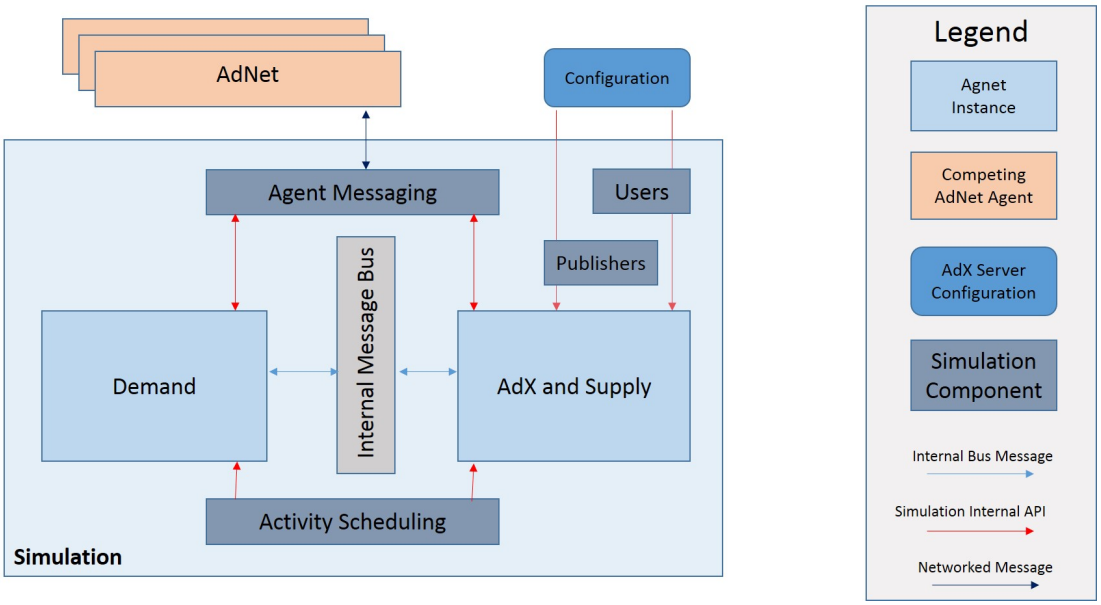


Figure 5.11: Simulation: Major components and interactions.

(implementing the TimeListener interface). The activation of agents is done however in two phases. First, the simulation agent spawns the actions of the demand agent through a call to its preNextTimeUnit callback, giving the Demand agent the opportunity to handle campaigns creation and allocation (among other things to be detailed below), and only then the NextTimeUnit of registered agents is called, giving the AdX and Supply agent the opportunity to simulate the user's population visits to publisher's web sites and related AdX auctions (and other related activities as detailed below). Before activating the game agents, the simulation instance sends the Bank Status message (implicitly indicating the start of a new day to the competing Ad Networks). After all agents completed their daily actions the Simulation Status message is sent by the simulation instance to the competing Ad Networks (this will trigger the competing agents sending of their daily decisions back to the game agents).

5.6.1.1 Demand

The demand agent implements all aspects of the daily campaigns (creation, allocation, tracking, etc.), the Users Classification Service manager, and the Ad Network's Quality Manager. It interacts with the competing Ad Networks (i.e., sending related notification and reports, receiving bids) and with the other system components (e.g., getting notifications upon user impressions) through different messaging and callback mechanisms, as illustrated in the diagram of Figure 5.12: During simulation day n,

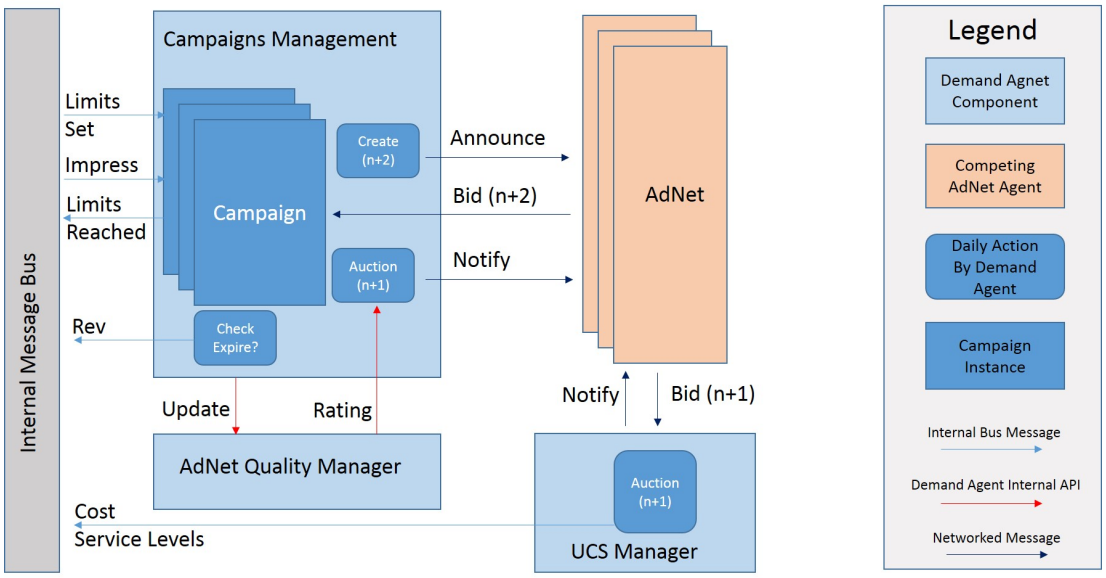


Figure 5.12: Demand: Major components and interactions.

the campaign to start at day $n + 2$ is created and announced to the competing Ad Networks' agents. The AdNetworks respond later (that same day) with related bids. The campaign that was announced the previous day (starting at dat $n + 1$) is auctioned and the results are notified to the Ad Networks' agents. Similarly for the Users Classification Service, the service level to be in effect on day $n + 1$ is auctioned during simulation day n (based on bids received earlier that same day) and notified to the the competing Ad Networks' agents.

During the simulation, user impressions and the associated campaign allocation are notified to the demand agent through a dedicated *impress* message of an internal messaging bus (which also serves the demand agent to notify other entities regarding Ad Network's campaign revenue and UCS levels and costs). Finally, upon a campaign's last, day the resulting reach ratio (5.7) and income (5.8) are calculated and informed respectively: internally to the Quality Manager and through the message bus to the bank entity. The Quality Manager updates the quality rating of the Ad Network accordingly using (5.9) - this rating is used thereafter to squash the campaign bids of the Ad Networks.

5.6.1.2 Supply and AdX

In its daily routine, the *AdX Users Manager* iterates over all users and performs the following for each: First (indicated by **A** in Figure 5.13), a user *query* is generated by

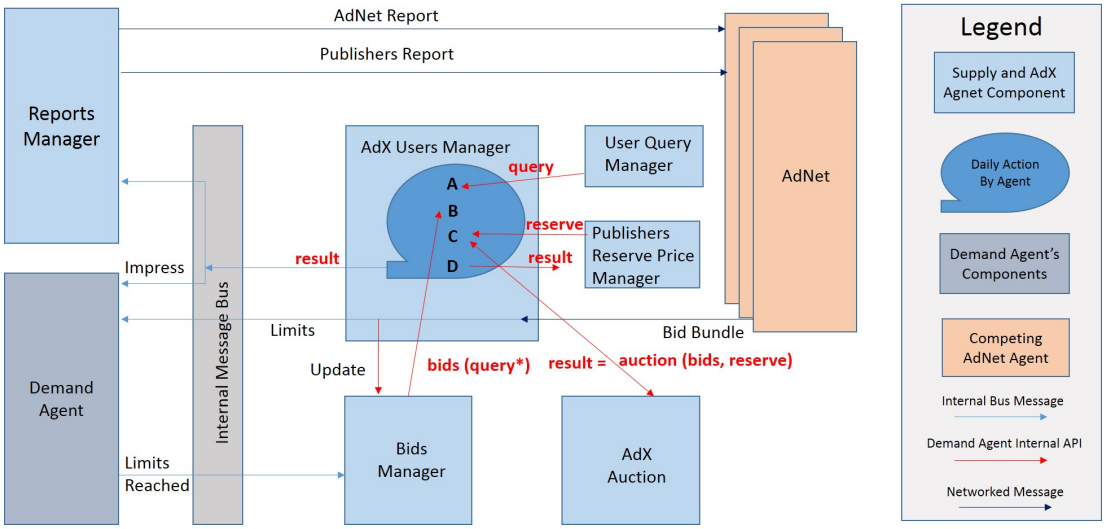


Figure 5.13: Supply and AdX: Major components and interactions.

the *User Query Manager*. The query represents a user visit to a publishers' web site (that is, an impression opportunity) and is randomly generated according to the web site's orientation configuration of (5.4) as detailed later in this section. The bids of the Ad Networks for the impression opportunity are then collected (**B** in Figure 5.13) from the *Bids Manager* - this is done by consulting the applicable BidBundles received from the Ad Networks. The user attributes of *query** (the query used to fetch the bid of each Ad Network) may be masked, according to the User Classification Service level in effect of the Ad Network. A second-price auction is then performed based on the Ad Networks bids and a reserve price that is fetched from the related *Publisher's reserve price manager* (**C** in Figure 5.13). Finally (**D** in Figure 5.13)), the result of the auction (the winning advertiser's campaign and cost) is communicated back to the reserve price manager (this will be used to later update the reserve price baseline according to (5.2)) and posted in the internal message bus for the demand agent to update campaign statistics accordingly (potentially resulting in the campaign limits being reached, triggering a further message bus notification to the Bids Manager). The auction's result message is also triggered by the *Reports Manager* which updates the associated Adnet and Publisher's reports accordingly. The daily consolidated reports (containing information about the previous day) are sent to the competing Ad Network at the beginning of the daily routine.

In what follows, some key aspects of the design and implementation of each of the modules mentioned above are detailed. The **User Query Manager**, implemented in

the tau.tac.adx.users package, creates upfront a *sampler* for each user (a sampler is an object that allows to randomly choose an item from a weighted list of items). The items of each users' sampler are the set of the user's possible queries[18] and the weight of each item (primarily characterized by the web site to be visited) is determined by the user's conditinal probabilities (5.4) as configured in the server configuration file and maintained by a dedicated *Publishers* instance of the simulation.

The **User Bids Manager**, implemented by a *AdxBidManagerImpl* class in the tau.tac.adx.auction.manager package provides the bid of each publisher given a user's query by maintaining for each Ad Network the bidding map from the most recent Bid Bundle received. Recall that to simulate real-time bidding, and to avoid the associated messaging load, the competing Ad Networks pass a daily Bid Bundle that contains a mapping of the potential context of an impression opportunity (the publisher's web site, the user's market segments, the access device and the ad type) to a related bid and campaign allocation. Each Ad Network's bid is generated by a sampler over the Ad Network's Bid Bundle entires that match the query (with the item's weight as specified in the entry), where a match is declared when the market segments indicated in the Bid Bundle entry are all contained in the market segments set of the query (and all other query attributes of the entry, i.e., publisher's web site, access device, and ad type, are identical). Note that before passing the query to the User Bids Manager, the original query is masked (resulting in passing *query**) according to the User Classification Service level applicable to the Ad Network and with some probability an empty set of market segments may be passed (a match in that case would be for entries having the empty market segment set - the "unknown" market segment). The Users Bids Manager also maintains a daily list of excluded campaigns. Campaigns are added to this list upon indication through the message bus that the limits for a campaign were reached (spend limits or impression limits, daily limits or total campaign limits) and bid bundle entires with campaigns for the list are ignored.

The **Publishers Reserve Price Manager** implements (5.1) upon every user query by perturbing the maintained baseline reserve associated to each possible impression context (publisher, access device, and ad type). The baselines are updated daily using (5.2) based on the cached auction results indicated. The AdX auction run by the **AdX Auction** component (mainly implemented by an instance of the *SimpleAdxAuctioneer*

[18]Recall that a query represents a user visit to a web site and is also characterized by the access device used and the ad type.

class of the tau.tac.adx.sim package) uses the Ad Network's bids provided by the Users Bid Manager and the reserve price from the Publishers Reserve Price Manager to run a second-price auction.

5.6.2 Parameters

The execution of the AdX game depends on several parameters that are pre-configured (in a dedicated configuration file) and known to the competing agents. Defining such parameters gives flexibility and allows tailored execution of the game (changing the default value of some parameters, however, should be done in conjunction with other parameters, e.g., to maintain game balance). The value of parameters, however, may influence the agent's strategies (e.g., for training) and therfore must be set ahead of time (to allow for proper design and training). Table 5.1 details the many game parameters and their standard values.

Table 5.1: AdX Game Parameters and Standard Values

Parameter	Symbol	Std. Setting
User Continuation Probability	$P_{\texttt{Continue}}$	0.3
Random Campaign Allocation Probability	$P_{\texttt{RandomCampaign}}$	0.36
Max User Daily Impressions	$N_{\texttt{ContinueMax}}$	6
Publisher's Initial Reserve Price	$R_{\texttt{ReserveInit}}$	0.005
Reserve Price Variance	$R_{\texttt{Variance}}$	0.02
Reserve Price Learning Rate	$R_{\texttt{LearnRate}}$	0.2
Short Campaign Duration (Days)	$C_{\texttt{CampaignL1}}$	3
Medium Campaign Duration (Days)	$C_{\texttt{CampaignL2}}$	5
Long Campaign Duration (Days)	$C_{\texttt{CampaignL3}}$	10
Low Campaign Reach Factor	$C_{\texttt{CampaignLowReach}}$	0.2
Medium Campaign Reach Factor	$C_{\texttt{CampaignMediumReach}}$	0.5
High Campaign Reach Factor	$C_{\texttt{CampaignHighReach}}$	0.8
Max Campaign Cost Per Impression	$R_{\texttt{CampaignMax}}$	0.001
Min Campaign Cost Per Impression	$R_{\texttt{CampaignMin}}$	0.0001
Quality Rating Learning Rate	$L_{\texttt{Rating}}$	0.6
Game Length	$T_{\texttt{Gamedays}}$	60
Real Time Seconds per Simulated Day	$T_{\texttt{Dayseconds}}$	10
User Classification Service Revelation Probability	$P_{\texttt{UserRevelation}}$	0.9
Initial Days Classification Service Accuracy	$Z_{\texttt{UCSaccuracy}}$	0.9

5.6.3 Real Data

The game setting also depends on the user population distribution (Table 5.2) and the web site's orientation (Table 5.3), which are based on real figures from [38], [3]. Note that in Table 5.2 the income is in $1000 units and the probabilities are in 0.0001 units.

Table 5.2: User Population Probabilities, in 0.0001 units

Age	Gender	Income	Probability	Age	Gender	Income	Probability
18-24	M	0-30	526	25-34	M	0-30	371
35-44	M	0-30	263	45-54	M	0-30	290
55-64	M	0-30	284	65+	M	0-30	461
18-24	F	0-30	546	25-34	F	0-30	460
35-44	F	0-30	403	45-54	F	0-30	457
55-64	F	0-30	450	65+	F	0-30	827
18-24	M	30-60	71	25-34	M	30-60	322
35-44	M	30-60	283	45-54	M	30-60	280
55-64	M	30-60	245	65+	M	30-60	235
18-24	F	30-60	52	25-34	F	30-60	264
35-44	F	30-60	255	45-54	F	30-60	275
55-64	F	30-60	228	65+	F	30-60	164
18-24	M	60-100	11	25-34	M	60-100	140
35-44	M	60-100	185	45-54	M	60-100	197
55-64	M	60-100	157	65+	M	60-100	103
18-24	F	60-100	6	25-34	F	60-100	75
35-44	F	60-100	104	45-54	F	60-100	122
55-64	F	60-100	109	65+	F	60-100	53
18-24	M	100+	5	25-34	M	100+	51
35-44	M	100+	125	45-54	M	100+	163
55-64	M	100+	121	65+	M	100+	67
18-24	F	100+	3	25-34	F	100+	21
35-44	F	100+	47	45-54	F	100+	57
55-64	F	100+	48	65+	F	100+	18

5.7 AdX Game Competitions and Conclusion

The performance of an agent implementing the Ad Network strategy may vary across games. This may be due to inter-game adaptations done by competing agents (as part of a higher-level strategy) but also because of the stochastic nature of the game simulation. Therefore, a competition comprising of several games is required in order to assess the

Table 5.3: Publisher's audience orientation, access device ratios and popularity, for news, shopping, and web information services

Name	18-24	25-34	35-44	45-54	55-64	0-30	30-60	60-100	Male	Mobile	Pop.
Yahoo	12.2	17.1	16.7	18.4	16.4	53	27	13	49.6	26	16
CNN	10.2	16.1	16.7	19.4	17.4	48	27	16	48.6	24	2.2
NY Times	9.2	15.1	16.7	19.4	17.4	47	26	17	47.6	23	3.1
Hfngtn	10.2	16.1	16.7	19.4	17.4	47	27	17	46.6	22	8.1
MSN	10.2	16.1	16.7	19.4	17.4	49	27	16	47.6	25	18.2
Fox	9.2	15.1	16.7	19.4	18.4	46	26	18	48.6	24	3.1
Amazon	9.2	15.1	16.7	19.4	18.4	50	27	15	47.6	21	12.8
Ebay	9.2	16.1	15.7	19.4	17.4	50	27	15	48.6	22	8.5
Wal-Mart	7.2	15.1	16.7	20.4	18.4	47	28	19	45.6	18	3.8
Target	9.2	17.1	17.7	18.4	17.4	45	27	19	45.6	19	2.0
BestBuy	10.2	14.1	16.7	20.4	17.4	46.5	26	18	47.6	20	1.6
Sears	9.2	12.1	16.7	20.4	18.4	45	25	20	46.6	19	1.6
WebMD	9.2	15.1	15.7	19.4	18.4	46	26.5	18.5	45.6	24	2.5
EHow	10.2	15.1	15.7	19.4	17.4	50	27	15	47.6	28	2.5
Ask	10.2	13.1	15.7	20.4	18.4	50	28	15	48.6	28	5.0
TripAdvisor	8.2	16.1	17.7	20.4	17.4	46.5	26	17.5	46.6	30	1.6
CNet	12.2	15.1	15.7	18.4	17.4	48	26.5	16.5	50.6	27	1.7
Weather	9.2	15.1	16.7	20.4	18.4	45.5	26.5	18.5	47.6	31	5.8

quality of competing agents. A competition is scheduled through a dedicated web interface of the game server in which its duration (number of games, interval between games) and participating agents are set.[19] Usually, aiming at statistical significance, competitions last 40 games or even more (this of course may depend on the variance of the game performance and score differences of the competing agents).

The first AdX game competiton took place on May 5^{th} and 6^{th} as part of the AMEC/TADA 2014 workshop [2] (part of AAMAS [1]), where the agent from Shanghai Jiao Tong University won the first place [117] out of eight competing agents by teams from other universities.[20] A subsequent competition was held as part of a CS workshop dedicated to the AdX game that was held in Tel Aviv University during the fall semester of 2013.[21] In the workshop, six teams of students designed and im-

[19] A web interface may also be used to access interim and final results of the competition, detailing for each agent its per-game scores and other related metrics

[20] Tel Aviv University (3 agents), University of Liverpool, University of Edinburgh, University of Michigan, and Brown University.

[21] The competition took place upon the end of the workshop, in May 2014.

plemented the Ad Network strategy for an agent of the AdX game. The concluding competition was held as part of the workshop during May 2014 and the teams reported their work [10]. The reports by the teams and the phenomena observed reinforce the relevance and applicability of the AdX game, both as a test-bed for related Ad Network strategies and as a research platform for the choice of the different mechanisms simulated (e.g., empirically evaluate the effect of replacing the reserve price optimization method detailed in Section 5.3.2.2 by the one proposed in [89]).

More specifically, the teams reported the importance of establishing a high quality rating early in the game, during the inherent fierce competition over users impressions as competing agents aim at fulfilling the pre-assigned campaigns. Failing to properly execute those first campaigns results in a low quality rating that is hard to overcome, even in the presence of randomly allocated campaign. Moreover, towards the end of a game, remaining as one of the few competing agents with a permissible quality rating, gives a great opportunity to leverage the situation (reminiscent of a monopoly) and increase profits considerably. This resulting partition of the game to a strong opening period in which credibility is established (even at the cost of sacrificing profits), followed by a *middle game* epoch of survival, and an *end game* situation in which the few remaining competitors are highly rewarded seems natural and suggest that the AdX setting may be poised for technology transfer (in the sense that key insights and related algorithms regarding agents strategies, such as those just mentioned, may be relevant in real scenarios).

The UCS bidding strategies reported mainly aim at learning from past reports the bid level needed for getting a required service level and then consider the agent's campaign's state (e.g., needed amounts of impressions and remaining days) to choose the level to aim for. Interestingly, the agents did not directly analyze the economic benefits of the UCS service level (probably due to the campaigns interdependence, since the UCS cost has to be allocated across all ongoing campaign to carry a meaningful analysis). Finally, for the AdX bid bundles, the main factor most agents took into account was the expected competition level (resulting from the attributes of the overall campaigns allocated to other competitors, such as the reach and days left) and adjust their bids accordingly.

All in all, the AdX game provides a test bed for assessing the effects and performance of the many different mechanisms used by the different entities in the AdX setting. Not

in isolation of course, but with respect to the implementations of the other entities' mechanisms and algorithms. A key purpose of the game is for this to be facilitated in the future by the availability of different implementations of the different strategies (that is, of competing agents, but also, for example, of modular components for alternative reserve price optimization mechanisms at the publishers), and used by researchers as a tool to empirically evaluate related mechanisms and algorithms in a controlled context and environment.

Part II

Robust Domain Adaptation

Chapter 6

The Domain Adaptation Problem

This background chapter presents the domain-adaptation problem and related learning settings, followed by a short review of the theory and algorithms relevant to the main result to be presented in Chapter 8.

6.1 Train-Test Discrepancy - Motivation

A key assumption underlying the generalization bounds and results of Supervised Learning theory[1] is that the performance of the learning algorithm is evaluated in a 'test environment' that is statistically identical to the 'training environment' (the source of the training samples). Furthermore, this assumption of test and training environments being the same allowed for methods that are agnostic of such statistical properties altogether! That is, generalization is possible even without estimating the probability distribution over which samples are drawn. Nevertheless, in many practical situations, due to a variety of reasons, such an assumption might not hold.

For example, a gender classification algorithm could be first trained using a set of portraits of persons from one geographical area (say the USA) and its resulting classifier may be later required to classify portraits of persons from a different country. Another example is Spam email filtering - where a trained and deployed classifier might face changes in the wording used in Spam emails (compared to the words used for training) by malicious senders trying to avoid detection or in case it was trained for a certain set of users while used by others.

The reason for not re-training may be the availability or associated cost of labeling

[1] And as a result, a crucial justification for Empirical Risk Minimization (ERM) algorithms

a new training set, or lack of time. To overcome a lack of labeled training data (mainly due to the cost of labeling), the training set itself may be created using synthetic data (in which the sample's label is known upfront). Such an approach is used, for example, for detecting humans in crowded scenes by training using simulated (yet realistic) video game scenes. The resulting detectors however are required to perform well when fed with real scenes. Yet another such media classification setting is voice recognition. Consider the task of identifying spoken words. The conditions during training (including background conditions such as noise but also inherent conditions of the speaker that may depend on the time of day, tiredness, state of mind, etc.) might significantly differ from the conditions where and when the actual classifier is used (we might even want to use it for other speakers!) and should be accounted for.

Now, email Spam detection mentioned earlier is just one of a long list of Natural Language Processing such settings: A sentiment analysis classifier that is trained to detect positive or negative hotel reviews and then applied to restaurant reviews, A Part of Speech (POS) tagger trained on a corpus of medical documents to be later used to tag over a different domain, and the list goes on.

Considering all the above-mentioned settings, two key questions stand out:

- How will the difference between the training and testing environments affect the performance of the learning algorithm ?

- How can we adapt the learning algorithm to achieve better test performance ?

Addressing each of the above questions may be valuable. Knowing the impact of not re-training or having methods to adapt existing algorithms[2] can assist in the decision whether to train again (that is, to invest in the process of labeling target-domain samples) or not. Indeed, methods to bound the generalization performance in such settings[3] as well as algorithms accounting for the train-test differences were offered in the last few years (see the following two reviews [102, 114]). A model for the train-test discrepancy problem and a review of related algorithms is presented in the next section. Within that context, the subsequent section introduces the specific *domain-adaptation* problem treated in our work and related theoretical results.

[2]Based on partial information or prior knowledge regarding the target domain.
[3]And also some impossibility results, to be described.

6.2 Train-Test Discrepancy - Learning Settings

In the Supervised Learning setting we are given a sample (*training*) set $S = \{(x_i, y_i)\}_{i=1\cdots m}$, where the samples are generated i.i.d. from a distribution $Q(x, y)$. As discussed in Section 1.1, a learning algorithm in this setting is required to come up with a hypothesis (that is, a mapping h that given only the attributes x of a *test* item (x, y) provides a prediction $\hat{y} = h(x)$, for the *actual* label y) of small expected loss

$$\mathcal{L}_Q(h) = E_{(x,y)\sim Q} l(h(x), y) \ ,$$

where $l(\cdot, \cdot)$ is a predefined loss function.[4] Indeed, by Theorem 1, ERM algorithms that given sample set S minimize the empirical loss

$$\hat{\mathcal{L}}_S(h) = \frac{1}{m} \sum_{i=1}^{m} l(h(x_i), y_i)$$

need a manageable number of training samples to result with high probability in a hypothesis of small error.[5] Furthermore, ERM algorithms are model-free as they avoid the need to estimate the *model* $Q(x, y)$ as a step in computing the hypothesis h.

Those key properties of ERM algorithms highly depend on the assumption that the test items (x, y) are drawn from the same *source* distribution $Q(x, y)$. Otherwise, in general, learning is impossible without modeling and assessing the relation between the training $Q(x, y)$ and testing $P(x, y)$ sample distributions. This is easily verified by considering a binary classification setting in which the intersection of the supports of $Q_X(x)$ and $P_X(x)$ is empty. Any training set used by a learning algorithm in this setting is irrelevant since the sought-after test mapping $P_{Y|X}(y|x)$ may be arbitrary, and not related in any way to the observed samples that follow $Q_{Y|X}(y|x)$.[6]

In practice, there are many reasons causing the difference between the training domain $Q(x, y)$ and test domain $P(x, y)$, each requiring a different model and resulting in a dedicated algorithmic approach. A short review of the different models follow,

[4] For example, the zero-one loss $l(x, y) = 0$ if x equals y and otherwise 1, widely used in binary classification settings.

[5] More precisely, with $1 - \delta$ probability, the resulting hypothesis has error that is ϵ close to the best possible among all $h \in H$.

[6] Technically, there exists a hypothesis class H such that for any learning algorithm in this setting and for every $\epsilon > 0$, there exists a test distribution P such that with positive probability its output $h \in H$ has at least ϵ expected error. Therefore, it is not PAC-learnable.

mainly considering change[7] in the elements $Q_X(x)$, $Q_{Y|X}(y|x)$, $Q_Y(y)$ and $Q_{X|Y}(x|y)$ of the breakdown $Q(x,y) = Q_{Y|X}(y|x)Q_X(x) = Q_{X|Y}(x|y)Q_Y(y)$.

6.2.1 Covariate Shift: $P_{Y|X} = Q_{Y|X}$

A setting in which the conditional $Q_{Y|X}(y|x)$ remains unchanged (that is, for all x and y, $P_{Y|X}(y|x) = Q_{Y|X}(y|x)$) is called *Covariate Shift*. In this setting, a difference between the source and target probabilities may exist in the covariate probabilities $Q_X(x)$ and $P_X(x)$ (where for some x, $P_X(x) \neq Q_X(x)$). This typically occurs in situations where there is a casual relationship $Q(y|x)$ that is independent of the way x was generated. Anticipation of the future occurrence of a medical situation given current habits, for example, may be suitable for covariate shift modeling (e.g., when the habits of sampled individuals differ in the source and target domains due to different circumstances). Another example of a covariate shift setting is the estimation of a demographic label y (such as age) based on face image attributes x of a person (with test environment having different conditions, say lighting, that are independent of x).

Being also one of the simplest forms of train-test discrepancy, many algorithms were suggested to cope with covariate shift (see for example [31, 115]), all applying the common *importance weighting* technique. In importance weighting, a non-negative weight w_i is assigned to each sample (x_i, y_i) of the training set S. The weighted sample set is designated S_w. The weight w_i may be interpreted as the relative cost of the output hypothesis making an error on this sample. Hence, the weighted empirical error of a hypothesis h on the weighted sample S_w is

$$\hat{\mathcal{L}}_{S_w}(h) = \sum_{i=1}^{m} w_i l(h(x_i), y_i) \ .$$

Any supervised learning algorithm may then be applied to a re-sampled training set in which the multiplicity of a sample (x_i, y_i) is relative to w_i.

To illustrate the so called *importance-weighting* method, consider the following example of regression. Assume our aim is to estimate the relation $y = f(x)$ by using a hypothesis class $H = \{f_\theta : \theta \in \mathcal{R}^{n+1}\}$, where n is fixed and for $\theta = (\theta_0, \theta_1, \theta_2, \ldots, \theta_n)$ we have $f_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_n x^n$[8]. Now, for the quadratic loss $l(\hat{y}, y) = (\hat{y} - y)^2$,

[7] That is, the difference, in Q vs. P

[8] For example, For a linear relation $f(x) = ax + b$ the parameters are $\theta = (a, b)$ and for a quadratic relation $f(x) = ax^2 + bx + c$ the parameters are $\theta = (a, b, c)$.

a weighted regression algorithm given a sample set S with weights w_i should minimize

$$\theta^* = \arg\min_\theta \frac{1}{m} \sum_{i=1}^{m} w_i (f_\theta(x_i) - y_i)^2 \ .$$

Similarly, we can get importance-weighted versions of many other supervised learning algorithms such as SVM, Boosting, Decision Trees, and Logistic Regression.

Now, a common choice for the weights is

$$w_i = \frac{P(x_i, y_i)}{Q(x_i, y_i)} = \frac{P_X(x_i)}{Q_X(x_i)}.$$

This choice is motivated by the empirical weighted loss being an unbiased estimate of the *target* loss (see e.g., Proposition 1 in [46]):

$$E_{S \sim Q^m} \hat{\mathcal{L}}_{S_w}(h) = \mathcal{L}_P(h) = E_{(x,y)\sim P} l(h(x), y) \ ,$$

further motivating Importance-Weighted ERM algorithms that minimize the empirical weighted loss

$$h^* = \arg\min_{h \in H} \hat{\mathcal{L}}_{S_w}(h) \ .$$

In practice, however, the weights w_i are rarely given and have to be estimated. Nevertheless, even when the weights are perfectly known, the importance weighting method might miserably fail (see e.g., Figure 1 in [44]) when the hypothesis class H does not include f, the true underlying relation $y = f(x)$. Learning bounds relating $\hat{\mathcal{L}}_{S_w}(h)$ and $\mathcal{L}_P(h)$ are presented in [44] for both bounded and unbounded loss functions. The effects of having to estimate the weights w_i (assuming availability of unlabeled samples from the target domain, to estimate P_X) and the related stability of some popular learning algorithms facing estimation error is analyzed in [46].

6.2.2 Prior Shift: $P_{X|Y} = Q_{X|Y}$

In a *Prior Shift* setting we assume a casual relation $Q_{X|Y}(x|y)$ that remains unchanged while the difference between the source and target probabilities is in the prior probability $Q_Y(y)$ (that is, for all x and y, $P_{X|Y}(x|y) = Q_{X|Y}(x|y)$ and for some y, $P_Y(y) \neq Q_Y(y)$). Prior shift typically occurs in *Imbalanced Data* situations, where the training set samples are filtered to increase the portion of samples with rare values

of y (e.g., a reasonable practice when the whole purpose of the learning task is the prediction of rare events, which would not make it into the training set otherwise). Compensating for prior shift is easy in the imbalanced data case[9] by associating a sample weight

$$w_i = \frac{P(x_i, y_i)}{Q(x_i, y_i)} = \frac{P_Y(y_i)}{Q_Y(y_i)} \ ,$$

and employing ERM. Alternatively, some learning algorithms directly use the conditional label probability $P_{Y|X}$ (e.g., Logistic Regression), which may be recovered using the observed $Q_{Y|X}$ by

$$P_{Y|X}(y|x) = \frac{Q_{X|Y}(x|y)P_Y(y)}{P_X(x)} = \frac{Q_{Y|X}(y|x)w(y)Q_X(x)}{P_X(x)} = \frac{Q_{Y|X}(y|x)w(y)}{\sum_{y'} Q_{Y|X}(y'|x)w(y')} \ ,$$

where the importance weight at y is defined $w(y) \triangleq \frac{P_Y(y)}{Q_Y(y)}$. Note that in both cases (using ERM, or $P(y|x)$ directly) access to the original $P_Y(y)$ or an estimate is essential.[10]

6.2.3 Other Related Settings: $P_X = Q_X$ and Beyond

The $P_X = Q_X$ setting and other related models of train-test discrepancy are presented in this section. Later on, in the subsequent section, the general case[11] of train-test discrepancy adaptation when unlabeled target domain samples are available is addressed.

An adaptation setting in which the covariates probabilities remain unchanged (that is, for all x, $P_X(x) = Q_X(x)$ and for some x and y the labeling function differs $Q_{Y|X}(y|x) \neq P_{Y|X}(y|x)$) can be treated as a special case of *transfer learning*. In Transfer Learning (also called multi-task learning, see [97] for a somewhat recent survey) there are multiple *tasks* $i = 1, \ldots, n$, each characterized by an unknown (and sought for) labeling function $Pr_i(y|x)$, but sharing a common input space and distribution $Pr(x)$. Transfer Learning algorithms aggregate (thereby *transfer*) relevant observations across tasks by exploiting modeled inter-task relations. Therefore, when labeled target-domain data is available (i.e., the so called *semi-supervised domain adaptation* setting) the adaptation problem is an instance of two-tasks transfer learning (where one task is the source domain and the other is the target domain) and related algorithms

[9] Note that the *target* distribution in this setting is the original pre-filtered P.

[10] A method to cope with unknown $P_Y(y)$ in a discrete label setting is presented in [41].

[11] That is, when no assumptions are made regarding the discrepancy of $P(x, y)$ and $Q(x, y)$ in terms of the covariate or conditional breakdown.

may be applied.

Closely related to the transfer learning model are settings in which relatively large training data sets are available for a set of source domains together with a relatively small training set for the target domain. Algorithms that combine the source domain data (either directly, or through *experts* - learning algorithms, each trained on the data of a specific source domain) to result in a hypothesis for the target domain were proposed for this setting, both for classification [47, 103] and regression [100], with the latter reporting usage of such methods for predicting resulting auction prices in the TAC-Travel [62] and TAC-SCM [105] games.

Another related model is *Mixture* Shift. In a mixture shift setting the data source is a probabilistic mixture of several domains, all sharing the same labeling function (i.e., each having a different distribution over the sampled x values), and the mixing proportions vary between the training and target scenarios. A learning algorithm in this setting is also given a set of well-performing hypotheses, one for each domain, and is required to output a combination of the given hypotheses to perform well in the target mixture. Such a setting is presented and analyzed in [82], where a simple adaptation algorithm is proposed and the existence of an optimal combining rule is established.

Yet another related setting is *Active Learning* [43]. In this setting the train-test discrepancy is controlled, as the learning algorithm has the power to choose the attributes x of each training sample. Inherently, active learning results in covariate shift, and although the somewhat different objective[12] some solution approaches are similar in their use of sample re-weighting through importance sampling [71].

Finally, the above mentioned *imbalanced data* settings (where the observed *source* training set is a filtered version an original, yet inaccessible sampled *target* domain) may be all framed as a special case of *Sample Selection Bias*. Sample selection bias situations are common in population surveys where the sampling procedure causes a bias (e.g., a survey conducted by phone calls ignores people that don't have a phone), where training data is discarded for 'cleaning' purposes, and in several other settings. Noting that this is basically a domain shift situation (since the actual testing is performed in the original - unfiltered - domain), typical bias correction methods to address sample selection bias perform importance weighting (see e.g., [46]). Note however that in general, sample selection bias (and many other domain adaptation settings) do not necessarily fall in

[12] Active learning aims at generating a training set, rather than directly resulting in a hypothesis of minimal generalization error

one of the special cases detailed in this section (e.g., covariate shift, prior shift). We therefore turn now to a precise definition and review of the general case for domain adaptation in the presence of unlabeled target data.

6.2.4 Domain Adaptation with Unlabeled Target Data

In the general case of domain adaptation, the learning algorithm might be facing an arbitrary change in the original *training* domain $Q(x, y)$ compared to the *test* domain $P(x, y)$. We refer to Q as the *source* distribution and to P as the *target* distribution. Since generalization for the target domain given only source domain samples is impossible in general, the model also assumes availability of unlabeled data from the target domain (indeed, in many applications, the actual labeling is the main cost factor in obtaining a training set and abundance of unlabeled data from the target domain is typical).

The learning algorithm is therefore provided with a labeled-samples set S of source domain samples drawn i.i.d. from Q, and unlabeled samples set T of target domain samples drawn i.i.d. from P. The learning algorithm is then required to output a predictor h from a predefined set of predictors H such that h performs well (that is, h has small average prediction error) in the target domain.

The setting is now formalized: Let X be the input space and Y be the output space.[13] The (unknown) input-output relation may be captured by a deterministic target function $f\colon X \to Y$ or by a probability distribution $f\colon X \to \Delta_Y$.[14] A domain $D(x, y)$, a distribution over the input-output space is characterized by the pair (D_X, f_D), where we denote by D_X the marginal of $D(x, y)$ over the input coordinates, and f_D is either a deterministic function or a conditional probability distribution as indicated above.

Let the input-output space be $Z = X \times Y$, and let H be a hypothesis class of functions $h : X \to Y$, used to learn f. For a bounded non-negative loss function $l : Y \times Y \to [0, M]$, we define the point-wise loss of h with respect to a labeled example $z = (x, y)$,

$$l(h, z) \triangleq l(h(x), y) \ .$$

[13] In a binary classification scenario, for example, Y is the label set $\{-1, 1\}$.

[14] The stochastic output is represented by the conditional probability of a $y \in Y$ given $x \in X$, which we also denote by f to simplify notation.

We also define the expected loss of a hypothesis $h \in H$ with respect to a domain D as

$$\mathcal{L}_D(h) \triangleq E_{z\sim D}[l(h,z)] \ . \tag{6.1}$$

We also define the relative loss of non deterministic labeling functions $f_1, f_2 : X \to \Delta_Y$ with respect to an input domain D_X as follows:

$$\mathcal{L}_{D_X}(f_1, f_2) \triangleq E_{x\sim D_X} E_{(y_1,y_2)\sim(f_1(x),f_2(x))}[l(y_1,y_2)] \ . \tag{6.2}$$

This captures deterministic labeling as a special case and we can define for a hypothesis $h \in H$ the expected loss as

$$\mathcal{L}_{D_X}(h, f) \triangleq E_{x\sim D_X} E_{y\sim f(x)}[l(h(x),y)] \ . \tag{6.3}$$

As a special case, for a domain $D = (D_X, f_D)$ we have the following alternative formulation for the expected loss of a hypothesis,

$$\mathcal{L}_D(h) = \mathcal{L}_{D_X}(h, f_D) \ . \tag{6.4}$$

Note that in the special case of a deterministic relation $y = f_D(x)$ we have

$$\mathcal{L}_D(h, f_D) \triangleq E_{x\sim D_X}[l(h(x), f_D(x))] \ .$$

Also, for a hypothesis class H, we denote by h^*_D the optimal hypothesis (that is, of minimal expected loss) w.r.t a domain D:

$$h^*_D \triangleq \arg\min_{h\in H} \mathcal{L}_D(h) \ .$$

We denote by $S = \{s_i\}_{i=1..m}$ the set of m labeled samples $s_i = (x_i, y_i)$ drawn i.i.d. from the source domain Q. We denote by $T = \{t_j\}_{j=1..n}$ the set of n unlabeled examples drawn i.i.d. from P_X. Now, applying (6.1) for the uniform distribution induced by a finite sample set $S \subset Z$ we have the following *empirical* loss of a hypothesis $h \in H$,

$$\mathcal{L}_S(h) \triangleq \frac{1}{|S|} \sum_{s\in S} l(h,s).$$

Finally, as illustrated in Figure 6.1, an *adaptation learning algorithm* A uses the

labeled source-sample set S and the unlabeled target-sample set T to return a hypothesis $h_A \in H$. Since in the domain adaptation problem we are interested in $\mathcal{L}_P(h_A)$, using ERM (that is, setting $h_A = h_S^*$) is no longer justified by the PAC bound (1) that ensures with high probability that for a large enough (yet feasible) training sample S, $\mathcal{L}_Q(h_S^*)$ and $\mathcal{L}_Q(h_Q^*)$ are close.

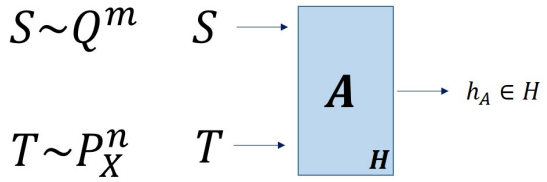


Figure 6.1: A schematic Domain Adaptation Algorithm A. Inputs are a source domain Q labeled sample set S and a target-domain P unlabeled sample set T. Output is $h_A \in H$ where H is the hypothesis class from which A chooses the output.

As a result, there is a need to relate the adaptation learning algorithm's performance $\mathcal{L}_P(h_A)$ (and specifically for ERM, $\mathcal{L}_P(h_S^*)$) to that of the optimal achievable $\mathcal{L}_P(h_P^*)$. Both quantities are inaccessible in general since the labeled samples are from a potentially unrelated domain Q. Therefore, to allow for adaptation, the relation between the sorce and target domains has to be modeled. Modeling the *discrepancy* between source and target domains and providing related generalization bounds was a key motivation in the line of work that is reviewed in the next section.

6.3 Domain Adaptation Theory and Algorithms

It is natural to expect that the ability of the learning algorithm to generalize using unlabeled target samples will depend significantly on the similarity of the source and target distributions Q_X and P_X.

The L^1 norm[15]

$$L^1(D_1, D_2) \triangleq \sup_a |D_1(a) - D_2(a)| \ , \tag{6.5}$$

also called *total variation*, is a natural choice for quantifying the discrepancy between two distributions. The following holds:

Proposition 3. *For a non-negative, symmetric, and M bounded loss function l that satisfies the triangle inequality, for all source and target domains Q and P respectively,*

[15]The supremum in the definition that follows is taken over all measurable sets a of the probability space implied by D_1 and D_2.

for all $h \in H$

$$\mathcal{L}_P(h) - \mathcal{L}_Q(h) \leq ML^1(Q_X, P_X) + \min\{\mathcal{L}_{Q_X}(f_Q, f_P), \mathcal{L}_{P_X}(f_Q, f_P)\} \tag{6.6}$$

Proof. Recall that $\mathcal{L}_P(h) = \mathcal{L}_{P_X}(h, f_P)$ and $\mathcal{L}_Q(h) = \mathcal{L}_{Q_X}(h, f_Q)$. Now,

$$\begin{aligned} \mathcal{L}_P(h) - \mathcal{L}_Q(h) &\leq |\mathcal{L}_{Q_X}(h, f_P) - \mathcal{L}_{Q_X}(h, f_Q)| + |\mathcal{L}_{P_X}(h, f_P) - \mathcal{L}_{Q_X}(h, f_P)| \\ &\leq \mathcal{L}_{Q_X}(f_P, f_Q) + ML^1(Q_X, P_X) \end{aligned}$$

By alternatively adding and subtracting $\mathcal{L}_{P_X}(h, f_Q)$ we also get

$$\mathcal{L}_P(h) - \mathcal{L}_Q(h) \leq \mathcal{L}_{P_X}(f_P, f_Q) + ML^1(Q_X, P_X)$$

and (6.6) follows by taking the lower of the two bounds. □

As a special case, for binary classification in a covariate shift setting we have the following adaptation bound for all $h \in H$

$$\mathcal{L}_P(h) - \mathcal{L}_Q(h) \leq L^1(Q_X, P_X) \ . \tag{6.7}$$

As a means to introduce notation and intuition, consider the following simple proof of the above special case (6.7) of the bound (6.6). We assume the deterministic labeling f and denote by $h\Delta f$ the domain set on which $h \in H$ and f differ. We now have

$$\mathcal{L}_P(h) - \mathcal{L}_Q(h) = E_{x \sim P_X}[1_{h(x) \neq f(x)}] - E_{x \sim Q_X}[1_{h(x) \neq f(x)}] = P_X(h\Delta f) - Q_X(h\Delta f) \leq L^1(Q_X, P_X)$$

We conclude that if P_X and Q_X are statistically indistinguishable (have a small total variation distance, i.e., a small L^1 norm distance), then simply learning with respect to the source labeled data may be (as in the above setting) a very beneficial strategy.

The bound above is, however, impractical. Although the L^1 norm has some desirable properties such as scale invariance, in many settings it can not be estimated from finite samples [21] and is therefore of limited practical value. Moreover, in the domain adaptation setting, it is easy to construct scenarios in which $L^1(P, Q)$ is significant, yet adaptation is trivial, and in general, one can get a good domain adaptation even

in cases when the two distributions are statistically very far in L^1 sense, or even have disjoint support.

To illustrate the over sensitivity of the L^1 norm, consider the following example (again, assuming covariate shift and deterministic labeling): Let the domain X be $\{\frac{i}{2m}\}_{i=0}^{2m-1}$ with Q_X uniform on $\{\frac{i}{m}\}_{i=0}^{m-1}$ and P_X uniform on $\{\frac{2i+1}{2m}\}_{i=0}^{m-1}$. Although the total variation is 1, the L^1 norm of P_X and Q_X is irrelevant for the task of adaptation for binary classification using the class of threshold functions $H = \{h_t = 1_{x \leq t} : t \in [0,1]\}$. Indeed, any two hypotheses h_{t_1} and h_{t_2} disagree in an interval I_{t_1,t_2}. The probability of such an interval represents the potential loss of a hypothesis output by a learning algorithm (h_A) compared to the best achievable using a member of H. Now, for all t_1, t_2, we have $|P_X(I_{t_1,t_2}) - Q_X(I_{t_1,t_2})| \leq \frac{1}{m}$ meaning that adaptation is easy since the change from domain Q to P does not influence our potential loss for any algorithms and any underlying labeling.

It is therefore evident that a discrepancy measure relevant to adaptation should account for the class H as well. Such a measure, merely limiting the maximization in (6.5) to relevant subsets of $A \subseteq 2^X$

$$d_A(P,Q) \triangleq \sup_{a \in A} |P(a) - Q(a)| \ , \tag{6.8}$$

was first proposed in [73] and later applied to domain adaptation for binary classification by [22] with the choice of $A = H\Delta H \triangleq \{h_1 \Delta h_2 : h_1, h_2 \in H\}$, where $h_1 \Delta h_2$ is the symmetric difference operator - the set of domain points over which h_1 and h_2 differ. The similarity measure (6.8) captures the 0-1 loss and was used to derive the following generalization bound for domain adaptation

Theorem 4. *[22] Let H be a hypothesis class of VC-dimension d, and S and T unlabeled sample sets of size m each, drawn from the source domain Q and target domain P, respectively. Then, for any $\delta \in (0,1)$, with probability at least $1-\delta$ (over the choice of the samples), for every $h \in H$:*

$$\mathcal{L}_P(h) - \mathcal{L}_Q(h) \leq d_{H\Delta H}(S,T) + \min_{h' \in H} \left\{ \mathcal{L}_P(h') + \mathcal{L}_Q(h') \right\} + 4\sqrt{\frac{2d\log(2m) + \log(\frac{2}{\delta})}{m}} \tag{6.9}$$

Note the terms bounding the adaptaion regret $\mathcal{L}_P(h) - \mathcal{L}_Q(h)$ above: The first is an empirical d_A distance, which is (with high probability) close to the source-target

discrepancy $d_{H\Delta H}(Q,P)$ (this is by combining simple applications of Hoeffding's inequality (see Theorem D.1 in [90]) to $Pr(|Q(a)-S(a)|>\epsilon)$ and $Pr(|P(a)-T(a)|>\epsilon)$) but might be infeasible to calculate in general. The second term is inaccessible but hints that the existence of a joint hypothesis of small error in both domains is a condition for adaptability (this was shown to be an insufficient condition in [23]). The last term (the residual of approximating the true discrepancy via the samples) vanishes for a fixed required confidence δ as the sample size is increased. Actually, the bound (6.9) can be stated in terms of the original discrepancy for all $h \in H$ as follows:

$$\mathcal{L}_P(h) - \mathcal{L}_Q(h) \leq d_{H\Delta H}(Q,P) + \min_{h' \in H} \left(\mathcal{L}_P(h') + \mathcal{L}_Q(h')\right) . \tag{6.10}$$

The d_A distance (6.8) was extended by [81], addressing general loss functions and allowing settings beyond binary classification, such as regression

$$disc_L(P,Q) \triangleq \max_{h_1,h_2 \in H} |\mathcal{L}_P(h_1,h_2) - \mathcal{L}_Q(h_1,h_2)| . \tag{6.11}$$

They show that for bounded loss functions this discrepancy measure may also be estimated from finite samples (this time using the sharper Rademacher Complexity bounds - see e.g., Chapter 3 of [90]). Specifically, for bounded loss functions of the form $l_q(y,y') \triangleq |y-y'|^q$, they prove (Corollary 7 in [81]) that with high probability (over the sampling S and T) the difference between $disc_{L_q}(Q,P)$ and $disc_{L_q}(S,T)$ is bounded by $4q(\hat{\mathcal{R}}_S(H) - \hat{\mathcal{R}}_T(H))$ and a term that efficiently vanishes as the size of S and T is increased. This leads to a generalization bound for domain adaptation with symmetric loss functions obeying the triangle inequality, on the difference between the target loss of a hypothesis $\mathcal{L}_P(h)$ and its source-optimality $\mathcal{L}_Q(h,h^*_Q)$:

Theorem 5. *[81] For any $h \in H$,*

$$\mathcal{L}_P(h) - \mathcal{L}_Q(h,h^*_Q) \leq disc_L(P,Q) + \mathcal{L}_P(h^*_P) + \mathcal{L}_Q(h^*_P,h^*_Q) \tag{6.12}$$

Note that for the deterministic consistent case in a covariate shift setting (i.e., $P(y|x) = Q(y|x) = 1_{y=f(x)}$ and $f \in H$), the left hand side of both bounds (6.10) and (6.12) is $\mathcal{L}_P(h) - \mathcal{L}_Q(h)$ and the only term remaining in the right hand side is the source-target domain's discrepancy. This reinforces the usage of the discrepancy as a domain-adaptability measure.

The nature of the generalization bounds (6.10) and (6.12) is that they relate the expected error on the target domain to an observable quantity (related to the error) on the source domain. From this perspective they should be viewed more as studying what guarantee we can give, when we learn with respect to the source domain and later are tested with respect to the target domain, rather than giving constructive algorithmic tools (with the exception of [33], which uses a proxy of the d_A-distance as a criterion for the applicability of domain adaptation). Nevertheless, such bounds may serve as theoretical justification for domain adaptation algorithms that aim to minimize the generalization bounds. Indeed, reweighing algorithms that optimize the discrepancy distance were presented in [81] and subsequently, for regression in [45]. However, in general, minimization of discrepancy distance does not insure an improved performance, since the reweighing might result in overfitting (see e.g., example 8 in[23]). Moreover, [23] show (although not constructively, using the probabilistic method) that for binary classification, small $d_{H\Delta H}$ discrepancy (even in covariate shift settings) is not sufficient to allow for domain adaptation. This implies that for domain adaptation to be possible, both terms in the right hand side of (6.10) should be small. Therefore, re-weighting Q_P to reduce the discrepancy $d_{H\Delta H}(Q_P, P)$ might result in overfittiong, that is, increasing the other term $\min_{h' \in H} \mathcal{L}_P(h') + \mathcal{L}_{Q_P}(h')$ which is not accessible.

Domain Adaptation (DA) algorithms that utilize unlabeled target data were first presented by [33] (that is, beyond importance-sampling re-weighting methods based on *Sample Selection Bias* algorithms such as those presented by [30, 67, 115][16]). Their method uses unlabeled samples of both domains to identify pivot features (frequently occurring in both domains) and re-weight input-space features based on their correlation with the pivot features. Other feature-based DA algorithms (where a new feature space is sought to better represent the inter domain similarity) based on metric learning were presented in [75, 106]. Similar to [33], the work in [88] iteratively updates a predictor, based on re-weighted features. The predictor is updated by gradually labeling target-domain instances (thereby adding them to the train set) based on their confidence level with regard to the current predictor. Two other conceptually similar such *self-labeling* iterative algorithms are *DASVM* [36], which gradually labels and adds to the training set target samples (and removes source samples) based on the sample's margin with respect to the iteratively learned classifier, and *SLDAB* [63], an

[16] As mentioned before, such reweighting methods are instable in general.

AdaBoost variant that simultaneously maintains per-iteration distributions over the source and (iteratively self labeled) target sets while maintaining a low discrepancy distance between the distributions. The domain adaptation algorithm presented in [24] also recovers the labels of the target data set, this time using a nearest neighbor method (based on the labeled source-domain data). A proper learning algorithm then results by applying any standard learning method using the now-labeled target training set.

The method of [24] for domain adaptation relies, however, on the following two assumptions (in addition to covariate shift) regarding the labeling function and difference between the source and target distributions[17]: First, the labeling function is assumed to be of low variation with high probability[18] (this is related to algorithmic robustness, as detailed next in Chapter 7). Second, a lower bound is assumed on the ratio of source and target probabilities $\inf_{P_X(a)\neq 0} \frac{Q_X(a)}{P_X(a)}$ over relevant subset a of the domain[19]. In [25] it is shown that for settings in which those assumptions hold, a nearest neighbor algorithm that only relies on source-domain samples will have arbitrary small error[20] (although requiring a number of samples that is exponential in the dimension of the domain), and that proper learning (that is, when the algorithm is required to return a classifier h from a predefined class H) is impossible in general without utilizing target domain samples.

The impossibility results of [25] however, only apply to binary classification. Indeed, for the case of regression, a bound on the pointwise loss of the hypothesis returned by a kernel-based regularization algorithm is provided in [45]. The bound includes a term that depends on global properties of the loss function (again, lipschitzness is assumed) and another that depends on the discrepancy distance $disc_L(S,T)$ as defined in (6.11), motivating the approach of reweighting S such that $S_X^w = \arg\inf_D disc_L(D,T)$ and then learning using the labled sample set S^w. Note that minimizing the discrepancy is intractable in general. However, an efficient method for the case of bounded linear functions (H being class of linear operators $h_w(x) = w^t x$ where $\|w\| \leq \Lambda$) is presented in [45]. See [85] for a systematic review of domain adaptation methods and the usage

[17] The authors also prove that both assumptions are necessary. That is, domain adaptation is impossible if any of the two assumptions is relaxed while the other remains.

[18] This is denoted *Probabilistic Lipschitzness* and implies that with high probability (over the selection of a domain sample x) the labeling function is $\lambda-$Lipschitz, that is, for all y, $|l(x) - l(y)| \leq \lambda\|x - y\|$

[19] This may be regarded as bounding a multiplicative form of the L^1 norm in general, or the discrepancy distance 6.8

[20] This is intuitive - if the labeling does not change much in the vicinity of target sampled points, and if (due to the lower bound on the ratio) the vicinity of all target points is eventually sampled, then a nearest neighbor labeling will be correct with high probability.

of unlabeled target domain samples.

The *Semi Supervised Learning* (SSL) setting in which abundance of unlabeled data is available in conjunction with few labeled samples (although of the same domain[21]) is of special interest due to its resemblance to the domain adaptation setting. SSL algorithms usually assume low probability in the vicinity of the underlying decision boundary (or informally, that points in a high density cluster are similarly labeled, denoted the *clustering assumption*). SSL algorithms exploit this property and implement schemes to propagate the labels from S to T (see e.g., [127]). Similar methods are applied for domain adaptation for settings in which the the clustering assumption holds (see [124]).

The domain adaptation method [84] to be presented later in Chapter 8 may be conceptually regarded as based on a clustering assumption, although the labeling continuity property is defined differently, using the *algorithmic robustness* framework. Furthermore, the domain adaptation algorithm optimizes the source-loss subject to constraints that relate the source and target domain distributions, but contrary to re-weighting or iterative self-labeling methods that directly aim at making the distributions similar, the algorithm takes a *robust optimization* approach and optimizes subject to constraints that represent a worst case target-labeling given the source and target unlabeled samples. It turns out that the resulting algorithm can be interpreted as performing a re-weighting of the source samples (although indirectly). The essence of algorithmic robustness and robust optimization is presented next.

[21] Not to be confused with Semi Supervised Domain Adaptation in which few *labeled* samples from the target domain are also available.

Chapter 7

Robust Optimization and Algorithmic Robustness

This background chapter introduces the concepts of Robust Optimization and Algorithmic Robustness, both essential for the derivation of the main result to be presented in Chapter 8.

7.1 Robuts Optimization

Many settings in science and engineering may be formulated as optimization problems. That is, in a very abstract form, given a set of predefined knowledge (this may be due, e.g., to observations or modeling assumptions), find among a related set of alternatives one that best fits the predefined knowledge. In mathematical terms, such an optimization problem is specified as

$$\arg \min_{h \in H(S)} f(h, S) \ , \tag{7.1}$$

where S represents the predefined knowledge, and $f(\cdot, \cdot)$ is a function that scores the (miss)fit of a potential solution $h \in H(S)$ to the a-priori knowledge S.

Specifically, fitting a regression line using the least squares method is an instance of the above setting in which S is the set of sampled points $\{(x_i, y_i)\}_{i=1}^n$ and we are looking for the linear mapping characterized by w and b that minimizes the empirical sum of distances $f((w, b), S) = \sum_{i=1}^n (y_i - wx_i - b)^2$. Another ubiquitous instance is linear programming, where the objective is to find $\arg\min_x c^t x$ subject to linear constraints

$Ax \geq b$ and $x \geq 0$ (This standard form of a minimization linear program is equivalent to a maximization linear program in the following standard form: find $\arg\max_x c^t x$ subject to linear constraints $Ax \leq b$ and $x \geq 0$). Here, the vectors c and b and the matrix A plays the role of the a-priori knowledge S (e.g., in a production optimization setting, c may pertain to assumed revenue per unit sold of each product type, b to available resources, and A to the required amount of each resource to produce each product type), and the feasible set of the constraints is H.[1]

Many Machine Learning settings and algorithms can be formulated as (7.1), which can be efficiently solved in many cases, especially for convex loss function.[2] SVM (See Section 1.1) for example looks for the linear separator w of bounded norm B that minimizes the empirical loss $\frac{1}{n}\sum_{i=1}^{n} l(w^t x_i, y_i)$, where the loss function is the hinge loss $l(r, y) = \max\{0, 1 - ry\}$. From a more general perspective, many machine learning algorithms aim for the optimal member $h \in H$ minimizing the average average loss

$$\arg\min_{h \in H} E_Z[f(h, z)] \ , \tag{7.2}$$

where Z is the (unknown) probability distribution generating some future random *test* instance. Since Z is unknown, it is accessed through a *training* set $S = \{z_i = (x_i, y_i)\}_{i=1}^{n}$ and the resulting optimization problem of minimizing the empirical loss becomes (7.1).

This statistical casting of Machine Learning suggest that the optimization may be also done iteratively, through *Stochastic Gradient Descent* methods, where the value of the optimal h is updated by considering one sample at every iteration. We may refer to such *Stochastic Optimization* methods as addressing the uncertainty in the true target to be optimized $E_Z[f(h, z)]$ (i.e., the uncertainty in Z) by optimizing a related objective (7.1) and relying on probabilistic assumptions regarding the relation of S and Z to ensure consistency. Therefore, stochastic optimization methods for machine learning address uncertainty using probabilistic assumptions and immunize the learning algorithm from failure in a probabilistic sense (and this is the nature of PAC learning bounds (1). Uncertainty, however, might result in instability. Indeed, stochastic optimization algorithms can be very sensitive to perturbations in the sampled data (due, e.g., to noise) or to modeling deviation due to uncertainties in the parameters

[1] Note the dependence of H in S in this case.

[2] The ability to efficiently solve such optimization problems depends of course also on the structure of the feasible set.

S of (7.1), sometimes even rendering the algorithms useless [26].

As opposed to stochastic optimization, the *Robust Optimization* approach to uncertainty in the parameters of (7.2) and (7.1) is deterministic in nature and the learning algorithm is immunized against failure for *every* realization (hence the name, robust) of the uncertain parameters as long as they remain in a specified set Δ:

$$\arg \min_{h \in H(S)} \max_{S \in \Delta} f(h, S) \ , \tag{7.3}$$

thereby providing *worst case* guarantees. Therefore, the robust variant of empirical risk minimization has the form

$$\arg \min_{h \in H} \max_{\delta_i \in \Delta} \frac{1}{n} \sum_{i=1}^{n} l(h, z_i + \delta_i) \ , \tag{7.4}$$

adversarially protecting the learning algorithm from any sampling accuracy deviations of at most Δ.

A robust versions of SVM for the case of uncertainty in the attributes x_i, for example, is formulated as follows:

$$\min_{w,b,\xi} C \sum_{i=1}^{n} \xi_i + \frac{1}{2} \|w\|^2 \tag{7.5}$$

subject to

$$y_i(w^T x_i + b) \geq 1 - \xi_i + \|\Sigma_i^{\frac{1}{2}} w\|_2 \qquad \forall i = 1..n \tag{7.6}$$

$$\xi_i \geq 0 \qquad \forall i = 1..n \tag{7.7}$$

This regularization of the constraints pertains to an additive uncertainty model $x_i^{\text{true}} = x_i + u_i$, where the uncertainty set U_i of u_i is an ellipsoid characterized by Σ_i as follows: $U_i = \{u : u^t \Sigma_i u \leq 1\}$. Alternatively, when the uncertainty in the attributes is coupled, reflected in a global[3] uncertainty set $\Delta = \{(u_1, \ldots, u_n) : \sum_{i=1}^{n} \|u_i\|^2 \leq C\}$, then (as shown in [125]) the related robust optimization program

$$\min_{w,b} \max_{u \in \Delta} \sum_{i=1}^{n} \max\{1 - y_i[w^t(x_i - u_i) + b], 0\}$$

[3] As opposed to the *local* uncertainty of each attribute x_i separately, as expressed in the constraints 7.6

is equivalent to the regularized SVM formulation

$$\min_{w,b} C\|w\| + \sum_{i=1}^{n} \max\{1 - y_i[w^t x_i + b], 0\}.$$

This connection between robust optimization and regularization justifies the notion of robustness that is usually associated to the usage of regularization in machine learning algorithms. It also shows the flexibility of robust optimization methods, that may be used to capture non-adversarial settings as well (due to the assumed coupling of location deviations, which are reminiscent of stochastic uncertainty), yielding as a special case the familiar stochastic optimization formulation. Efficient convex optimization algorithms exist to solve the above formulations (and other robust versions of SVM, modeling other sources of uncertainty). See [125] for more details.

The Robust Optimization approach to Domain Adaptation, as introduced in our Robust Domain Adaptation algorithm [84], considers the source-target domain discrepancy as if it was parameter uncertainty in the Robust Optimization terminology. Conceptually, we suggest a template method for converting a stochastic optimization-based machine learning algorithm (7.2)[4] to a domain adaptation algorithm (7.3) by adding uncertainty related constraints for S that reflect the source-target domain discrepancy, and the additional optimization step over worst-case realizations of S.

Specifically, since only unlabeled samples are available in the target domain, the uncertainty is regarding the statistical properties of the target lables (e.g., the probability of the label of a sample being positive, given its unlabeled attributes). Since such statistical properties are available for the source-domain training set, we may quantify the allowable change between source and target through some distance measure and define the constraining uncertainty set in the resulting robust optimization program to include only target labellings that are within a predefined distance budget. This is the purpose of the λ-shift measure of distribution disengagement to be introduced in Chapter 8 and used to formulate our robust domain adaptation SVM. See [29] for more on different notions and definitions of *budget of uncertainty*, how they are used to choose the uncertainty sets of robust optimization programs, and related solution methods.

[4] Which actually requires to solve (7.1), where the target function f is the empirical risk.

7.2 Algorithmic Robustness

Robust Optimization, as discussed, addresses uncertainty in the input parameters to an optimization problem and ensures optimality of the resulting optimum in a worst-case deterministic sense. From a more general algorithmic perspective, *Algorithmic Robustness* was introduced by [126] as a measure of the sensitivity of a learning algorithm to changes in its training data. Conceptually, a robust algorithm will output a hypothesis that has limited variation in the vicinity of the training samples, thereby (intuitively) precluding over-fitting.

Before presenting the precise definition of algorithmic robustness, consider a regression algorithm that tries to fit a polynomial to a set of samples as a motivating setting that illustrates the danger of high variation in the vicinity of training samples. A sample set of any size may be perfectly fit by a polynomial, however of large degree and hence of high variation. Such a high degree polynomial will perform poorly when the actual model is of low dimension or if the training data was somewhat inaccurate. Motivated by the connection between robust optimization and regularization as presented in the previous section, we may use regularization to induce the regression learning algorithm to prefer polynomials of low degree and achieve algorithms of better generalization capabilities. Similarly, the notion of algorithmic robustness is related to the ability of an algorithm to generalize. Indeed, a similar connection (actually equivalence) between algorithmic robustness and generalization ability is established in [126].

Specifically, the algorithmic robustness of a learning algorithm is characterized by a parameter K and a real function $\epsilon(S)$ mapping a sample set S as follows:

Definition 6. *[126] Algorithm A is $(K, \epsilon(\cdot))$-robust if the input-output space Z can be partitioned to K disjoint sets $\{C_k\}_{i=1}^{K}$ such that $\forall S$, $\forall s \in S$, $\forall z$ such that $s, z \in C_k$,*

$$|l(h_S, s) - l(h_S, z)| \leq \epsilon(S),$$

An algorithm A is $(K, \epsilon(S))$ robust if there is a partition of the input-label space $Z = X \times Y$ to K subsets such that the loss of the learned hypothesis h_S [5] has $\epsilon(S)$-bounded variation in every region of the partition that contains a training sample. Note that K in the above definition is the number of regions the input-output space Z has to be partitioned. Note also that K and the regions do not depend on the sample set S

[5] recall that we denote by h_S the output hypothesis of A given as input the training set S.

and should uniformly apply to any. Consequently, a K robustness level of an algorithm induces a partition of the input-output space to multiple regions, and in each region the hypothesis of the robust algorithm has limited variation in its loss. For partitions that are a Cartesian product of input and output space partitions we have that for any training set S the algorithm outputs a hypothesis that has $\epsilon(S)$ variation in any region of any sample. The robustness of popular learning algorithms (such as SVM) was established in [126].

Intuitively, the above definition states that for a $(K, \epsilon(\cdot))$-robust algorithm, the loss of its output hypothesis h_S has $\epsilon(S)$ variation within each region C_k and as a result the empirical error (on samples $s \in S$) of h_S is a good approximation for the expected error of h_S. Therefore, a robust algorithm that minimizes empirical error is expected to generalize well. Indeed, [126] prove this precise result, and bound the difference between the empirical error and the expected error of $(K, \epsilon(\cdot))$-robust algorithms:

Theorem 7. *[126] If A is a $(K, \epsilon(S))$-robust algorithm then for any $\delta > 0$, with probability at least $1 - \delta$,*

$$|\mathcal{L}_Q(h_S) - \mathcal{L}_S(h_S)| \leq \epsilon(S) + M\sqrt{\frac{2K \ln 2 + 2 \ln \frac{1}{\delta}}{|S|}} \ ,$$

where M is an upper bound on the underlying loss function. Note the dependence of ϵ on the sample set S[6]. Indeed, the SVM algorithm, explored later, is (K, ϵ)-robust, where ϵ does not depend on the actual training set S but only on its size m.

The most important property of robustness that we utilize for domain adaptation is the limited variation in the loss of the output hypothesis within each region. This implies that the output hypothesis of a robust algorithm (and ours in particular) would have a limited variation within each region (where in our case, the regions of the input-output space each have a constant output-space value, e.g., a constant label[7]).

Now, since the overall expected loss of a candidate hypothesis is an average of the losses in each region, bounding the loss in a region is our main tool to derive generalization bounds. The main difficulty that we encounter is that the regions guaranteed by the robustness depend on the label which is not observable for the target distribution sample. We use the Robust Optimization approach to address this difficulty

[6]The parameter ϵ may also depend on K, and [126] provide a uniform bound for all K.

[7]We assume partitions of Z that are a Cartesian product of input space regions and the set of output labels.

by bounding the target-domain expected loss in a region of a candidate hypothesis as a function of its (accessible!) source-domain expected loss. This is possible due to the limited variation of the output label of any candidate hypothesis in a region, and the assumption of constrained target-domain label distribution discrepancy in a region. Furthermore, the resulting domain adaptation generalization bound naturally leads to a regularized domain adaptation algorithm that minimizes the generalization bound, as detailed in the next chapter.

Chapter 8

Robust Domain Adaptation

We derive a generalization bound for domain adaptation by using the properties of robust algorithms. Our new bound depends on λ-shift, a measure of prior knowledge regarding the similarity of source and target domain distributions. Based on the generalization bound, we design an SVM variants for binary classification and regression domain adaptation algorithms. Our Robust Domain Adaptation SVM algorithm is Algorithmically Robust *by design*.

From the algorithmic perspective we develop SVM variants for binary classification and regression domain adaptation algorithms. The algorithms are formulated as convex optimization programs where the optimized term is based on the generalization bound and the constraints are set to match the λ-shift level assumed. Specifically, the optimized term includes a weighted average (by the target domain distribution) of the bound on the loss in each region, and the constraints on the primal variables (the losses in each region) are the worst case average errors in the regions given the source domain empirical errors and the assumed λ-shift level. Finally, we use the dual representation of the convex optimization program to offer a reweighing interpretation of the resulting robust domain adaptation algorithms.

Two interesting extreme cases are the pessimistic case (assuming that there is no relationship between the probability over labels across the source and target distributions) and the optimistic case (assuming that the probability over labels in the source and target distributions is identical, for any region of the input domain). This will lead us in the former case to pessimistic bounds, where we will use the worse case loss (over the labels) for each region, and to optimistic bounds in the latter.

We use the λ-shift and the Robust Optimization approach to address the absense of

target-domain labels: the λ-shift serves as a prior assumption regarding the similarity of the source and target domains. The λ-shift also serves to relate the sampled empirical distribution of output domain values (e.g., the conditional probability of each label value in a given input-domain region) to a set of such distributions over label values in the target domain. We subsequently apply robust optimization to come up with the hypothesis that achieves the best loss over the worst-case possible target-domain distributions over labels. In that respect, our approach may be viewed as utilizing the domain uncertainty (captured by the λ-shift property) as the training sample uncertainty of the Robust Optimization method. We also derive related parameterized generalization bounds for domain adaptation.

8.1 Model

Using the terminology and notation presented in section 6.2.4, let X be the input space and $f\colon X \to Y$ be the unknown target function[1] (where the label set Y is $\{-1, 1\}$ in case of binary classification and a finite set $\{y_1, \ldots y_r\}$ otherwise). The input-output space is $Z = X \times Y$, and H is the hypothesis class used by a learning algorithm to learn f.

A learning algorithm A has access to a labeled sample set S sampled from the source domain Q, and an unlabeled set T sampled from P_X, where P is the target domain. The algorithm is expected to return a hypothesis $h_{S,T} \in H$ of low target loss $\mathcal{L}_P(h_{S,T})$. Recall that an algorithm A is $(K, \epsilon(S))$ robust if there is a partition of Z to K subsets such that the loss of the learned hypothesis has $\epsilon(S)$-bounded variation in every region C of the partition that contains a training sample. In what follows, we assume (by design) that the associated partition of Z to K regions is of the following form: $Z = \cup_{i,j} X_i \times Y_j$, where the input space and output space partitions are $X = \cup_{i=1}^{K_x} X_i$, $Y = \cup_{j=1}^{K_y} Y_j$, and $K = K_x K_y$. This partition implies that the output hypothesis of a (K, ϵ)-robust algorithm has at most an ϵ variation in the loss in each region of constant label $C_k = X_i \times Y_j$.

[1] As noted in section 6.2.4, f may alternatively represent the conditional probability $P(Y|X)$ in case of stochastic labeling.

8.1.1 λ-shift

Our main goal is to use the notion of robustness to overcome key difficulties in the domain adaptation setting. The most important difficultly is that we would like to learn with respect to a distribution P, from which we have only unlabeled samples, while the labeled samples are given with respect to a distribution Q.

The notion of robustness would guarantee that in every region $X_i \times Y_j$ the loss of the algorithm would be similar, up to ϵ, regardless of the distribution (source, or target) inside the region. However, a main difficulty still remains since the regions depend on the (unavailable) label of the target function. Therefore, our strategy is to consider the conditional distribution of the label in a given region X_i and the relation to its sampled value over the given labeled sample S. For a distribution σ over $Y = \{y_1, \dots y_r\}$ (where $r = K_y$, the number of output labels) we denote the probability of every label y_v by σ^v. We start with a definition of the λ-shift of a given distribution $\sigma \in \Delta_Y$:

Definition 8. *Distribution $\rho \in \Delta_Y$ is λ-shift w.r.t. to $\sigma \in \Delta_Y$, denoted $\rho \in \lambda(\sigma)$, if for all $y_v \in Y$ we have $\rho^v \leq \sigma^v + \lambda(1 - \sigma^v)$ and $\rho^v \geq \sigma^v(1 - \lambda)$. If for some v we have $\rho^v = \sigma^v + \lambda(1 - \sigma^v)$ we say that ρ is strict-λ-shift w.r.t to σ*

A λ-shift therefore restricts the change of the probability of a label - the shift may be at most a λ portion of the probability of the other labels (in case of increase) or of the probability of the label (in case of decrease). To simplify notation, for $\rho \in \lambda(\sigma)$ we denote the upper bound of the probability ρ^v of a label y_v by $\overline{\lambda}^v(\sigma) \triangleq \sigma^v + \lambda(1 - \sigma^v)$, and the lower bound on ρ^v by $\underline{\lambda}^v(\sigma) \triangleq \sigma^v(1 - \lambda)$.

For a non-negative function $l : Y \to \mathbb{R}_+$ we now consider its maximal possible average $E_\rho(l) \triangleq \sum_{y_v} \rho^v l(y_v)$ as a result of a λ-shift:

Definition 9.

$$E_{\lambda,\sigma}(l) \triangleq \max_{\rho \in \lambda(\sigma)} E_\rho(l) \ .$$

Since the maximum is achieved when ρ is strict-λ-shift to the label y_v of maximal value of l, we have the following:

$$E_{\lambda,\sigma}(l) = \max_v \{l(y_v)\overline{\lambda}^v(\sigma) + \sum_{v' \neq v} l(y_{v'})\underline{\lambda}^{v'}(\sigma)\} \ .$$

Note that for the special case of no restriction (i.e., $1-$shift) we have $E_{1,\sigma}(l) =$

$\max_j\{l(y_j)\}$ and for the special case of total restriction (i.e., 0−shift) we have $E_{0,\sigma}(l) = E_\sigma(l)$.

To apply the above definitions to the domain adaptation problem first note that the labeled sample S induces in every region X_i a distribution σ_i on the labels: $\sigma_i^v \triangleq \frac{|S_{i,v}|}{|S_i|}$, where $|S_{i,v}|$ is the number of samples labeled y_v in region X_i and $|S_i|$ is the total number of samples in region X_i. Now, we say that the target distribution P is λ-shift of the source distribution Q w.r.t. a partition of the input space X, if in every region X_i the conditional target distribution on the labels $P(y|x \in X_i)$ is λ-shift w.r.t. the conditional source distribution on the labels $Q(y|x \in X_i)$.

We define for each region X_i a function that given a hypothesis h maps every possible label y_v to its maximal sampled empirical loss:

Definition 10.

$$l_i(h, y_v) \triangleq \begin{cases} \max_{s \in S \cap X_i \times y_v} l(h, s) & \text{if } S \cap X_i \times y_v \neq \emptyset \\ M & \text{otherwise} \end{cases}$$

Now, for a fixed h, viewing $l_i(h, y)$ as a function of the label y (denote $l_i(h, y_v)$ by l_i^v) and restricting the target distribution in each region X_i to be λ-shift of the empirical σ_i we get that the average loss in region X_i is bounded by $E_{\lambda,\sigma_i}(l_i)$. Specifically, we bound the maximal average loss of a hypothesis h under the λ-shift assumption in region X_i, denoted $l_S^\lambda(h, X_i)$, by

$$l_S^\lambda(h, X_i) \leq E_{\lambda,\sigma_i}(l_i) = \max_v \{l_i^v \overline{\lambda}^v(\sigma_i) + \sum_{v' \neq v} l_i^{v'} \underline{\lambda}^{v'}(\sigma_i)\} \ . \tag{8.1}$$

Note that a distribution P can be a 0-shift of Q, even if they have disjoint support. What will be important for us is that due to the robustness the loss of the algorithm in any region $X_i \times Y_v$ will be almost the same. Therefore, the major issue would be how to weigh the losses w.r.t the different labels. The λ-shift captures this issue very nicely. Assuming $\lambda = 1$ may be interpreted as a pessimistic assumption, where there is no restriction on the weights of the labels. Assuming $\lambda = 0$ represents an optimistic assumption for which in every region X_i the target distribution assigns the same probability to the samples as the source distribution. In general a $\lambda \in (0, 1)$ represent a trade-off between the two extremes.

8.2 Adaptation Bounds using Robustness

We now prove the following generalization bound for $\mathcal{L}_P(h_{S_T})$, where $h_{S,T}$ is the output hypothesis of a (K,ϵ)-robust learning algorithm A which is given a set of labeled samples S and a set of unlabeled samples T of size n.

Theorem 11. *For a (K,ϵ)-robust algorithm A and the related partition of $Z = X \times Y$, if P is λ-shift of Q w.r.t. the partition of X then $\forall\delta > 0$, with probability at least $1-\delta$, $\forall h \in H$:*

$$\mathcal{L}_P(h) \leq \epsilon + M\sqrt{\frac{2K\ln 2 + 2\ln\frac{1}{\delta}}{n}} + \sum_{i=1}^{K_x} T(X_i)\ l_S^{\lambda}(h, X_i) \tag{8.2}$$

Proof. The loss of h w.r.t. P is,

$$\mathcal{L}_P(h) = \sum_{k=1}^{K}(P(C_k) - T(C_k))\mathcal{L}_{P|C_k}(h) + \sum_{i=1}^{K} T(C_k)\ \mathcal{L}_{P|C_k}(h)\ .$$

Now, for the second sum above we have

$$\begin{aligned}\sum_{i=1}^{K} T(C_k) l_{P|C_k}(h) &= \sum_{i=1}^{K_x}\sum_{j=1}^{K_y} T(X_i \times Y_j)\mathcal{L}_{P|X_i\times Y_j}(h) \\ &= \sum_{i=1}^{K_x} T(X_i)\sum_{j=1}^{K_y} T(Y_j|X_i)\mathcal{L}_{P|X_i\times Y_j}(h)\ .\end{aligned}$$

By the robustness property, the loss of h in any region $X_i \times Y_j$ is at most ϵ away from the sampled loss at that region, so we may replace $\mathcal{L}_{P|X_i\times Y_j}(h)$ above with $\mathcal{L}_{T|X_i\times Y_j}(h) + \epsilon$. Also, since P is λ-shift of Q w.r.t. the given partition of X, in every region X_i we have that with probability at least $1-\delta$ the empirical target sample T is $(\lambda+\epsilon)$-shift of the empirical source sample S (for a sample size that depends polynomially on $\frac{1}{\epsilon}$ and $\log\frac{1}{\delta}$). We therefore get

$$\sum_{i=1}^{K} T(C_k)\ l_{P|C_k}(h) \leq \sum_{i=1}^{K_x} T(X_i)\ l_S^{\lambda}(h, X_i) + \epsilon\ .$$

Finally, from the bounded loss property we have $\mathcal{L}_{P|C_k}(h) \leq M$. Furthermore, as T is sampled from P, by the Bretagnolle-Huber-Carol inequality (as in the proof of Theorem

3 in [126]) we have that with probability $> 1 - \delta$,

$$\sum_{i=1}^{K} |P(C_k) - T(C_k)| \leq \sqrt{\frac{2K \ln 2 + 2 \ln \frac{1}{\delta}}{n}},$$

which completes the proof. □

Note that although the target sample probability $T(X_i \times y_j)$ of a label y_j in a region X_i is not available, given the hypothesis h and the partition $\{X_i\}_{i=1}^{K_x}$, the last term of the bound $\sum_{i=1}^{K_x} T(X_i) l_S^{\lambda}(h, X_i)$ can be evaluated from the sample sets S and T.

8.3 Robust Domain Adaptation SVM for Classification

We consider the classification problem for which the label set $Y = \{1, -1\}$. Robustness of SVM implies the existence of a partition $X = \cup_{i=1}^{K} X_i$ for which Eq. (8.2) holds (see [126]). Given the labeled sample set S and the unlabeled set T, our algorithms selects a hyperplane $h \in H$ that minimizes the generalization bound with an additional appropriate regularization term. We present a robust adaptation algorithm, a general scheme for the λ-shift case in which we assume that the target distribution is λ-shift of the source distribution w.r.t. the partition of X. We then consider two special cases: an *optimistic* variation in which we assume that in every region X_i the probability of each label is the same in the source and target distributions (i.e., 0-shift), and a *pessimistic* variation in which no relationship is assumed between the probability of the labels in the source and target distributions (i.e., 1-shift). We also use the notation $T_i = T(X_i)$ for the T-sampled probability of region X_i.

To simplify notation we set $S_i^+ = S_{i,1}$, $S_i^- = S_{i,-1}$, and $\sigma_i = \sigma_i^1$ the empirical probability of label 1 in region X_i. Using the notation $l_i^+ = l_i(h, 1)$ and $l_i^- = l_i(h, -1)$, the bound of Eq. (8.1) on $l_S^{\lambda}(h, X_i)$ for the general case is

$$\max\{l_i^+(\sigma_i + \lambda(1 - \sigma_i)) + l_i^-(1 - \sigma_i)(1 - \lambda), l_i^+\sigma_i(1 - \lambda) + l_i^-((1 - \sigma_i) + \lambda\sigma_i)\}.$$

This bound further simplifies to $l_i^+\sigma_i + l_i^-(1 - \sigma_i)$ for the optimistic case and to $\max\{l_i^+, l_i^-\}$ for the pessimistic case. Note that robustness of SVM implies that $l(h, s)$ varies at most ϵ over $s \in S_i^+$ (and similarly over $s \in S_i^-$). For SVM we use the hinge loss, $l(h, (x, y)) \triangleq \max\{0, 1 - yh(x)\}$. For a separating hyperplane $h_{w,b}(x) = w^T x + b$

we have $l(h_{w,b},(x,y)) = \max\{0, 1 - y(w^T x + b)\}$.

8.3.1 λ-shift SVM Adaptation

We assume that for some given $\lambda \in [0,1]$, the target distribution P is λ-shift of the source distribution Q w.r.t the partition of the domain X. We define a quadratic optimization program that finds the best separating hyperplane $h_{w,b}(x) = w^T x + b$ in the sense that the related set of losses l_i (the primal variables, together with w and b) minimizes the worst case bound of Eq. (11)[2]. In addition to the usual SVM constraints on the losses $l_i \geq l(h_{w,b}, s)$ for each sample $s \in S$ (where $l(\cdot,\cdot)$ is the hinge loss), we want to constrain the losses to satisfy $l_i \geq l_S^\lambda(h, X_i)$ for each region X_i (thereby minimizing l_i implies that we minimize $l_S^\lambda(h, X_i)$). We achieve the latter condition by using a lower bound on l_i which upper bounds $l_S^\lambda(h, X_i)$. Using a trade-off parameter C results in the following convex quadratic program which receives as input $2K$ source domain sample clusters and respective target-domain region probabilities (S_i^+, S_i^+, and T_i, for $i = 1..K$):

$$\min_{w,b,l_1,l_2,\ldots,l_K} C \sum_{i=1}^{K} T_i l_i + \frac{1}{2}\|w\|^2 \tag{8.3}$$

subject to

$$l_i^+ \geq 1 - (w^T x_j + b) \qquad \forall j = 1..m \text{ s.t. } (x_j, 1) \in S_i^+ \tag{8.4}$$

$$l_i^- \geq 1 + (w^T x_j + b) \qquad \forall j = 1..m \text{ s.t. } (x_j, -1) \in S_i^- \tag{8.5}$$

$$l_i \geq l_i^+(\sigma_i + \lambda(1 - \sigma_i)) + l_i^-(1 - \sigma_i)(1 - \lambda) \qquad i = 1..K \tag{8.6}$$

$$l_i \geq l_i^+ \sigma_i (1 - \lambda) + l_i^-((1 - \sigma_i) + \lambda\sigma_i) \qquad i = 1..K \tag{8.7}$$

$$l_i \geq 0, \quad l_i^+ \geq 0, \quad l_i^- \geq 0 \qquad i = 1..K \tag{8.8}$$

For each sample $(x_j, y_j) \in S, \quad j = 1 \ldots m$, we have a constraint (8.4) or (8.5) regarding one of the two primal variables l_i^+ or l_i^- (depending on the value of y_j) where i is the index of the region X_i to which x_j belongs. The other constraints (8.6), (8.7), and (8.8), bound the loss l_i using the λ-shift assumption.

To find the dual representation of this problem we introduce the dual variables

[2] Actually, it minimizes the last term of Eq. (11), which is the only part of the bound that depends on the hypothesis h

$\alpha_1, \ldots, \alpha_m$ pertaining to constraints (8.4) or (8.5), $\beta_1^+, \ldots, \beta_K^+$, and $\beta_1^-, \ldots, \beta_K^-$, pertaining to constraints (8.6) and (8.7) respectively, and $r_1 \ldots, r_K$, $s_1^+, \ldots, s_K^+$, and $s_1^-, \ldots, s_K^-$ pertaining to the primal variables in (8.8) respectively.

The Lagrangian is,

$$\begin{aligned} L(w, b, l, l^-, l^+, \quad \alpha \quad , \beta^+, \beta^-, r, s^+, s^-) = \\ C \quad \sum_{i=1}^{K} T_i l_i + \frac{1}{2}\|w\|^2 + \sum_{(x_j,1)\in S_i^+} \alpha_j (1 - (w^T x_j + b) - l_i^+) \\ + \quad \sum_{(x_j,-1)\in S_i^-} \alpha_j (1 + (w^T x_j + b) - l_i^-) \\ + \quad \sum_{i=1}^{K} \beta_i^+ (l_i^+ (\sigma_i + \lambda(1-\sigma_i)) + l_i^- ((1-\sigma_i) - \lambda(1-\sigma_i)) - l_i) \\ + \quad \sum_{i=1}^{K} \beta_i^- (l_i^+ (\sigma_i - \lambda\sigma_i) + l_i^- ((1-\sigma_i) + \lambda\sigma_i) - l_i) \\ - \quad \sum_{i=1}^{K} r_i l_i - \sum_{i=1}^{K} s_i^+ l_i^+ - \sum_{i=1}^{K} s_i^- l_i^- \end{aligned}$$

Applying the KKT conditions, at the optimal primal values the partial derivatives of the Lagrangian w.r.t. the primal variables w, b, l, l^+, l^- are 0, and we get respectively:

$$w - \sum_{(x_j,1)\in S_i^+} \alpha_j x_j - \sum_{(x_j,-1)\in S_i^-} \alpha_j x_j = 0 \tag{8.9}$$

$$\sum_{(x_j,1)\in S_i^+} \alpha_j - \sum_{(x_j,-1)\in S_i^-} \alpha_j = 0 \tag{8.10}$$

$$CT_i - r_i - \beta_i^+ - \beta_i^- = 0 \tag{8.11}$$

$$-\sum_{(x_j,1)\in S_i^+} \alpha_j + \beta_i^+ (\sigma_i + \lambda(1-\sigma_i)) + \beta_i^- (\sigma_i (1-\lambda)) + s_i^+ = 0 \tag{8.12}$$

$$-\sum_{(x_j,-1)\in S_i^-} \alpha_j + \beta_i^+ (1-\sigma_i)(1-\lambda) + \beta_i^- (1 - \sigma_i(1-\lambda)) + s_i^- = 0 \tag{8.13}$$

Now, from non-negativity of the dual variables $\alpha, \beta^+, \beta^-, r, s^+, s^-$, using (8.11) and summing (8.12) and (8.13) we get

$$A_i^+ + A_i^- \leq \beta_i^+ + \beta_i^- \leq CT_i \ . \tag{8.14}$$

(8.12) and (8.14) imply

$$A_i^+ = (CT_i - r_i)(\sigma_i + \lambda(1-\sigma_i)) - \lambda\beta_i^- - s_i^+ \leq (\sigma_i + \lambda(1-\sigma_i))CT_i \ , \tag{8.15}$$

and (8.13) and (8.14) imply

$$A_i^- = (CT_i - r_i)(1 - \sigma_i + \lambda\sigma_i) - \lambda\beta_i^+ - s_i^- \leq (1 - \sigma_i + \lambda\sigma_i)CT_i \ , \tag{8.16}$$

where we use the notation $A_i^+ \triangleq \sum_{(x_j,1)\in S_i^+} \alpha_j$ and $A_i^- \triangleq \sum_{(x_j,-1)\in S_i^-} \alpha_j$.

Finally, given nonnegative values $\{\alpha_j\}_{j=1}^m$ satisfying $A_i^+ + A_i^- \leq CT_i \quad i = 1..K$, any nonnegative assignment to $\{\beta_i^+, \beta_i^-\}_{i=1}^K$ that satisfies

$$\begin{aligned} A_i^+ &\geq \beta_i^+(\sigma_i + \lambda(1-\sigma_i)) + \beta_i^-(\sigma_i(1-\lambda)) \\ A_i^- &\geq \beta_i^+(1-\sigma_i)(1-\lambda) + \beta_i^-(1-\sigma_i(1-\lambda)) \ , \end{aligned}$$

uniquely determines nonnegative values for the rest of the dulal variables $\{r_i, s_i^+, s_i^-\}_{i=1}^K$ to satisfy (8.11), (8.12), and (8.13). The resulting dual program follows:

$$\max_{\alpha_1 \ldots \alpha_m} \left\{ \sum_{j=1}^m \alpha_j - \frac{1}{2} \| \sum_{j=1}^m \alpha_j y_j x_j \|^2 \right\} \tag{8.17}$$

subject to

$$A_i^+ + A_i^- \leq CT_i \qquad i = 1..K \tag{8.18}$$

$$A_i^+ \leq (\sigma_i + \lambda(1-\sigma_i))CT_i \qquad i = 1..K \tag{8.19}$$

$$A_i^- \leq (1 - \sigma_i + \lambda\sigma_i)CT_i \qquad i = 1..K \tag{8.20}$$

$$\sum_{j=1}^m y_j \alpha_j = 0 \tag{8.21}$$

$$\alpha_j \geq 0 \qquad j = 1..m \tag{8.22}$$

$$A_i^+ = \sum_{x_j \in S_i^+} \alpha_j, \ A_i^- = \sum_{x_j \in S_i^-} \alpha_j, \qquad i = 1..K \tag{8.23}$$

where the constraint (8.18) follows from (8.14), the constraints (8.19) and (8.20) follows from (8.15) and (8.16) respectively, and the constraint (8.21) is Eq. (8.10). Using Eq. (8.9), the primal solution w is related to the dual solution by $w = \sum_{j=1}^m \alpha_j y_j x_j$. For

the primal solution b we do the following: combining again (8.12) and (8.13) and using (8.11) we get

$$A_i^+ + A_i^- = CT_i - (r_i + s_i^+ + s_i^-) \leq CT_i \ ,$$

with inequality when one of the dual variables r_i, s_i^+, s_i^- is positive, or equivalently (by KKT) when one of the respective primal variables l_i^+ or l_i^- is 0. Since for support dual variables ($\alpha_j > 0$) either (8.4) or (8.5) are satisfied with equality, we conclude that the optimal primal b is recovered from primal constraints corresponding to dual support variables satisfying $A_i^+ + A_i^- < CT_i$.

The conditions of the dual program may be interpreted as reweighing of the samples of S. The constraints above imply that $A_i^+ + A_i^- \leq CT_i$. Therefore, the total weight of the samples in region X_i is bounded (up to multiplication by the trade-off parameter C) by the weight of region X_i as sampled from the target distribution T. Furthermore, in this general case, within each region X_i the total weights of positive labeled samples A_i^+ (or total weight of negative labeled samples A_i^-), is at most a λ-shift of the empirical positive (or negative, respectively) weight of the region.

We now proceed to consider the two special cases, the optimistic case ($\lambda = 0$) and the pessimistic case ($\lambda = 1$).

8.3.2 Optimistic SVM Adaptation

In this variation we assume that P is 0-shift of Q[3]. Setting $\lambda = 0$ in the primal program (8.3) - (8.8) we the following slightly simplified program in which (8.6) and (8.7) are replaced by $l_i \geq l_i^+ \sigma_i + l_i^- (1 - \sigma_i)$ and whose dual is:

$$\max_{\alpha_1,\ldots,\alpha_m} \left\{ \sum_{j=1}^{m} \alpha_j - \frac{1}{2} \| \sum_{j=1}^{m} \alpha_j y_j x_j \|^2 \right\} \tag{8.24}$$

[3]Note that this is not equivalent to assuming that $Q = P$. The source and target distributions might substantially differ and still have the same probability in each region X_i, and even more importantly, they can arbitrarily differ in the probability that they assign to different regions X_i.

subject to

$$A_i \leq CT_i \qquad i = 1..K \tag{8.25}$$

$$A_i^+ = \sigma_i A_i \qquad i = 1..K \tag{8.26}$$

$$A_i^- = (1 - \sigma_i) A_i \qquad i = 1..K \tag{8.27}$$

$$\sum_{j=1}^{m} y_j \alpha_j = 0 \tag{8.28}$$

$$\alpha_j \geq 0 \qquad j = 1..m \tag{8.29}$$

$$A_i^+ = \sum_{x_j \in S_i^+} \alpha_j, \quad A_i^- = \sum_{x_j \in S_i^-} \alpha_j, \qquad i = 1..K \tag{8.30}$$

$$A_i = A_i^+ + A_i^- \qquad i = 1..K \tag{8.31}$$

For a reweighing interpretation of the dual variables α_j (pertaining to the sample (x_j, y_j) in the primal solution $w = \sum_{j=1}^{m} \alpha_j y_j x_j$) note that at most σ_i portion of the weight allocated to region X_i is allocated to positive samples $(x_j, 1)$ and at most $1 - \sigma_i$ portion of the weight is allocated to negative samples $(x_j, -1)$. Note that this may differ from the naive re-weighting approach that assigns the weight $\alpha_j = C\frac{|T_i|}{|S_i|}$ to every sample $(x_j, y_j) \in S_i$. This is because the naive re-weighting satisfies (8.25) with equality, and is not restricted by (8.28).

8.3.3 Pessimistic SVM Adaptation

In this variation we make no assumptions on P (i.e., P is 1-shift of Q). Again, setting $\lambda = 1$ in (8.3) - (8.8) we get a primal program in which (8.6) and (8.7) are replaced by $l_i \geq l_i^+$, and $l_i \geq l_i^-$, respectively, and the resulting dual program is:

$$\max_{\alpha_1,\ldots,\alpha_m} \left\{ \sum_{j=1}^{m} \alpha_j - \frac{1}{2} \| \sum_{j=1}^{m} \alpha_j y_j x_j \|^2 \right\} \tag{8.32}$$

subject to

$$A_i = \sum_{x_j \in S_i} \alpha_j \leq CT_i \qquad i = 1..K \tag{8.33}$$

$$\sum_{j=1}^{m} y_j \alpha_j = 0 \tag{8.34}$$

$$\alpha_j \geq 0 \qquad j = 1..m \tag{8.35}$$

Again, the primal solution is related to the dual solution by $w = \sum_{j=1}^{m} \alpha_j y_j x_j$ and the dual variables may be interpreted as reweighing of the samples of S: The weight A_i, the total weight of the samples in region X_i, is bounded by the weight of region X_i in the set T. In this *pessimistic* variation there is no restriction on A_i^+ or A_i^- and the weight of region X_i is fully allocated to the region samples with the highest loss. This is natural since the support of the target distribution in every region might only include points of such worst-case loss.

8.4 Robust Domain Adaptation for Regression

In the regression setting the label set Y and the domain X are each a bounded convex subset of $\mathcal{R}$. The classification loss at a sample $z_j = (x_j, y_j)$ is $l(h, z_j) = (h(x_j) - y_j)^2$. Robustness of regression algorithms (e.g., Lasso, see [126]) implies that we may assume a partition $Y = \cup_{v=1}^{K_y} Y_v$ of the label range for which Eq. (8.2) holds, and we define the sample subsets $S_i^v \triangleq S \cap X_i \times Y_v$ and $S^v \triangleq S \cap X \times Y_v$. As before, we assume that the target distribution is λ-shift of the empirical distribution in every region X_i. We use the notation σ_i^v for the empirical probability (in sample set S) of label v in region X_i, and $l_i^v = l_i(h, v)$ for the maximal loss of hypothesis h in $X_i \times Y_v$. To solve the domain adaptation problem in this setting, in addition to the usual constraints on the losses $l_i^v \geq l(h_{w,b}, s)$ for each sample $s \in S_i^v$, we want to constrain the losses to satisfy $l_i \geq l_S^\lambda(h, X_i)$ for each region X_i (thereby minimizing l_i implies that we minimize $l_S^\lambda(h, X_i)$). As before, we achieve the latter condition by using a lower bound on l_i which upper bounds $l_S^\lambda(h, X_i)$ by Eq. (8.1). The algorithm selects among all linear functions $h_{w,b}(x) = w^T x + b$ the one that minimizes the generalization bound, i.e., Eq. (8.2) with an additional appropriate regularization term. We assume that for each region X_i the target probability distribution on the labels ρ_i is λ-shift of the empirical distribution

σ_i. To simplify notation we denote the upper bound of ρ_i^v, the probability of label y_v in region X_i by $\overline{\lambda}_i^v \triangleq \overline{\lambda}^v(\sigma_i)$, and the lower bound on ρ_i^v by $\underline{\lambda}_i^v \triangleq \underline{\lambda}^v(\sigma_i)$. Finally, using a trade-off parameter C results in the following convex quadratic program for Ridge Regression[4]:

$$\min_{w,b,l_1,l_2,\ldots,l_K} C\sum_{i=1}^{K} T_i l_i^2 + \frac{1}{2}\|w\|_2^2 \tag{8.36}$$

subject to

$$l_i^v \geq y_j - (w^T x_j + b) \qquad \forall j = 1..m \text{ s.t. } (x_j, y_j) \in S_i^v \tag{8.37}$$

$$l_i^v \geq (w^T x_j + b) - y_j \qquad \forall j = 1..m \text{ s.t. } (x_j, y_j) \in S_i^v \tag{8.38}$$

$$l_i \geq l_i^v \overline{\lambda}_i^v + \sum_{v' \neq v} l_i^{v'} \underline{\lambda}_i^{v'} \qquad \forall i = 1..K_X, \forall v = 1..K_Y \tag{8.39}$$

For the pessimistic case ($\lambda = 1$) the constraint (8.39) simplifies to $l_i \geq l_i^v$ and for the optimistic case ($\lambda = 0$) it simplifies to $l_i \geq \sum_v \sigma_i^v l_i^v$.

To find the dual representation of the Ridge Regression problem in the general case we introduce a dual variables α_j^+ associated with the first constraint above, α_j^- associated with the second, and B_i^v associated with the last one. Setting the partial derivatives (according to the primal variables w, b, l_i^v, and l_i) of the resulting Lagrangian to 0 we get the following relations: $w = \sum_j (\alpha_j^+ - \alpha_j^-) x_j$, $\sum_j \alpha_j^+ = \sum_j \alpha_j^-$, $B_i = 2CT_i l_i$, and $\sum_{(x_j,y_j)\in S_i^v} (\alpha_j^+ + \alpha_j^-) = \lambda_i^v B_i^v + \overline{\lambda}_i^v B_i^{\overline{v}}$, where $B_i = \sum_v B_i^v$ and $B_i^{\overline{v}} = \sum_{v' \neq v} B_i^{v'}$. Using the above relations we derive the following dual problem:

$$\max_{\alpha,B} \left\{ -\frac{1}{2}\|[\|\Big]\sum_j (\alpha_j^+ - \alpha_j^-) x_j + \sum_j (\alpha_j^+ - \alpha_j^-) y_j - \frac{1}{4C}\sum_i \frac{B_i^2}{T_i} \right\}$$

subject to

[4]We may similarly solve for Lasso Regression: replacing (8.36) with $\min_{w,b,l_1,l_2,\ldots,l_K} C\sum_{i=1}^{K} T_i l_i^2 + \|w\|_1$. The robustness of Ridge Regression may be established in a similar manner to the proof of Lasso robustness in [126].

$$\sum_{(x_j,y_j)\in S_i^v} (\alpha_j^+ + \alpha_j^-) = \lambda_i^v B_i^v + \overline{\lambda}_i^v B_i^{\overline{v}} \qquad i = 1..K_x \ , \ v = 1..K_y$$

$$\sum_j \alpha_j^+ = \sum_j \alpha_j^- \ ,$$

$$\alpha_j^+ \geq 0, \ \alpha_j^- \geq 0 \qquad j = 1..m$$

The primal solution is related to the dual solution by $w = \sum_j (\alpha_j^+ - \alpha_j^-) x_j$. Note also that $\alpha_j^+ \alpha_j^- = 0$.

8.5 Experiment

To illustrate the ability of performing the domain adaptation task by using our methods we considered a synthetic one dimensional binary classification problem. We run the λ-shift domain adaptation SVM on a synthetic data set containing train and test samples from significantly different domains. The experiment confirmed that for several values of λ (not necessarily 0 or 1) the test error of the optimal (with respect to the train set) linear separator may be improved by using the separator returned by our algorithm.

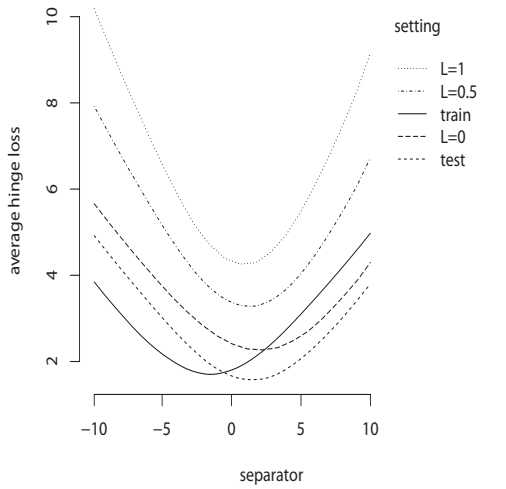


Figure 8.1: Separators performance w.r.t. experiment data

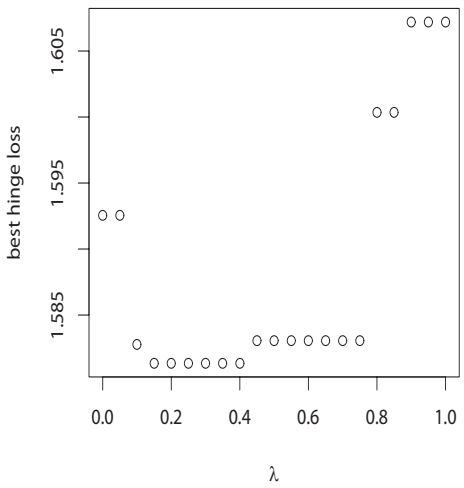


Figure 8.2: Performance of λ-shift SVM optimal separator

Figure 8.1 shows the resulting loss levels of the linear separators. The labeled train samples are a mixture of three Gaussians, centered at -5, 0, and 5, producing positive, negative, and positive labels respectively[5]. Standard deviation is 5 for the Gaussians

[5]Note that the positives and negatives are not linearly separable

generating the positive labels and 3 for those generating negative labels. In the source domain the probabilities of generating a sample for the first, second, or third Gaussians are 0.4, 0.5, and 0.1, respectively, while in the target domain the probabilities are 0.1, 0.5, and 0.4. The upper curves ($L = 1$, $L = 0.5$, and $L = 0$) correspond to the bounds L_S^λ on the average loss of the separator as computed by our λ-shift SVM for $\lambda = 1$, 0.5, and 0.

Now, the best linear separator for the train set will incur significantly higher loss on the test set. However, the best linear separator produced by a λ-shift SVM (corresponding to the lowest points of each of the three upper curves of figure 8.1) may be closer to the optimal linear separator of the test set, and therefore perform better in the target domain[6]. Indeed, as figure 8.2 shows, running the λ-shift SVM (on the same data sets) with λ ranging between 0.2 and 0.4 results in separators having loss that is comparable to the loss of the best test-set separator.

8.6 Conclusion and Future Directions

Assuming the λ-shift property and given a partition of the input-output space, we showed a domain adaptation bound and related robust algorithms for classification and regression. The algorithms were obtained by incorporating the λ-shift restriction into standard machine learning algorithms that minimize the domain adaptation bound. A natural question is whether similar methods may be applied to other robust algorithms described in [126] such as Principal Component Analysis and Feed-Forward Neural Networks to obtain additional domain adaptation algorithms.

Another interesting question is the dependence of our algorithms on the specific input-output space partition. Although the polynomial time-complexity dependence of our algorithms on the number of regions in the partition, the number of regions may be exponential in the dimension of the domain. Therefore, smart partition methods (that may utilize the λ-shift property and the available sample sets) are essential for making our algorithms feasible in high dimensions.

Finally, the sensitivity of our algorithms to the λ-shift assumption may be further investigated. That is, the effect of small changes in the source or target domain

[6]Note that the precise loss values of the λ-shift loss curve (loss bounds calculated by the λ-shift SVM) are not important, the value of interest is the specific separator that achieves minimal loss on the curve!

distributions on the actual λ-shift property and on the algorithm's output.

Part III

Multiagent Learning

Chapter 9

Distributed Information Aggregation and Prediction

The social learning setting is introduced, followed by related background and preliminary results to be used in the work presented in Chapter 10.

9.1 Social Learning

In a *Social learning* setting, a population of agents, each having access to some private information, collaborate (hence *social*) to compute (that is, *learn*) a quantity that depends on the aggregate of the information. The private information possessed by each agent is conveyed through an action of the agent[1] and the resulting computation consists of the total effect of all actions. Two contemporary examples of such social learning are the construction of instantaneous traffic load estimates based on the speed and location reports of sensors in driver's cellphones, and recommendation systems that aggregate rating by individual agents.

The concept of distributed computing, motivated by parallelizing computation time, savings in communication bandwidth, and other needs (e.g., privacy), is, of course, not new in computer science. However, a key characteristic of social learning settings (vs. the more traditional *distributed computing* scenario) is the treatment of the collaborating agents as rational economic entities. Namely, each agent has an incentive to take part (or not) in the computation, that is not necessarily aligned with the goal of having

[1] Such an action depends on the specific setting, and may be explicitly announcing the value of the private information, or some other action that depends on the private information.

a result that genuinely aggregates the private information of the agents.

9.2 Endogenous and Exogenous Settings

We therefore distinguish between two conceptually different scenarios for social learning. In the *Endogenous* scenario, each participating agent has direct stake in the outcome of the social computation. That is, the eventual benefit to the agent is a function of his private type and the computation result. An example of such a setting is *Resource Allocation*, where each agent has a (private) requirement for an amount of work to get done and the social computation (given the reported work requirements by the agents) results in an allocation of work to servers. The eventual time it takes for an agent's work item to complete depends on the work load of the server the item was allocated to. Another example is a *social choice* scenario, where the result of an election is determined by accumulating the individual votes of agents, each having a desired outcome of its own. Yet another example of an endogenous setting is an auction, where an agent places a bid for an item to be auctioned, hoping to win for a price lower than its value to him. The highest offered bid may be considered the result of the social computation - aimed (by the auctioneer) at value discovery through the auction. All in all, in an endogenous scenario, the benefit to the agent as a result of taking some action is not inherent in the action but may depend on the actions of the other agents. Note that agents can mis-report their information to improve their eventual utility (that is, act *strategically*) in such situations.

In the *Exogenous* scenario, the focus of this work, the eventual benefit to the agents depend on an *external* set of exclusive events (that is, exogenous, hence the name) and not on the result of the social computation. The private information of the agents is related to the probabilities of the events occurring and the social computation can be viewed as aiming at an estimate of the events occurrence probabilities, based on the aggregate information held by the agents. The eventual benefit to an agent in such an exogenous setting is inherent in the action taken by the agent (that is, regardless of the actions taken by other agents and the result of the social computation) and determined by the actual event occurring. To make this idea more concrete, consider the bonds market. A bond is a financial instrument in which a firm commits to pay to the bond holder certain amounts of money at different predefined times (as long as the firm does

not go bankrupt). Therefore, the price of a bond reflects a probability of the firm to bankrupt during the relevant time period.[2] Now, each trader (agent) may have a private valuation of a certain bond (depending on his own estimate of the probability of the related firm to bankrupt during the relevant period of the bond), and may take action (e.g., buy) in case the current outstanding price is relatively low. The resulting social computation in this case is the final price of the bond upon all transactions, and may be interpreted as reflecting the aggregated estimate (over all trader's beliefs) of the probability of the firm to bankrupt. The eventual benefit to a trader in this setting depends of course on the action taken (the price paid) and whether the firm is actually bankrupt or not.

9.3 Information Cascading

The above interpretation of the resulting computation as reflecting the aggregate belief of the agents regarding the probability of the occurrence of an event (specifically, the price of the bond as the aggregated beliefs of the traders regarding the bankruptcy probability of the firm) is justified by considering the agents as Bayesian. That is, the agents update their private estimate upon witnessing the actions of other agents, and may act again and again, whenever they may perceive such actions as beneficial to them. Eventually, no more actions take place, and the resulting estimate reflects the common consensus that was socially achieved. Moreover, in settings such as the bonds market where wealth is involved, successful traders making good predictions have their wealth increased over time, while others are taken out of the market by some sort of natural selection. In such situations, the social computation, reflecting the aggregated *wisdom of the crowd*, may also be regarded as an accurate estimate of the actual underlying probabilities of the events occurrences.

One should be careful, though, with such interpretations. A population of rational Bayesian agents in a sequential decision making setting (that is, agents that observe the actions of previous agents and use the Bayes rule to infer regarding those agent's private information) might result in an *Information Cascade* (also termed *herding*) - a situation where the early actions of a relatively small number of agents results in all subsequent agents ignoring their private information and instead imitating the actions

[2] In addition to the risk-free interest rate.

of their predecessors.

Consider for example the following scenario from [48]. An urn has either two blue balls and one red ball or two red balls and one blue. Agents in turn take out a ball from the urn (and put it right back) and predict based on the observed color of the ball and the history (past predictions are publicly available), the state of the urn (two red balls and one blue, or the other way around). A simple analysis shows that even when the initial state of the urn is two red balls and one blue ball, if the first two agents happen to pick a blue ball each in turn (and therefore predict an initial state of two blue balls and one red ball), the rest of the agents would predict the same, regardless of their observed ball. The resulting computation is therefore utterly wrong, whereas the aggregate of the private signals would have resulted in a correct prediction with high probability.

Evidently, such situations, when triggered, might lead to a resulting computation that is by no means representative of the aggregation of the population's private signals. A phenomenon that can't get fixed by merely increasing the population size. However, the risk of herding may be precluded by introducing history independent agents to the population[3] (namely, agents that ignore past actions for making their decision regarding the action to take).

It is interesting to note that an *endogenous* variant of herding exists, where agents may prefer imitating the actions of prior agents due to an eventual benefit that depends on their action and the result of the social computation (rather than on the outcome of some external event). This is the case of setting in which a *network effect* governs the benefit related to actions taken by agents. For example, when choosing a social network to join, the benefit to the agent depends also on the eventual size of the network (the result of the social computation) and therefore an agent might choose to imitate the actions of prior agents and to join a social network that was a-priori less preferred.

9.4 Ingredients of Agent's Strategic Behavior

Market Scoring Rules are used to elicit agents to truthfully act in a way to best represent their actual private information. This allows for the resulting computation (whether directly as a result of the sequence of actions, or by an observing entity) to truly

[3] And of course, having all agents being aware of the presence of such history independent agents.

represent the aggregate of the agents private signals. Market scoring rules make use of scoring rules, which have a very long history, going back to [49], [35] and [61], and studied in much subsequent work ([107], [123], [108], see also [60]). Basically, a scoring rule associates a real score $S(i, \vec{p})$ to a probability distribution $\vec{p} \in \Delta_\Omega$ over some finite result space Ω, and an actual result $i \in \Omega$. This may be interpreted as the benefit to the predictor of $\vec{p}$ upon the actual realization of the event i. If the true underlying distribution over Ω is some $\vec{b}$, then the expected score of a prediction $\vec{p}$ is $E_{i\sim\vec{b}}S(i, \vec{p})$, denoted $S_{\vec{b}}(\vec{p})$. A scoring rule is termed *proper* if for a fixed $\vec{b}$ the expected score $S_{\vec{b}}(\vec{p})$ has a unique maximum at $\vec{p} = \vec{b}$. A proper scoring rule, therefore, induces a forecasting agent believing that the true probability distribution is $\vec{b}$ to act by predicting his actual belief. Two widely used market scoring rules are the *Quadratic Scoring Rule* (QSR) $S(i, \vec{p}) = 1-(1-p_i)^2$ and the *Logarithmic Market Scoring Rule* (LMSR) $S(i, \vec{p}) = \log p_i$.

The abstract score resulting from the usage of a market scoring rule may represent actual money. Now, the actual choice of an action (or lack of action) by an agent takes place by considering a *utility* function that represents the desirability level of the different wealth levels attainable. Intuitively, for rational agents, such utility functions are strict monotone increasing. Linear utility, for example, indicates that a certain increase in wealth results in the same increase in utility, regardless of the initial level of wealth. Such a utility function might not be appropriate in many real situations since it suggests, for example, that an additional certain monetary amount (say 10000$) is equally desirable by a poor person and a rich person. A more adequate utility function for such circumstances is logarithmic utility, suggesting that the multiplication of the wealth by a certain amount is equally desirable at all wealth levels.

A rational agent facing choice is expected to act according to the action that maximizes expected resulting utility. Action is not necessarily expected (if *no action* is a possibility), however, since an agent might already posses a certain level of wealth with a related utility. If, for example, the current utility level of an agent (based on its current wealth) is equal to the expected utility upon taking action, a *risk averse* agent (having a concave utility function) will avoid taking action, a *risk taker* (having a convex utility function) will act, and a *risk neutral* agent (having a linear utility function) will be indifferent in taking action (or not). In prediction markets, presented in the next section, agents are assumed to be risk neutral. That is, will be inclined to act (and thereby hint regarding their private information) whenever their private information

indicates an opportunity of eventual positive benefit.

Finally, consider *game* settings, where the utility to an agent resulting from his choice of action depends also on the action taken by the other agents. To simplify notation we assume there are only two agents taking part in the game. Denote by $u_A(a, b)$ the utility to agent A when agents A and B take actions a and b, respectively. We similarly denote by $u_B(a, b)$ the utility to agent B upon that same choice of actions. Now, for a fixed action b taken by agent B, the *best response* action for agent A has highest utility $\max_a u_A(a, b)$. The pair of action choices (a, b) by agents A and B respectively is said to be an *equilibrium* if each action by an agent is a best response to the other agent's action $a = \arg\max_{a'} u_A(a', b)$ and $b = \arg\max_{b'} u_B(a, b')$. That is, given the other agent's choice, an agent can't improve his utility by switching actions.

9.5 Prediction Markets

A prediction market is a special case of an exogenous social learning setting in which agents may benefit from some private information they have regarding the probability of occurrence of some future event by trading related options[4] - financial instruments that pay a predefined amount to the holder upon realization of an event. For example, in a prediction market for the event of a Democrat or Republican president being elected, an agent estimating the probability of a Democrat winning at 60% would be willing to pay at most 60 cents (ignoring transaction costs, and assuming risk neutrality) for an option paying 1$ in case the Democrat candidate wins the elections (and nothing otherwise). As elaborated above, the outstanding price in a prediction market may be interpreted as representing the aggregate beliefs of the traders population regarding the probability of the relevant event occurring.

In many aspects, this setting resembles the stock market,[5] where the outstanding prices of a security reflects the aggregate private information of traders regarding future events effecting the financial prospects of a firm. In a prediction market, however, trading is usually facilitated by an automated market maker[6] that sets the instantaneous prices and allows for liquidity. As indicated below, on average such a market

[4]Formally, termed *Arrow-Debreu* securities.

[5]One crucial difference however is the stock market being endogenous, since the value of a stock has direct influence on the ability of a firm to pursue business endeavors.

[6]Whereas in a stock market a continuous double auction is used.

maker suffers a loss[7] (although bounded, by tailoring a liquidity parameter). Therefore, the market maker may be viewed as having interest in the the result of the related social computation, namely, the resulting outstanding price reflecting the aggregate of the trader's private signals. Indeed, prediction markets have been implemented in varied domains to assess the probability of significant events (such as the completion date of a strategic project) using the distributed and private information[8] of relevant stake-holders (see e.g., [96], and also[79]).

Now, the market maker seeking to estimate the true probabilities of the outcomes of a future event needs to elicit traders to take action (that is, purchase or sell options if they believe their estimate is higher or lower than the outstanding price, respectively). Furthermore, the prediction market should reward traders that increase the accuracy of the prediction (and penalize those that decrease it). A market scoring rule [65] is therefore used to set such rewards, eliciting traders to take action based upon their belief. This Arrow-Debreu securities setting is shown (see [42]) to be equivalent to the following alternative formulation (now formally defined), more resembling the sequential social learning scenario treated in this chapter so far.

Let the result space $\Omega = \{1, 2, ..., N\}$ be a set of mutually exclusive outcomes of a future event, and let $U = \Delta_\Omega$ be the set of possible states of the market (i.e., every state is a probability distribution over Ω). The market maker sets the initial state by posting an initial probability distribution $\vec{p}_0 \in \Delta_\Omega$, and thereafter traders $t \in \{1, 2, \ldots, T\}$ sequentially change the state of the market by posting probabilities $\vec{p}_1, \vec{p}_2, ..., \vec{p}_T$. Finally, a realization $i \in \{1, 2, ..., N\}$ of F takes place, and each trader t (that changed the state from the previous outstanding prediction $\vec{p}_{t-1}$ to his own prediction $\vec{p}_t$) is rewarded[9] using a proper scoring rule $S : U \times \Omega \to \mathcal{R}$ by the amount $S(i, \vec{p}_t) - S(i, \vec{p}_{t-1})$. To simplify notation we use $S_{\vec{b}}(\vec{p})$ to denote $E_{i\sim\vec{b}}(S(i, \vec{p})$.

Note that by the properties of proper scoring rules, if trader t believes (based on his private information) that $\vec{b}_t \in \Delta_\Omega$ is the true underlying probability of F, his expected reward upon changing the state of the market to $\vec{p}_t \in \Delta_\Omega$ is $S_{\vec{b}_t}(\vec{p}_t) - S_{\vec{b}_t}(\vec{p}_{t-1})$. Since this expected reward is maximized by setting $\vec{p}_t = \vec{b}_t$, it is optimal for a trading agent in a prediction market to post his true belief regarding F, implying that the resulting state

[7] Again, ignoring transaction costs, as assumed throughout this work.

[8] It is worthwhile noting (also considering the *herding* phenomenon detailed above) that the success of such *wisdom of the crowds* methods relies heavily on the private signals being independent conditioned on the exogenous event predicted.

[9] Note that a reward might be negative.

truly reflects the aggregation of the genuine private signals of the traders. Similarly, the expected cost to the market maker[10] if $\vec{b}$ is the true underlying probability of F is

$$S_{\vec{b}}(\vec{p}_T) - S_{\vec{b}}(\vec{p}_0) = \sum_{t=1}^{T} \left(S_{\vec{b}}(\vec{p}_t) - S_{\vec{b}}(\vec{p}_{t-1}) \right) . \tag{9.1}$$

Now, the Bregman divergence (or Bergman loss function) with respect to a convex differential function f,

$$D^f(x, y) \triangleq f(x) - [f(y) + \nabla f(y) \cdot (x - y)] , \tag{9.2}$$

is used by [12] to characterize proper[11] market scoring rules[12] as having the following form:

$$S^f(i, \vec{p}) = -D^f(\vec{e_i}, \vec{p}) , \tag{9.3}$$

where $\vec{e_i}$ is a unit vector with all mass concentrated in the i^{th} coordinate. This makes sense since the Bregman divergence is non negative and $S^f(i, \vec{p})$ is 0 when $\vec{p} = \vec{e_i}$. For example, the quadratic scoring rule

$$S^f(i, \vec{p}) = 1 - (1 - p_i)^2 , \tag{9.4}$$

corresponds to $f(\vec{p}) = Q(\vec{p}) \triangleq \frac{1}{2}||\vec{p}||^2$, and the logarithmic scoring rule

$$S^f(i, \vec{p}) = \log p_i , \tag{9.5}$$

to $f(\vec{p}) = L(\vec{p}) \triangleq \sum_i p_i \log p_i$.

An interesting connection between prediction markets and on-line learning for prediction using experts is presented in [42]. After showing the equivalence between a market scoring rule-based prediction market and a prediction market based on Arrow-Debreu options (i.e., by explicitly formulating the corresponding transaction in the latter to a state update in the former), the authors reduce the on-line learning setting to a prediction market scenario (specifically, by associating every expert with a possible

[10] The goal of the market marker is to learn the true probabilities, and thus, in general, would have to subsidize the market, especially if the resulting price $\vec{p}_T$ is close to the true underlying probability of F.

[11] A market scoring rule is called *proper* if it elicits truthful reporting by the agent.

[12] The characterization actually applies in a more general setting (and is stated accordingly in such general terms) where the state space U is different from Δ_Ω, allowing for efficient pricing even for scenarios of very large outcome spaces.

outcome, interpreting every reported loss vector as a market transaction, and the resulting probability state of the prediction market as the weight to be used over experts by the on-line learner). Specifically, they show that for the logarithmic market scoring rule (LMSR), the prediction market's pricing mechanism effectively implements the randomized weighted majority *no-regret* algorithm and re-derive the no-regret bounds for RWM using the smoothness properties of the LMSR.

Yet another application of prediction markets to learning is introduced in [13], where a setting in which agents collaborate by sequentially updating a hypothesis solution to a supervised learning problem is investigated. In this setting, the private information of an agent corresponds to the machine learning algorithm used, the probability of the event corresponds to the posted hypothesis by the agents, and the eventual outcome corresponds to the test set of the supervised learning problem (which is unavailable to the agents). Finally, the scoring rule of a prediction market corresponds to a loss function in this setting, such that after all agents posted their updates, the training set is revealed and each agent's profit is the difference between the loss (on the test set) of his posted hypothesis and the loss of the preceding hypothesis. As with prediction markets (which are shown to be a special case of the setting), the resulting hypothesis may be interpreted as the aggregate of all the methods (the private 'know how') used by the agents. Therefore, this *crowdsourcing* learning setup allows for collaboration among (potentially competing) agents in which every agent is compensated according to the relative improvement achieved by his proposed hypothesis.

9.6 A Prediction Market as an Estimator

As already noted, a prediction market may be viewed as a means for the market maker to get access to the private information available to the set of traders regarding the probability of occurrence of some future event. The market maker, however, is expected to suffer a loss in such a setting. Indeed, the market maker posts the first probability $\vec{p}_0$ (and related score $S_{\vec{b}}(\vec{p}_0)$), having little knowledge regarding the true underlying probability $\vec{b}$ of the event. Furthermore, at the end of the trading chain, the resulting probability in the market $\vec{p}_T$ is assumed to reflect the aggregate of the trader's private knowledge, that is, an accurate estimate of $\vec{b}$. It is therefore expected that the score $S_{\vec{b}}(\vec{p}_T)$ be higher than the initial $S_{\vec{b}}(\vec{p}_0)$. All in all, the expected market maker cost

$S_{\vec{b}}(\vec{p}_T) - S_{\vec{b}}(\vec{p}_0)$ (which by (9.1) is the sum of the payments to all participating traders) is expected to be positive, thereby, reflecting an expected loss to the market maker, which is effectively subsidizing the market.

In this section we quantify the relation between the expected cost to the market maker and the quality of the resulting price $\vec{p}_T$ as an estimator of the true underlying probability $\vec{b}$. More specifically, we establish and quantify the trade-off for the market maker between its expected cost and the expected estimation error by showing a *conservation rule* stating that the sum of those two quantities is constant. This conservation rule is stated in a general form, for market scoring rules and prediction error measures that are based on Bregman loss functions.

We consider settings in which the private information of every trading agent t is a random variable V_t. Since the trading agents use their private information to act (that is, to change of the outstanding price from $\vec{p}_{t-1}$ to $\vec{p}_t$), the outstanding price $\vec{P}_t$ is also a random variable (that depends on the set $\{V_i\}_{i=1}^t$). Hence, the resulting $\vec{P}_T$ may be regarded as an estimator of a true (yet unknown to the traders) underlying probability $\vec{b}$. A common metric for measuring the performance of such estimators is the *Mean Squared Error* (MSE). Specifically, for $\vec{P}_T$, an estimator for $\vec{b}$ we have that the MSE is the sum of the estimator's variance and squared bias:

$$\mathrm{MSE}(\vec{P}_T) \triangleq E[\|\vec{P}_T - \vec{b}\|^2] = \sigma^2(\vec{P}_T) + \|\beta(\vec{P}_t)\|^2 \ , \tag{9.6}$$

where $\beta(\vec{P}_T) = E[\vec{P}_T - \vec{b}]$ is the bias of the estimator and $\sigma^2 = E[\|\vec{P}_T - \beta(\vec{P}_T)\|^2]$ is the variance. Note that expectations above are taken over the random nature of the realization of the estimator $\vec{P}_T$, that is, over the realization of $\vec{V} = (V_1, \ldots, V_T)$.

As noted above using (9.3), any proper market scoring rules corresponds to a Bregman loss function (which in turn is characterized by a differential convex function f). We can use the Bregman divergence to also generalize the MSE as follows. Let $\vec{P} \in \Delta_N$ be a random prediction vector, and fix $\vec{b} \in \Delta_N$ (this may be, but not necessarily, the true underlying distribution that generates the signals in $\vec{V}$ which $\vec{P}$ depends on). Define the mean f-Bergman loss of $\vec{P}$ with respect to $\vec{b}$ as follows:

$$ME^f(\vec{b}, \vec{P}) \triangleq E_{\vec{V}}[D^f(\vec{b}, \vec{p})]. \tag{9.7}$$

We get as special cases $ME^Q(\vec{b}, \vec{P}) = E_{\vec{V}}[\|\vec{b} - \vec{p}\|^2]$ (the mean square error), and

$ME^L(\vec{b},\vec{P}) = E_{\vec{V},\vec{p}\sim\vec{P}}[KL(\vec{b}||\vec{p})] = E_{\vec{V},\vec{p}\sim\vec{P}}[\sum_i b_i \log\frac{b_i}{p_i}]$.

Turning back to scoring rules, we use the characterization (9.3) and the definition of the Bregman loss (9.2) to get the following formulation for the expected f-Bregman score of a resulting prediction $\vec{p}$, when the true underlying distribution over the outcomes is $\vec{b}$

$$S^f_{\vec{b}}(\vec{p}) \triangleq \sum_{i=1}^N b_i S^f_i(\vec{p}) = f(\vec{b}) - D^f(\vec{b},\vec{p}) - \sum_{i=1}^N b_i f(\vec{e_i}) \ . \tag{9.8}$$

Now, the above definitions of Bregman-based scoring rule S^f and mean error ME^f apply to any estimator $P_T = \theta(\vec{V})$. Noting that the only term in (9.8) that depends on $\vec{p}$ is the f-Bergman loss function, we get that the expected score of the estimator $\theta(\vec{V})$ is

$$E_{\vec{V}}[S^f_{\vec{b}}(\theta(\vec{V}))] = f(\vec{b}) - ME^f(\vec{b},\theta(\vec{V})) - \sum_{i=1}^N b_i f(\vec{e_i}) \ .$$

All in all, we have established that the sum of an estimator's score and squared error is constant:

Lemma 12. *For a scoring rule $S^f(i,\vec{p}) = -D^f(\vec{e_i},\vec{p})$ based on a convex differential function f and a fixed b, the performance of $\theta(\vec{V})$ a predictor of b (where the random variable $\vec{V}$ may depend on b) satisfies*

$$ME^f(\vec{b},\theta(\vec{V})) + E_{\vec{V}}[S^f_{\vec{b}}(\theta(\vec{V}))] = f(\vec{b}) - \sum_{i=1}^N b_i f(\vec{e_i}) \ .$$

Note the non positivity of the right hand constant (by convexity of f).

Finally, we define the following notation for the cost to the market maker

$$C^f(\vec{b},\vec{P_0},\vec{P_T}) \triangleq E_V[S^f_{\vec{b}}(\vec{P_T}) - S^f_{\vec{b}}(\vec{P_0})] = E_V[S^f_{\vec{b}}(\vec{P_T})] - S^f_{\vec{b}}(\vec{P_0}) \ ,$$

and get the desired result by applying Lemma 12 to our prediction market setting

Corollary 13. *For a convex differential function f, for a fixed $\vec{b}$ and initial price $\vec{P}_0$ in a prediction market based on a scoring rule $S^f(i,\vec{p}) = -D^f(\vec{e_i},\vec{p})$ that realizes $\vec{P_T}$ as an estimator for $\vec{b}$:*

$$ME^f(\vec{b},\vec{P_T}) + C^f(\vec{b},\vec{P_0},\vec{P_T}) = D^f(\vec{b},\vec{P_0})$$

The corollary follows since,

$$\begin{aligned} ME^f(\vec{b}, \vec{P_T}) + C^f(\vec{b}, \vec{P_0}, \vec{P_T}) &= ME^f(\vec{b}, \vec{P_T}) + E_{\vec{V}}[S^f_{\vec{b}}(\vec{P_T})] - S^f_{\vec{b}}(\vec{P_0}) \\ &= f(\vec{b}) - \sum_{i=1}^{N} b_i f(\vec{e_i}) - S^f_{\vec{b}}(\vec{P_0}) = D^f(\vec{b}, \vec{P_0}) \ , \end{aligned}$$

where the second equality by Lemma 12 and the last equality by (9.8).

Chapter 10

History Independent Learning

We study a simple model of collaborative learning where agents sequentially contribute based on their private signal but do not have access to the history, rather only to the most recent state. We compare the optimal (Bayes) estimator for the unknown bias of a coin given T independent Bernoulli signals, to the estimate produced by a learning process where the signals are distributed amongst T agents. For a simple strategy space available to the agents, we consider statistical properties of the resulting estimator in general, for a social optimal update rule, and for the equilibrium update rule.

More specifically, we study the following scenario: some future event is to occur with (unknown) probability b. Individual agents (sharing a common prior on b) get independent signals, 1 with probability b and 0 otherwise. Agents arrive in a random order and act exactly once (unaware of the past actions or their order of arrival) by updating the outstanding estimate. The utility to the agents, determined upon the realization of the event, using the quadratic scoring rule, is the difference between the score of their posted estimate and the score of the estimate at their arrival.

In order to analyze the dynamics in this setting, we need to designate the class of available agents' strategies. We suggest that exponential moving averages is an appropriate strategy class. Applying the exponential moving average to update the last prediction has the effect that more recent signals (and, in particular, the private signal of the agent) are given more weight than older signals whose effect will decay with time. Also, to update the exponentially moving average incrementally, the agent does not need access to the action's history and specifically does not need to know how many updates have been previously done.

Our setting can be considered to model a *trusted recommendation chain*, where the

perception regarding the quality of a product is sequentially updated through person-to-person recommendations.[1] After observing the recommendation of his predecessor, each person in line gets the opportunity to use the product and provide to the next an updated recommendation, based on his own experience. It is crucial to note that in such a setting people are only considering recommendations from trusted others, and will act truthfully since they care about their reputation by the advised person.

10.1 Model and Preliminaries

A model for a social computation that aggregates distributed signals is presented, where the observed signals are generated according to some (unknown) value. The interpretations of the resulting computation as an estimator of the underlying value is discussed as well as related performance metrics and a connection to prediction market's prices and scores.

10.1.1 The Unknown Bias Generative Model

In our model there is a random variable B which is distributed uniformly in $[0,1]$, i.e., $B \sim U[0,1]$. The random variable B is sampled once, and its realization is $b \sim B$. The realized value b represents the underlying *bias* of a binary (Bernoulli) random variable V. That is, given $b \in [0,1]$, $V \sim Bernoulli(b)$, with $\Pr[V=1] = b = 1 - \Pr[V=0]$. We denote a series of T i.i.d. such random variables by $\vec{V}_{[1,T]} = (V_1, \ldots, V_T)$ and their respective realizations by $\vec{v}_{[1,T]} = (v_1, \ldots, v_T)$. An *estimator* $\theta(\cdot) : \{0,1\}^T \to \mathcal{R}$ for the unknown bias b is a function that maps an observed sequence $\vec{v}_{[1,T]}$ to some estimated bias in $[0,1]$. Two such estimators are presented next.

10.1.1.1 Bayes estimator for b

Given $\vec{v}_{[1,T]}$, a sequence of T i.i.d. realizations of a binary random variable $V \sim Bernoulli(b)$, the *Bayes estimator* $\widehat{\theta}(\cdot)$ for b is

$$\widehat{\theta}(\vec{v}_{[1,T]}) = \frac{\sum_{t=1}^{T} v_t + 1}{T+2}. \tag{10.1}$$

[1] As opposed to public recommendation systems, e.g. Amazon ratings, where each recommendation is available to the whole population.

The Bayes estimator $\widehat{\theta}(\vec{v})$ is both the mean and the mode of the posterior distribution (assuming a uniform prior for B). Equivalently, the Bayes estimator can be computed iteratively as follows:

$$\widehat{\theta}(\vec{v}_{[1,t]}) = \left(1 - \frac{1}{t+2}\right)\widehat{\theta}(\vec{v}_{[t-1]}) + \frac{1}{t+2}v_t, \qquad \text{for } t = 1, \ldots, T. \tag{10.2}$$

Note that $\widehat{\theta}(\emptyset) = \frac{1}{2}$ which is consistent with $B \sim U[0,1]$.

10.1.1.2 The Exponential Moving Average Estimator for b

The Exponential Moving Average (EMA) estimator $\theta_\gamma(\cdot)$ is parameterized by a predefined constant γ:

$$\theta_\gamma(\vec{v}_{[1,T]}) = (1-\gamma)^T\theta_\gamma(\emptyset) + \sum_{t=1}^{T}(1-\gamma)^{T-t}\gamma v_t, \tag{10.3}$$

where $\theta_\gamma(\emptyset) = 1/2$. The exponential moving average "values" signals with higher indices more than signals with lower indices, so it behaves somewhat like a moving average.

Equivalently, $\theta_\gamma(\cdot)$ can be defined iteratively:

$$\theta_\gamma(\vec{v}_{[1,t]}) = (1-\gamma)\theta_\gamma(\vec{v}_{[1,t-1]}) + \gamma v_t \qquad \text{for } t = 1, \ldots, T. \tag{10.4}$$

If one compares the iterative form of the Bayes estimator $\widehat{\theta}(\cdot)$ with the iterative form of the exponential moving average $\theta_\gamma(\cdot)$, they seem quite similar. The difference being that $1/(t+2)$ in (10.2) is replaced with a fixed constant γ in (10.4).

10.1.2 Social Learning of the Unknown Bias Generative Model

A social learning process is established to learn the unknown probability b of some future event F occurring within some time frame. It is known that F will occur with some unknown probability b, where b was uniformly generated over $[0,1]$. That is, we may identify the occurrence of F with a random variable $V \sim Bernoulli(b)$, taking the value 1 if F occurs and 0 otherwise.

Now, each of T participating agents receives one binary signal v_t, the realization of a random variable V_t (identical to V as defined above, and independent of the others). The T agents sequentially update (each in turn, only once) a posted probability estimate. Namely, agent t posts at time t an estimate p_t that may depend on the

previously posted (outstanding) estimate p_{t-1} and its private signal v_t. The resulting posted estimate p_T, an aggregation of the T agent's private signals, is an estimator for b, the unknown probability of F occurring. Upon the last agent posting p_T, yet a final a realization of V takes place to determine the outcome F.

To motivate rational agents, the quadratic scoring rule (9.4) is used to compensate agents as follows: $S_F(p) = 1-(1-p)^2$ and $S_{\overline{F}}(p) = 1-p^2$ defines the score attributed to an agent posting a price p upon the eventual occurrence or no occurrence (respectively) of the event F. For an underlying ('true') probability b of F occurring, we denote by $S_b(p)$ the expected score related to a posted probability estimate p,

$$S_b(p) \triangleq bS_F(p) + (1-b)S_{\overline{F}}(p) \ ,$$

and we have for the quadratic scoring rule

$$S_b(p) = b(1-(1-p)^2) + (1-b)(1-p^2) = 2bp - p^2 + 1 - b \ . \tag{10.5}$$

Now, the net score of an agent that updated the probability from p_{t-1} to p_t (that is, agent $t-1$ posted p_{t-1}) is $S_F(p_t) - S_F(p_{t-1})$ if F occurred and $S_{\overline{F}}(p_t) - S_{\overline{F}}(p_{t-1})$ otherwise. Again, for an underlying probability b of F occurring, the net score of such agent is $S_b(p_t) - S_b(p_{t-1})$. Recall that the quadratic scoring rule (being strictly proper) motivates the agents to post their best estimate of the probability of F occurring [2].

If agents knew their position in the sequence, and all agents were rational, then they could update the estimate for b using the iterative Bayes update (10.2). We consider the alternative setting where agents are history independent, and that the order of updates is a random permutation of the agents. In this case, agents cannot update the estimate for b using the Bayes estimator — the t^{th} agent does not know how many updates have been done, i.e., $t-1$, and therefore does not know $\frac{1}{t+2}$. For reasons of symmetry, in this history independent setting, the update strategies of all agents could be identical, since we know that the update cannot depend on the position in the trading sequence. It is thus natural to consider exponential moving average updates. As we also assume that agents participate only once, agent t means the agent who updates at the t'th time period. Let $P_t(\vec{V})$ be the random variable giving the distribution of the prediction of

[2] Note that such posterior estimate should take into account the outstanding posted probability estimate resulting from previous updates.

agent t, *i.e.*, $p_t \sim P_t$. P_t depends on $\vec{V}_{[1,t]}$ and how agents $1, \ldots, t-1$ compute their prediction.

10.1.3 Estimator Performance Metrics

A common metric for assessing the performance of an estimator is the *Mean Squared Error* (MSE). Specifically, for $P_T = \theta(\vec{v})$, an estimation for b we have that the MSE is the sum of the estimator's variance and squared bias:

$$\mathrm{MSE}(P_T, b) \triangleq E_{\vec{V}}[P_T - b)^2] = \sigma^2_{\vec{V}}(P_T) + \beta^2_{\vec{V}}(P_T) \ , \tag{10.6}$$

where $\beta(P_T) = E_{\vec{V}}[P_T - b]$ is the bias of the estimator $P_T = \theta(\vec{V})$ and $\sigma^2(P_T) = E_{\vec{V}}[(P_T - E_{\vec{V}}[P_T])^2]$ is its variance.

To assess the quality of the random variable $P_T(\vec{V})$ as an estimator[3] of b we consider the following properties:

- $\max_b \mathrm{MSE}(b, P_T)$: The worst case MSE, (providing a measure for the closeness of a realized estimate $p_t \sim P_T$ to the true b.
- $\max_b \mathrm{Prob}_{\vec{V}}(|P_T - b| > \epsilon)$: The worst case confidence $\delta(\epsilon)$ of a desired ϵ-accuracy.[4] Quantifying the confidence-accuracy trade-off in terms of T and γ where applicable.
- $E_B[\mathrm{MSE}(b, P_T)]$: The expected MSE, over the realization of b.

The following claim, a corollary of Corollary 13, indicates that in this social learning setting, the performance of an estimator may also be quantified by the expected total net score of the agents $C(b, P_0, P_T) \triangleq E_{\vec{V}}[S_b(P_T) - S_b(P_0)|B = b]$.

Claim 14. *For a fixed b and initial price P_0 in a sequential social learning setting based on the quadratic scoring rule,*

$$\mathrm{MSE}(b, P_T) + C(b, P_0, P_T) = (b - P_0)^2.$$

Indeed, for a fixed b, by the above claim, the sum of the expected total net score and the MSE of P_T is constant, implying that optimizing for total net score is equivalent to optimizing for minimal worse-case MSE, the first metric above.

[3] To simplify notation we omit the dependence on $\vec{V}$ of the estimator P_T.
[4] And the related inverse, worst case accuracy $\epsilon(\delta)$ of a desired δ-confidence.

Revisiting the notation of Corollary 13, note that the Bayes estimator $\widehat{\theta}(\cdot)$ is the conditional expectation of the posterior distribution of B given the signals $\vec{v}$. We conclude that for any Bergman loss function (9.3), as shown in [20], the Bayes estimator minimizes (among all estimators $\theta(\vec{V})$) the estimation error ME^f and is therefore our key benchmark in assessing the performance of EMA estimators $\theta_\gamma(\cdot)$ for $\gamma \in [0,1]$.

10.1.4 Notation

To simplify notation, we use the following abbreviated notation throughout the remainder of this chapter. The process we consider has two stages, first, a random choice $b \sim B$ is sampled. Following which, signals $\vec{v}_{[1,T]} \sim V^T$ are generated. We consider expectation of two kinds: expectations over the realization of $\vec{v}_{[1,T]}$ for a fixed b, and the joint expectation over both realizations of b and $\vec{v}_{[1,T]}$.

Therefore, for any random variable X, we use the notation $E(X) \triangleq E_{\vec{V}}(X|B=b)$ for taking the expectation over $\vec{V}$ while conditioning over $B=b$. The notation $E_B(X)$ and $E_{B,\vec{V}}$ means taking the expectation also with respect to B. Similarly, we use the notation $\Pr(X) \triangleq \Pr_{\vec{V}}(X|B=b)$, *i.e.*, the conditional probability given $B=b$ of the event X.

10.2 Estimator's Performance

In this section we study the basic performance measures for the Bayes estimator and EMA as a function of T, the total number of agents. We first compute the worse case MSE and high probability deviation for the Bayes estimator and the EMA estimator (for a general γ). Those results would be particularly handy when we later compute for specific values of γ and this will allow us to measure the performance in such cases and compare them to the Bayes estimator.

Later in this section, we compute the value of γ that maximizes the expected profits of the agents in a social learning process that is based on the quadratic scoring rule, and study its performance. The main goal is to show that the loss due to the restriction of the agents to use EMA is rather minimal, assuming a non-strategic behavior of the agents. This would be especially important in the next section where we consider the equilibrium under the assumption of strategic agents. It will allow us to compare the performance that results from limiting the agents to use a dictated γ for the EMA

versus the performance due to the strategic behavior of the agents.

We start with the analysis of the Bayes estimator, which is optimal as mentioned in Section 10.1.3 and therefore a natural benchmark. Before stating the first key theorem of this section, we need three technical lemmas. The first is McDiarmid's inequality.

Lemma 15 ([86]). *Let the function $f : D_1 \times \cdots \times D_n \to \mathbb{R}$ satisfy for any $i \in 1, \ldots, n$*

$$\sup_{x_1,\ldots,x_n,\hat{x}_i} |f(x_1, \ldots, x_n) - f(x_1, \ldots, x_{i-1}, \hat{x}_i, x_{i+1}, \ldots, x_n)| \leq c_i.$$

Let $X_1, \ldots, X_n$ be independent random variables, where $X_i \in D_i$. Then

$$\Pr[|f(X_1, \ldots, X_n) - E[f(X_1, \ldots, X_n)]| \geq \epsilon] \leq 2e^{-2\epsilon^2/(\sum_{i=1}^n c_i^2)}$$

The second lemma is a bound for the worse case probability (over possible values of b, the expected value of each of the signals $\{V_t\}_{t=1}^T$) of ϵ-deviation of an estimator $P_T = \widehat{\theta}(\vec{V})$ from b.

Lemma 16. *For any estimator $P_T = \widehat{\theta}(\vec{V})$ and any $b \in [0, 1]$, define*

$$\beta_w \triangleq \max_b |E(P_T) - b| \text{ , then}$$

$$\max_b Pr(|P_T - b| > \epsilon) \leq \max_b Pr(|P_T - E(P_T)| > \epsilon - \beta_w) \ .$$

Proof. Fix $\epsilon > 0$. For any $b \in [0, 1]$ and any p_t (a realization of P_T) we have

$$|p_T - b| = |p_t - E(P_T) + E(P_T) - b| \leq |p_t - E(P_T)| + |E(P_T) - b| \leq |p_T - E(P_T)| + \beta_w$$

Therefore, the event $|P_T - b| > \epsilon$ implies the event $|P_T - E(P_T)| + \beta_w > \epsilon$, and we conclude that for any $b \in [0, 1]$, $Pr(|P_T - b| > \epsilon) \leq Pr(|P_T - E(P_T)| + \beta_w > \epsilon)$.

Finally, since the above holds for any $b \in [0, 1]$ it holds for $b^* \triangleq \arg\max_b Pr(|P_T - b| > \epsilon)$ and we get

$$\begin{aligned}\max_b Pr(|P_T - b| > \epsilon) &\leq Pr(|P_T - E(P_T)| > \epsilon - \beta_w | B = b^*) \\ &\leq \max_b Pr(|P_T - E(P_T)| > \epsilon - \beta_w)\end{aligned}$$

□

The third lemma is a computation of the expected score[5] (10.5) of an agent using the Bayes estimator in the social learning process.

Lemma 17. *For the Bayes estimator $P_T = \widehat{\theta}(\vec{V})$, and $P_0 = \frac{1}{2}$,*

$$E_{B,\vec{V}}[2bP_T - P_T^2] = \frac{1}{3} - \frac{1}{6(T+2)}$$

Proof. Recall that $E[V_t] = b$, $E[V_iV_j] = b^2$ for $i \neq j$, and $E[V_i^2] = b$. Also, we have $E_B[b] = \frac{1}{2}$, and $E_B[b^2] = \frac{1}{3}$. Using the Bayesian update $P_T = \frac{(\sum_{t=1}^T V_t)+1}{T+2}$, we have

$$\begin{aligned}
E_{B,\vec{V}}[2bP_T - P_T^2] =& E_{B,\vec{V}}\left[2b\frac{\sum_{t=1}^T V_t + 1}{T+2} - \left(\frac{\sum_{t=1}^T V_t + 1}{T+2}\right)^2\right] \\
=& E_B\left[2b\frac{Tb+1}{T+2}\right] - E_B\left[\frac{T^2b^2 + T(3b-b^2)+1}{(T+2)^2}\right] \\
=& \frac{(2/3)T+1}{T+2} - \frac{(1/3)T^2 + (7/6)T + 1}{(T+2)^2} \\
=& \frac{1}{3} - \frac{1}{6(T+2)}
\end{aligned}$$

□

The following theorem establishes upper bounds on the MSE and high probability deviation, for any value of $b \in [0,1]$, and the expected MSE assuming a uniform prior for $b \sim B$. The interpretation of a worse case upper bound is a guarantee about the performance measures regardless of the realization $b \sim B$, in contrast to the expectation over $b \sim B$ whose guarantee is much harder to interpret given a specific realization of $b \in [0,1]$.

Theorem 18. *For the Bayes estimator $P_T = \widehat{\theta}(\vec{V})$, for any $b \in [0,1]$,*

(1) $\text{MSE}(b, P_T) \leq \frac{1}{4(T+2)}$,

(2) With probability at least $1-\delta$ we have $|P_T - b| \leq \frac{1}{T+2} + \sqrt{\frac{\ln\frac{1}{\delta}}{2T}}$, or equivalently, $Pr(|P_T - b| > \epsilon) \leq \exp(-2(\epsilon - \frac{1}{T+2})^2 T)$, and

(3) For $P_0 = \frac{1}{2}$, $E_B[\text{MSE}(b, P_T)] = \frac{1}{6(T+2)}$.

Proof. For the first claim (1), recall that $\text{MSE}(b, P_T) = \beta^2(P_T) + \sigma^2(P_T)$. Now, we have

$$\beta^2(P_T) = (E(P_T) - b)^2 = \left(\frac{Tb+1}{T+2} - b\right)^2 = \left(\frac{1-2b}{T+2}\right)^2 \text{ , and}$$

[5] Disregarding the terms that do not depend on the estimate P_t, those are canceled out when computing the net score $S_b(P_t) - S_b(P_{t-1})$

$$\sigma^2(P_T) = \sigma^2\left(\frac{\sum_{t=1}^{T} V_t + 1}{T+2}\right) = \sigma^2\left(\frac{\sum_{t=1}^{T} V_t}{T+2}\right) = \frac{1}{(T+2)^2}\sum_{i=1}^{T}\sigma^2(V_t) = \frac{Tb(1-b)}{(T+2)^2} ,$$

where the last equality follows since each V_t is a Bernoulli random variable with probability of success b. Taking the sum of the two expressions above we can see that the maximum is attained for $b = \frac{1}{2}$ resulting in an upper bound for the MSE of $\frac{T}{4(T+2)^2} \leq \frac{1}{4(T+2)}$.

For the high probability bound (2), we first use McDiarmid's inequality (Lemma 15) as follows. Recall that $P_T = \widehat{\theta}(\vec{V}) = \frac{1}{T+2} + \sum_{t=1}^{T}\frac{1}{T+2}V_t$. This implies that the influence of V_t on P_T is abounded by $c_t = \frac{1}{T+2}$. Therefore, $\sum_t c_t^2 = \frac{T}{(T+2)^2}$ and we get:

$$Pr(|P_T - E(P_T)| > \epsilon) \leq e^{\frac{-2\epsilon^2}{\sum_t c_t^2}} = e^{\frac{-2\epsilon^2(T+2)^2}{T}} \leq e^{-2\epsilon^2 T}$$

Plugging into Lemma 16, we get

$$\max_b Pr(|P_T - b| > \epsilon) \leq \max_b Pr(|P_T - E(P_T)| > \epsilon - \frac{1}{T+2}) \leq e^{-2(\epsilon - \frac{1}{T+2})^2 T}$$

As required. The equivalent formulation is achieved by setting $\delta = e^{-2(\epsilon - \frac{1}{T+2})^2 T}$ and solving for ϵ.

For the third claim (3), we have

$$E_{B,\vec{V}}[MSE_V(b, P_T)] = E_{B,\vec{V}}[S_b(P_0) - S_b(P_T)] + E_B[(b - P_0)^2] \tag{10.7}$$

$$= \frac{1}{3} - E_{B,\vec{V}}[2bP_T - P_T^2] = \frac{1}{6(T+2)} , \tag{10.8}$$

where the first equality is by Claim 14, the second equality by the definition of the quadratic scoring rule (9.4), and the last equality by Lemma 17. □

Next we establish bounds for the performance of the Exponential Moving Average (EMA) as a function of the parameter γ and the number of agents T. Again, we first need a couple of technical lemmas. The first, a closed form for the variance of the EMA estimator.

Lemma 19. *For the EMA estimator (10.3),* $\sigma^2(\theta_\gamma(\vec{V})) = \frac{b(1-b)\gamma}{2-\gamma} - \frac{b(1-b)\gamma}{2-\gamma}(1-\gamma)^{2T}$

Proof. For $P_T = \theta_\gamma(\vec{V})$ we have

$$E[P_T^2] = E\left[\left(P_0(1-\gamma)^t + \sum_{t=1}^{T}\gamma(1-\gamma)^{T-t}V_t\right)^2\right]$$
$$= P_0^2(1-\gamma)^{2T} + 2P_0(1-\gamma)^T E[\sum_{t=1}^{T}\gamma(1-\gamma)^{T-t}V_t] + \sum_{i,j=1}^{T}\gamma^2(1-\gamma)^{2T-i-j}E[V_iV_j]$$

The first term is a scalar. For the second term we have,

$$E[\sum_{t=1}^{T}\gamma(1-\gamma)^{T-t}V_t] = \gamma b\sum_{t=0}^{T-1}(1-\gamma)^t = \gamma b\frac{1-(1-\gamma)^T}{\gamma} = b\left(1-(1-\gamma)^T\right) .$$

For the last term, recall that $E[V_i^2] = b$ and $E[V_iV_j] = b^2$ for $i \neq j$. We have

$$\begin{aligned}
\sum_{i,j=1}^{T}\gamma^2(1-\gamma)^{2T-i-j}E[V_iV_j] &= \sum_{i,j=1,i\neq j}^{T}\gamma^2(1-\gamma)^{2T-i-j}E[V_iV_j] + \sum_{t=1}^{T}\gamma^2(1-\gamma)^{2T-2t}E[V_t^2] \\
&= \sum_{i,j=1}^{T}\gamma^2(1-\gamma)^{2T-i-j}b^2 + \sum_{t=1}^{T}\gamma^2(1-\gamma)^{2T-2t}(b-b^2) \\
&= \left(\sum_{t=1}^{T}\gamma(1-\gamma)^{T-t}b\right)^2 + \sum_{t=1}^{T}\gamma^2((1-\gamma)^2)^{T-t}(b-b^2) \\
&= \left(b\gamma\frac{1-(1-\gamma)^T}{\gamma}\right)^2 + \gamma^2\frac{1-(1-\gamma)^{2T}}{1-(1-\gamma)^2}b(1-b) \\
&= (1-(1-\gamma)^T)^2b^2 + \gamma\frac{1-(1-\gamma)^{2T}}{2-\gamma}b(1-b).
\end{aligned}$$

Also, $E(P_T) = b + (1-\gamma)^T(P_0 - b)$ and we have $E^2(P_T) = b^2 + 2b(1-\gamma)^T(P_0-b) + (1-\gamma)^{2T}(P_0-b)^2$. Finally, $\sigma^2(P_T) = E(P_T^2) - E^2(P_T)$ and we get

$$\begin{aligned}
\sigma^2(P_T) =& P_0^2(1-\gamma)^{2T} + 2bP_0(1-\gamma)^T(1-(1-\gamma)^T) + \gamma\frac{1-(1-\gamma)^{2T}}{2-\gamma}b(1-b) + (1-(1-\gamma)^T)^2b^2 \\
&- (b^2 + 2b(1-\gamma)^T(P_0-b) + (1-\gamma)^{2T}(P_0-b)^2) \\
=& \frac{\gamma}{2-\gamma}b(1-b) + b^2 - b^2 + (1-\gamma)^T\left(2bP_0 - 2b^2 - 2bP_0 + 2b^2\right) \\
&+ (1-\gamma)^{2T}\left(P_0^2 - 2bP_0 - \frac{\gamma}{2-\gamma}b(1-b) + b^2 - (P_0-b)^2\right) \\
=& \frac{\gamma}{2-\gamma}b(1-b) - (1-\gamma)^{2T}\frac{\gamma}{2-\gamma}b(1-b)
\end{aligned}$$

□

The following lemma provides closed form expressions for several expectations involving the EMA estimator.

Lemma 20. *For the EMA estimator* $P_T = \theta_\gamma(\vec{V})$

(1) $E_{B,\vec{V}}[2bP_T] = \frac{2}{3} + (1-\gamma)^T(P_0 - \frac{2}{3})$

(2) $E_{B,\vec{V}}[P_T^2] = \frac{4-\gamma}{6(2-\gamma)} + (1-\gamma)^T[P_0 - \frac{2}{3}] + (1-\gamma)^{2T}(P_0^2 - P_0 + \frac{4-3\gamma}{6(2-\gamma)})$

(3) $E_{B,\vec{V}}[2bP_T - P_T^2] = \frac{4-3\gamma}{6(2-\gamma)} - (1-\gamma)^{2T}(P_0^2 - P_0 + \frac{4-3\gamma}{6(2-\gamma)})$

Proof. Note that $E[P_T] = b + (1-\gamma)^T(P_0 - b)$, $E_B[b] = \frac{1}{2}$, and that $E_B[b^2] = \frac{1}{3}$. We therefore have

$$E_{B,\vec{V}}[2bP_T] = E_{b\sim B}[2b^2 - 2b^2(1-\gamma)^T + 2bP_0(1-\gamma)^T] = \frac{2}{3} + (1-\gamma)^T(P_0 - \frac{2}{3}) \ ,$$

proving (1). To prove (3), we use the derivation of $E[P_T^2]$ in the proof of Lemma 19 and average over $b \sim B$ to get

$$\begin{aligned} E_{B,\vec{V}}[2bP_T - P_T^2] &= \frac{2}{3} + (1-\gamma)^T(P_0 - \frac{2}{3}) - P_0^2(1-\gamma)^{2T} - P_0(1-\gamma)^T[1-(1-\gamma)^T] \\ &\quad - \frac{1}{3}(1-(1-\gamma)^T)^2 - \frac{\gamma}{6(2-\gamma)}(1-(1-\gamma)^{2T}) \\ &= \frac{4-3\gamma}{6(2-\gamma)} - (1-\gamma)^{2T}(P_0^2 - P_0 + \frac{4-3\gamma}{6(2-\gamma)}). \end{aligned}$$

Finally, since $E_{B,\vec{V}}[P_T^2] = E_{B,\vec{V}}[2bP_T] - E_{B,\vec{V}}[2bP_T - P_T^2]$, (2) follows immediately from (1) and (3) . □

We can now establishing performance bounds for the EMA estimator.

Theorem 21. *For the EMA estimator* $P_T = \theta_\gamma(\vec{V})$*, for any* $b \in [0,1]$*,*

(1) $MSE(b, P_T) \le \frac{\gamma}{4} + (1-\gamma)^{2T}$ *,*

(2) With probability at least $1-\delta$*, we have* $|P_T - b| \le (1-\gamma)^T + \sqrt{\frac{\gamma}{2}\ln\frac{1}{\delta}}$*, or equivalently,* $Pr(|P_T - b| > \epsilon) \le \exp(-2\frac{(\epsilon-(1-\gamma)^T)^2}{\gamma})$ *, and*

(3) For $P_0 = \frac{1}{2}$*,* $E_B[MSE(b, P_T)] = \frac{\gamma}{6(2-\gamma)} + (1-\gamma)^{2T}\frac{(2-3\gamma)}{12(2-\gamma)}$ *.*

Proof. To bound the MSE in (1), we first take the expectation of $P_T = \widehat{\theta}(\vec{V})$ w.r.t. $\vec{V}$ in (10.3) and rearranging we get $E(P_T) = b + (1-\gamma)^T(P_0 - b)$. We therefore have

$$\beta^2(P_T) = (E(P_T) - b)^2 = (1-\gamma)^{2T}(P_0 - b)^2 \le (1-\gamma)^{2T} \ .$$

Using Lemma 19, the bound (1) follows by maximizing the sum $MSE(b, P_T) =$

$\beta^2(P_T) + \sigma^2(P_T)$ over b. For the high probability bound (2) we first apply Lemma 16

$$\max_b Pr(|P_T - b| > \epsilon) \leq \max_b Pr\left(|P_T - E(P_T)| > \epsilon - (1-\gamma)^T\right) . \tag{10.9}$$

To further bound the right hand term of (10.9) above, we use McDiarmid's inequality (Lemma 15) as follows. Consider the definition (10.3) of $P_T = \theta_\gamma(\vec{V})$

$$P_T = P_0(1-\gamma)^T + \sum_{t=1}^{T} \gamma(1-\gamma)^{T-t} V_t .$$

Each signal V_t contributes to P_T at most $c_t \triangleq \gamma(1-\gamma)^{T-t}$. and we have

$$\sum_{t=1}^{T} c_t^2 = \gamma^2 \sum_{t=1}^{T} (1-\gamma)^{2(T-t)} = \gamma^2 \sum_{t=0}^{T-1} (1-\gamma)^{2i} = \frac{\gamma}{2-\gamma}[1-(1-\gamma)^{2T}] \leq \gamma$$

Therefore by Lemma 15) we have

$$Pr(|P_T - E(P_T)| > \epsilon) \leq e^{-2\epsilon^2/\sum_t c_t^2} \leq e^{-2\epsilon^2/\gamma},$$

and combining with (10.9) we get the desired bound,

$$\max_b Pr(|P_T - b| > \epsilon) \leq \max_b Pr\left(|P_T - E(P_T)| > \epsilon - (1-\gamma)^T\right) \leq e^{-2\left(\epsilon - (1-\gamma)^T\right)^2/\gamma} .$$

Finally, for the expectation (3) we use (10.7) again and apply Lemma 20 (giving a closed expressions for $E_{B,\vec{V}}[2bP_T - P_T^2]$ for EMA). □

The statistical consistency (MSE approaching 0 when $T \to \infty$) of the Bayes estimator is evident [6]. EMA estimators $\theta_\gamma(\cdot)$ are consistent for $\gamma = \omega\left(\frac{1}{T}\right)$ (which guarantees that $(1-\gamma)^T$ vanishes). Note also that if the agents know b (i.e., $P_T \equiv b$) then $MSE(b, P_T) = 0$ and the resulting expected score of the agents is $E_B[(b - P_0)^2] = \frac{1}{3}$ (assuming $P_0 = 1/2$. We may therefore interpret $\frac{1}{3} - E_{B,\vec{V}}[S_b(P_T)] = E_B[MSE(b, P_T)]$ as the penalty to the agents of the remaining uncertainty regarding b.

10.2.1 Non-Strategic Agents

We now turn to study the performance of the exponential moving average estimator $\theta_\gamma(\vec{V})$ for a specific value of γ, namely, the γ that maximizes the total score of the

[6] This also follows from Theorem 18 above

traders. This is not in equilibrium (we compute the equilibrium γ later, and the two are not the same). For a fixed $\gamma \in [0,1]$, the expected agent score is

$$\Phi_T(\gamma, b) \triangleq \frac{1}{T}\sum_{t=1}^{T} E\left(S_b(P_t) - S_b(P_{t-1})\right) = \frac{1}{T}E[S_b(P_T) - S_b(P_0)],$$

where the second equality follows by the linearity of expectation and the telescopic nature of the summation.

To find the γ maximizing the expected profit we can ignore the constants $\frac{1}{T}$ and $S_b(P_0)$ and then,

$$\gamma^*_{\max} \triangleq \arg\max_{\gamma} E_B \Phi_T(\gamma, b) = \arg\max_{\gamma} E_{B,\vec{V}} S_b(P_T) = \arg\max_{\gamma} E_{B,\vec{V}}(2bP_T - P_T^2),$$

where the last identity is from (9.4), and noting that the additive term of $1-b$ is independent of γ.

Theorem 22. *For the EMA estimator $P_T = \theta_\gamma(\vec{V}_{[1,T]})$, and $P_0 = 1/2$,*

$$\gamma^*_{\max} = \frac{\ln T}{2T} + \phi, \quad \textit{where } |\phi| \le \frac{2}{T}.$$

Proof. From Lemma 20 we have that for the EMA estimator,

$$a_T(\gamma) \triangleq E_{B,\vec{V}}[2bP_T - P_T^2] = \frac{4-3\gamma}{6(2-\gamma)} - (1-\gamma)^{2T}(P_0^2 - P_0 + \frac{4-3\gamma}{6(2-\gamma)}). \tag{10.10}$$

Now, Let $r(\gamma) \triangleq \frac{4-3\gamma}{6(2-\gamma)} = \frac{1}{2} - \frac{1}{3(2-\gamma)}$, we have

$$a_T(\gamma) = r(\gamma)[1-(1-\gamma)^{2T}] + (P_0 - P_0^2)(1-\gamma)^{2T}$$
$$a'_T(\gamma) = r'(\gamma)[1-(1-\gamma)^{2T}] + 2T(1-\gamma)^{2T-1}[r(\gamma) - (P_0 - P_0^2)]$$
$$a''_T(\gamma) = r''(\gamma)[1-(1-\gamma)^{2T}] + 4Tr'(\gamma)(1-\gamma)^{2T-1} - 2T(2T-1)(1-\gamma)^{2T-2}[r(\gamma) - (P_0 - P_0^2)]$$

Note that $r'(\gamma) = -\frac{1}{3(2-\gamma)^2}$ and $r''(\gamma) = -\frac{2}{3(2-\gamma)^3}$. For $\gamma \in [\frac{1}{2}, 1]$ we have

$$a'_T(\gamma) < -\frac{4}{27}(1-2^{-2T}) + 2T2^{-2T+1}(\frac{1}{6} - \frac{1}{4}) < 0\ ,$$

so the maximum value of $a_T(\gamma)$ is at $\gamma = \frac{1}{2}$ which is included in the next case. For $\gamma \in [0, \frac{1}{2}]$ we have $r(\gamma) - (P_0 - P_0^2) \ge r(\frac{1}{2}) - \frac{1}{4} = \frac{1}{36}$ and also $r(\gamma) - (P_0 - P_0^2) \le \frac{1}{3}$.

We consider two values of γ: $\gamma_+ = \frac{3+\ln T}{2T}$ and $\gamma_- = \frac{-3+\ln T}{2T}$.

We have that $(1-\gamma_+)^{2T} = \frac{1}{Te^3}$ and $(1-\gamma_-)^{2T} = \frac{e^3}{T}$. This implies that

$$a'_T(\gamma_+) = r'(\gamma_+)[1-(1-\gamma_+)^{2T}] + 2T(1-\gamma_+)^{2T-1}[r(\gamma_+) - (P_0 - P_0^2)] < -\frac{1}{12} + \frac{2}{3e^3} < 0.$$

Similarly,

$$a'_T(\gamma_-) = r'(\gamma_-)[1-(1-\gamma_-)^{2T}] + 2T(1-\gamma_-)^{2T-1}[r(\gamma_-) - (P_0 - P_0^2)] > -\frac{4}{27} + \frac{4e^3}{27T} + 2e^3\frac{1}{36} > 0$$

This implies that $\gamma^*_{\max} \in [\gamma_-, \gamma_+]$ □

Note that a similar proof applies to any value of $P_0 \in (0,1)$. It is worthwhile to compare the resulting $\gamma^*_{\max}$, maximizing agent score using the EMA estimator with that of the Bayesian estimator. In the Bayesian estimator the agent is aware of the history and knows his location in the permutation, and when his location is t he updates using $\gamma_{\text{bayes}} = \frac{1}{t+2}$. If we average over all the locations we have that the average update magnitude is $\frac{1}{T}\sum_{t=1}^{T}\frac{1}{t+2} \approx \frac{\ln T}{T}$. Note that this is only a factor of 2 larger than the resulting update maximizing total agents' profit[7] using EMA.

Now, based on Theorem 21, we derive the following performance of the EMA estimator $\theta_{\gamma^*_{\max}}(\vec{V})$:

Corollary 23. *For the EMA estimator $P_T = \theta_{\gamma^*_{\max}}(\vec{V})$, for any $b \in [0,1]$,*
(1) $MSE(b, P_T) = O(\frac{\ln T}{T})$,
(2) With probability at least $1-\delta$, we have $|P_T - b| = O(\sqrt{\frac{\log(T)\log(\frac{1}{\delta})}{T}})$, and
(3) For $P_0 = \frac{1}{2}$, $E_B[MSE(b, P_T)] = \frac{\ln T}{24T} - \frac{1}{12T} + O(\frac{1}{T^2})$.

Prrof Sketch. Note that $(1-\gamma^*_{\max})^{2T} \approx 1/T$, and that for $\gamma = \frac{\ln T}{2T}$ we have $(1-\gamma)^{2T} = \frac{1}{T}$. Also, for $\gamma = \frac{1}{\sqrt{2T}}$ we have $(1-\gamma)^{2T} = e^{-\sqrt{T}}$. Finally, the term $\frac{\gamma}{6(2-\gamma)}$ approaches $\frac{\gamma}{12}$ and the term $\frac{2-3\gamma}{12(2-\gamma)}$ approaches $\frac{1}{12}$. Plugging the above in Theorem 21 yields the corollary. □

We can contrast the bounds with those of the Bayes estimator $P_T = \widehat{\theta}(\vec{V})$. The MSE bound increased by a logarithmic factor $O(\ln T)$ (from $O(\frac{1}{T})$ to $O(\frac{\ln T}{T})$) and the high probability bound increases only by a factor of $O(\sqrt{\log T})$. This logarithmic increases show that the impact of the limitation of the updates to EMA is rather limited.

[7] Which by Claim 14 is the update minimizing estimator's MSE!

10.3 Strategic Agents

We consider the learning process with agents using the EMA estimator, $\theta_\gamma(\vec{V})$, where agents are strategic. *I.e.*, we seek a value of γ such that, for all t, given that agents $1, \ldots, t-1$ compute their prediction using EMA with parameter γ, then it is a best response for agent t to do likewise. Such a choice of γ gives a symmetric equilibrium for the more general setting where agent predictions use individual update parameters.

To find such a value of γ, let $\lambda(\gamma)$ be the best response of an agent, assuming that all other agents use update factor γ. (For brevity we will use λ, when clear from the context.) An agent arriving at time t will update the outstanding prediction P_{t-1} as follows

$$P_t(\gamma, \lambda) = (1-\lambda)P_{t-1}(\gamma) + \lambda V_t \ ,$$

where $P_{t-1}(\gamma) = P_{t-1}(\gamma, \gamma)$ assumes that the first $t-1$ agents update using θ_γ. Since the agent does not know her location, her expected score is,

$$u(\gamma, \lambda) = \frac{1}{T}\sum_{t=1}^{T} E_{B,\vec{V}}[S_b(P_t(\gamma, \lambda)) - S_b(P_{t-1}(\gamma))] \ .$$

Therefore, an agent maximizes her expected score, given that all other agents update using γ, by choosing the best response $\lambda^*(\gamma)$,

$$\lambda^*(\gamma) = \arg\max_{\lambda} u(\gamma, \lambda) \quad \text{and in equilibrium} \quad \lambda^*(\gamma) = \gamma \ . \tag{10.11}$$

An update parameter that achieves equilibrium is denoted γ^*_{eq}, *i.e.*, $\lambda^*(\gamma^*_{eq}) = \gamma^*_{eq}$. Note that we are assuming that the total number of agents T is known by the agents. Hence the utilities and updates defined above may all depend on T (e.g., $\gamma^*_{eq}(T)$) which is omitted from the notations for clarity when not needed explicitly. In Section 10.4.1 we discuss the extension to the case where the agents have only a prior distribution over the number of agents.

Aiming at the value of $\gamma^*_{eq}(T)$ we first derive a closed form for $\lambda^*(\gamma)$.

Lemma 24. *Let* $P_T = \theta_\gamma(\vec{V})$. *For* $P_0 = 1/2$,

$$\lambda^*(\gamma) = \frac{\gamma^2 T + (1-\gamma)^2(1-(1-\gamma)^{2T})a(\gamma)}{2\gamma T + (1-\gamma)^2(1-(1-\gamma)^{2T})a(\gamma)}$$

where $a(\gamma) = \frac{1}{2}(1 - \frac{2\gamma}{2-\gamma})$.

Proof. Recall that,

$$P_t(\gamma,\lambda) = (1-\lambda)P_{t-1}(\gamma) + \lambda V_t \ ,$$

For $\lambda^*(\gamma)$ to be a best response, we need to maximize

$$u(\gamma,\lambda) = \frac{1}{T}\sum_{t=1}^{T} E_{B,\vec{V}}[2bP_t(\gamma,\lambda) - P_t^2(\gamma,\lambda) - S_b(P_{t-1}(\gamma))] \ .$$

Now, Since $S_b(P_{t-1}(\gamma))$ does not depend on λ, we only need to maximize

$$\frac{1}{T}\sum_{t=1}^{T} E_{B,\vec{V}}[2bP_{t+1}(\gamma,\lambda)] - E_{B,\vec{V}}[P_{t+1}^2(\gamma,\lambda)] \ .$$

Taking the derivatives w.r.t. λ we have

$$\frac{d}{d\lambda}P_t(\gamma,\lambda) = V_t - P_{t-1}(\gamma),$$

and similarly,

$$\frac{d}{d\lambda}P_t^2(\gamma,\lambda) = 2P_t(\gamma,\lambda)\frac{d}{d\lambda}P_t(\gamma,\lambda) = 2P_t(\gamma,\lambda)(V_t - P_{t-1}(\gamma)).$$

Therefore, to maximize $u(\gamma,\lambda)$ we need that

$$LHS = \frac{1}{T}\sum_{t=1}^{T} E_{B,\vec{V}}\left[2b(V_t - P_{t-1}(\gamma))\right] = \frac{1}{T}\sum_{t=1}^{T} E_{B,\vec{V}}\left[2P_t(\gamma,\lambda)(V_t - P_{t-1}(\gamma))\right] = RHS \ .$$

For the LHS, taking the expectation first w.r.t. $\vec{V}$ and then w.r.t. B we have,

$$LHS = \frac{1}{T}\sum_{t=1}^{T} E_B[2b(b - b - (1-\gamma)^t(P_0 - b))] = \frac{-1}{T}(1-\gamma)\frac{1-(1-\gamma)^T}{\gamma}[P_0 - \frac{2}{3}] \ .$$

For the RHS we have

$$\begin{aligned}
RHS &= \frac{1}{T}\sum_{t=1}^{T} E_{B,\vec{V}}[2P_{t+1}(\gamma,\lambda)(V_{t+1}-P_t(\gamma))] \\
&= \frac{2}{T}\sum_{t=1}^{T} E_{B,\vec{V}}\left[((1-\lambda)P_t(\gamma)+\lambda V_{t+1})(V_{t+1}-P_t(\gamma))\right] \\
&= \frac{2}{T}\sum_{t=1}^{T} E_{B,\vec{V}}\left[(1-\lambda)P_t(\gamma)V_{t+1}-(1-\lambda)P_t^2(\gamma)+\lambda V_{t+1}^2-\lambda V_{t+1}P_t(\gamma)\right] \\
&= \frac{2}{T}\sum_{t=1}^{T}(1-2\lambda)E_{B,\vec{V}}\left[P_t(\gamma)V_{t+1}\right]-(1-\lambda)E_{\vec{V},B}\left[P_t^2(\gamma)\right]+\lambda E_{B,\vec{V}}\left[V_{t+1}^2\right] \ .
\end{aligned}$$

We now compute each of the expectations:

$$\begin{aligned}
\frac{2}{T}\sum_{t=1}^{T} E_{\vec{V},B}\left[V_{t+1}^2\right] =& \frac{2}{T}\sum_{t=1}^{T} E_B[b] = 1; \\
\frac{2}{T}\sum_{t=1}^{T} E_{\vec{V},B}\left[V_{t+1}P_t(\gamma)\right] =& \frac{2}{T}\sum_{t=1}^{T} E_B[b(b+(1-\gamma)^t(P_0-b))] \\
&= \frac{2}{3}+(1-\gamma)\left(P_0-\frac{2}{3}\right)\frac{1-(1-\gamma)^T}{\gamma T}; \\
\frac{2}{T}\sum_{t=1}^{T} E_{\vec{V},B}\left[P_t^2(\gamma)\right] =& \frac{2}{T}\sum_{t=1}^{T}\frac{4-\gamma}{6(2-\gamma)} \\
&+(1-\gamma)^t\left(P_0-\frac{2}{3}\right)+(1-\gamma)^{2t}\left(P_0^2-P_0+\frac{4-3\gamma}{6(2-\gamma)}\right) \\
=& \frac{4-\gamma}{3(2-\gamma)}+2(1-\gamma)\frac{1-(1-\gamma)^T}{\gamma T}\left(P_0-\frac{2}{3}\right) \\
&+2(1-\gamma)^2\frac{1-(1-\gamma)^{2T}}{(2-\gamma)\gamma T}\left(P_0^2-P_0+\frac{4-3\gamma}{6(2-\gamma)}\right) \ .
\end{aligned}$$

Then we have

$$\begin{aligned}
RHS =& (1-2\lambda)\left(\frac{2}{3}+(1-\gamma)\left(P_0-\frac{2}{3}\right)\frac{1-(1-\gamma)^T}{\gamma T}\right) \\
&-(1-\lambda)\left(\frac{4-\gamma}{3(2-\gamma)}+2(1-\gamma)\frac{1-(1-\gamma)^T}{\gamma T}\left(P_0-\frac{2}{3}\right)\right) \\
&-2(1-\lambda)(1-\gamma)^2\frac{1-(1-\gamma)^{2T}}{(2-\gamma)\gamma T}\left(P_0^2-P_0+\frac{4-3\gamma}{6(2-\gamma)}\right) \\
&+\lambda \ .
\end{aligned}$$

Putting it all together we get the equation

$$\frac{\gamma}{3(2-\gamma)} - (1-\gamma)\frac{1-(1-\gamma)^T}{\gamma T}\left(P_0 - \frac{2}{3}\right) = \lambda\frac{2}{3(2-\gamma)} - (1-\gamma)\frac{1-(1-\gamma)^T}{\gamma T}\left(P_0 - \frac{2}{3}\right) - 2(1-\lambda)(1-\gamma)^2\frac{1-(1-\gamma)^{2T}}{(2-\gamma)\gamma T}\left(P_0^2 - P_0 + \frac{4-3\gamma}{6(2-\gamma)}\right).$$

Simplifying, we conclude that $\lambda(\gamma)$ is maximized only if

$$\frac{\gamma}{3} = \lambda\frac{2}{3} - 2(1-\lambda)(1-\gamma)^2\frac{1-(1-\gamma)^{2T}}{\gamma T}\left(P_0^2 - P_0 + \frac{4-3\gamma}{6(2-\gamma)}\right). \tag{10.12}$$

Setting $P_0 = \frac{1}{2}$ in (10.12) and rearranging we get

$$\lambda^*(\gamma) = \frac{\gamma^2 T + (1-\gamma)^2(1-(1-\gamma)^{2T})a(\gamma)}{2\gamma T + (1-\gamma)^2(1-(1-\gamma)^{2T})a(\gamma)} \tag{10.13}$$

where $a(\gamma) = \frac{1}{2}(1 - \frac{2\gamma}{2-\gamma})$. □

The following theorem derives the value of $\gamma^*_{eq}(T)$ and shows that it gives the unique symmetric equilibrium.

Theorem 25. *Let $P_0 = \frac{1}{2}$ and let γ^*_{eq} be the update parameter in a symmetric equilibrium, then*

$$\gamma^*_{eq}(T) = \sqrt{\frac{1}{2T}} - \beta \quad , \quad \text{and } \beta \in [0, \frac{6}{T}] \ ,$$

and this is the unique symmetric equilibrium.

Proof. By Lemma 24, since in equilibrium we have $\lambda^*(\gamma) = \gamma$, we need

$$\gamma = \frac{\gamma^2 T + (1-\gamma)^2(1-(1-\gamma)^{2T})a(\gamma)}{2\gamma T + (1-\gamma)^2(1-(1-\gamma)^{2T})a(\gamma)}.$$

Reorganizing we get

$$\gamma^2 T = (1-\gamma)^3(1-(1-\gamma)^{2T})\frac{1}{2}\left(1 - \frac{2\gamma}{2-\gamma}\right). \tag{10.14}$$

Now, For $\gamma < \frac{\ln T}{T}$, the right hand side of (10.14) (RHS) is $O(1)$ whereas the left hand side (LHS) is $O(\frac{\log^2 T}{T})$. For $\gamma > \frac{\ln T}{T}$, the LHS increases with γ while the RHS decreases with γ therefore there is a unique equilibrium.

First, we show that for $\gamma = \frac{1}{\sqrt{2T}}$ the LHS of (10.14) is larger than the RHS.

$$\frac{1}{2T}T = \frac{1}{2} > \frac{1}{2}(1-\gamma)^3(1-(1-\gamma)^{2T})\left(1-\frac{2\gamma}{2-\gamma}\right)$$

since the three rightmost terms of the RHS above are less than 1. Second, we show that for $\gamma = \frac{1-\epsilon}{\sqrt{2T}}$ the LHS is smaller, for $\epsilon > \frac{12}{\sqrt{2T}}$. In this case, for the LHS of (10.14) we have,

$$\frac{(1-\epsilon)^2}{2T}T = \frac{(1-\epsilon)^2}{2}.$$

For the RHS of (10.14) we have,

$$\frac{1}{2}(1-\frac{1-\epsilon}{\sqrt{2T}})^3\left(1-\left(1-\frac{1-\epsilon}{\sqrt{2T}}\right)^{2T}\right)\left(1-\frac{2\frac{1-\epsilon}{\sqrt{2T}}}{2-\frac{1-\epsilon}{\sqrt{2T}}}\right)$$
$$> \frac{1}{2}\left(1-3\frac{1-\epsilon}{\sqrt{2T}}\right)\cdot\left(1-\frac{1-\epsilon}{\sqrt{2T}}\right)\left(1-2\frac{1-\epsilon}{\sqrt{2T}}\right) > \frac{1}{2}\left(1-12\frac{1-\epsilon}{\sqrt{2T}}\right).$$

So we need that

$$(1-\epsilon)^2 < 1-\epsilon < 1-12\frac{1-\epsilon}{\sqrt{2T}}$$

which holds for $\epsilon > \frac{12}{\sqrt{2T}}$. □

We now revisit the EMA estimator $P_T = \theta_\gamma(\vec{V})$ performance for the equilibrium update $\gamma^*_{eq} = \frac{1}{\sqrt{2T}}$. As in the case for non-strategic agents, note that $(1-\gamma^*_{eq})^T \approx e^{-\sqrt{\frac{T}{2}}}$, and calculating similarly to the proof of Corollary 23, we derive the following corollary of Theorem 21 for the case $P_T = \theta_{\gamma^*_{eq}}(\vec{V})$

Corollary 26. *For the EMA estimator $P_T = \theta_{\gamma^*_{eq}}(\vec{V})$, for any $b \in [0,1]$,*
(1) $MSE(b, P_T) = O(\frac{1}{\sqrt{T}})$,
(2) With probability at least $1-\delta$, we have $|P_T - b| = O(T^{-1/4}\sqrt{\log(T)\log(\frac{1}{\delta})})$, and
(3) For $P_0 = \frac{1}{2}$, $E_B[MSE(b, P_T)] = \frac{1}{12\sqrt{2T}-6} + O(e^{-\sqrt{2T}})$.

Comparing the bounds above with those of the Bayes estimator $\widehat{\theta}(\vec{V})$ and with the exponential moving average $\theta_{\gamma^*_{\max}}(\vec{V})$. Both $\widehat{\theta}(\vec{V})$ and $\theta_{\gamma^*_{\max}}(\vec{V})$ achieve a mean square error of $\tilde{O}(\frac{1}{T})$ *vs.* $O(\frac{1}{\sqrt{T}})$ for the symmetric equilibrium. For the high probability bound (2) the gap is between $\tilde{O}(\frac{1}{\sqrt{T}})$ and $\tilde{O}(\frac{1}{T^{1/4}})$. This is both good news and bad news. The good news is the process converges to the true probabilities even when agents are unaware of the trading history (and use EMA updates). The bad news is that the convergence rate deteriorates due to selfish strategic behavior.

10.4 Extensions

Three separate extensions to the basic setting are considered. Analysis of the equilibrium update is presented for the case where the total number of agents is unknown and for a setting in which all agents except one are fully informed (that is, have direct access to all the signals). Finally, the utility gain for a single aware agent is analyzed.

10.4.1 Distribution Over the Number of Agents

An interesting extension is to assume further uncertainty, where even the total number of agents, T, is unknown. It may be unrealistic to forecast the number of agents. A more reasonable assumption may be a common prior over the number of agents. The obvious question is how this additional uncertainty impacts our results. So, we want to compute the symmetric equilibrium in this setting.

Theorem 27. *For strategic agents that know neither their position in line, nor the total number of agents, but share a prior on the total number of agents with* $E[\frac{1}{T}] \leq \frac{1}{8}$*, the equilibrium update is* $\gamma^*_{dist} = \Theta(\sqrt{E(\frac{1}{T})})$.

Proof. We first establish the following condition for the update γ^*_{dist} of strategic agents at equilibrium.

$$(\gamma^*_{dist})^2 = 6(1-\gamma^*_{dist})^3 E_T[\frac{1-(1-\gamma)^{2T}}{T}]\rho(P_0, \gamma^*_{dist}) \ , \tag{10.15}$$

where $\rho(P_0,\gamma) \triangleq (P_0 - \frac{1}{2})^2 + \frac{2-3\gamma}{12(2-\gamma)}) \in (0, \frac{1}{3})$. Repeating the derivation of Eq. (10.12) from the proof of Lemma 24, we have that γ^*_{dist} must satisfy

$$\gamma = 2\lambda - 6(1-\lambda)(1-\gamma)^2(P_0^2 - P_0 + \frac{4-3\gamma}{6(2-\gamma)})E_T[\frac{1-(1-\gamma)^{2T}}{\gamma T}] \tag{10.16}$$

Since at equilibrium $\lambda = \gamma$ we must have that the following expression is positive,

$$6(1-\lambda)(1-\gamma)^2(P_0^2 - P_0 + \frac{4-3\gamma}{6(2-\gamma)})E_T[\frac{1-(1-\gamma)^{2T}}{\gamma T}]$$

Now, since all the terms of the above expression except $(P_0^2 - P_0 + \frac{4-3\gamma}{6(2-\gamma)})$ are guaranteed to be positive, we must have $\rho(P_0,\gamma) = P_0^2 - P_0 + \frac{4-3\gamma}{6(2-\gamma)} > 0$. Now, at equilibrium, setting $\lambda = \gamma = \gamma^*_{dist}$ in (10.16), and simplifying we get the condition (10.15). We

therefore have

$$(\gamma^*_{dist})^2 \leq 2E_T[\frac{1-(1-\gamma)^{2T}}{T}] = \Theta(E_T[\frac{1}{T}]) \ .$$

We now need to show that γ^*_{dist} is not too small. First note that by the above calculations we can bound $(\gamma^*_{dist})^2 \leq 2E[\frac{1}{T}] \leq \frac{1}{4}$. Therefore, $\gamma^*_{dist} \leq \frac{1}{2}$ and we have $\rho(P_0, \gamma^*_{dist}) \geq \frac{1}{36}$. Substituting in (10.15) we get

$$(\gamma^*_{dist})^2 \geq \frac{6}{8 \cdot 36} E_T[\frac{1-(1-\gamma)^{2T}}{T}] = \Theta(E_T[\frac{1}{T}]) \ .$$

□

Note that by the Jensen Inequality,[8] the condition $E[\frac{1}{T}] \leq \frac{1}{8}$ implies $E[T] \geq 8$. Namely, we are dealing with situations in which the agents assume there are not too few of them (in expectation). Note also that the resulting equilibrium update parameters is $\Theta(\sqrt{E(\frac{1}{T})})$, which is different from $\Theta(E(\frac{1}{\sqrt{T}}))$. Conceptually, this is very good news. Recall that the Bayes update would have mean square error equal $\Theta(E(1/T))$. This implies that the EMA equilibrium update γ^*_{dist}, which is only square-root of that quantity, has a mean square error of $\Theta(\sqrt{E(\frac{1}{T})})$, assuming that for $\alpha = \sqrt{E(\frac{1}{T})}$ we have $E[Te^{-\alpha T}] = O(\sqrt{E(\frac{1}{T})})$. This establishes the following corollary to Theorem 21 part (1), for $\gamma^*_{dist} = \Theta(\sqrt{E(\frac{1}{T})})$.

Corollary 28. *If the Bayes estimator MSE is bounded by ϵ then for agents in equiliria, the mean square error is at most $O(\sqrt{\epsilon} + E[Te^{-\epsilon T}])$.*

10.4.2 Single Unaware Agent

Assume all agents do the correct (fully informed, Bayesian) update $\hat{\theta}(\cdot)$, except for one agent which is not aware of the history and his location. Such a setting assesses the penalty of an agent not knowing its location. Alternatively, this measures the maximum price that such an agent would be willing to pay to gain the information, in the extreme case that all other agents know their location. One can view the unaware agent as a late adopter of a technology that determines an agent's location, and we compute the penalty associated with this late adaptation.

Technically, this implies that when the unaware agent arrives at the process, the price is set by the Bayesian update $\hat{\theta}(\vec{v})$. We now compute the γ that maximizes the

[8] Stating that $E[f(X)] \leq f(E[X])$ for a convex function f of a random variable X.

agent's score. Let t be the unaware agent, then we have,

$$P_t^\gamma = (1-\gamma)P_{t-1}^B + \gamma V_t \quad \text{and} \quad P_{t-1}^B = \frac{1+\sum_{i=1}^{t-1} V_i}{t+1} \ ,$$

where $P_0 = \frac{1}{2}$. The average profit of the unaware trader, assuming a uniform distribution over his arrival $t \in \{1, \cdots, T\}$, is,

$$\frac{1}{T}\sum_{t=1}^{T} E_{B,\vec{V}}[S_b(P_t^\gamma) - S_b(P_{t-1}^B)].$$

The following theorem establishes the optimal update parameter.

Theorem 29. *The optimal update parameter γ_1^* for a single unaware trader model, when $P_0 = \frac{1}{2}$, is*

$$\gamma_1^* = \frac{\ln T}{T + \ln T}$$

Proof. First we compute

$$\begin{aligned}
E_{B,\vec{V}}[2bP_t^\gamma] =& E_{B,\vec{V}}[2b(1-\lambda)P_{t-1}^B + 2b\lambda V_t] \\
=& E_{B,\vec{V}}[2b(1-\lambda)\frac{1+b(t-1)}{t+1} + 2b\lambda V_t] \\
=& 2(1-\lambda)E_B[\frac{b+b^2(t-1)}{t+1}] + 2\lambda E_B[b^2] \\
=& 2(1-\lambda)\frac{(1/2)+(t-1)/3}{t+1} + \frac{2}{3}\lambda = \frac{2}{3} - \frac{1-\lambda}{3(t+1)}
\end{aligned}$$

Summing over t,

$$\sum_{t=1}^{T} E_{B,\vec{V}}[2bP_t^\gamma] = \sum_{t=1}^{T} \frac{2}{3} - \frac{1-\lambda}{3(t+1)} \approx \frac{2T}{3} - \frac{(1-\lambda)\ln T}{3}$$

The expected second moment is,

$$\begin{aligned}
E_{B,\vec{V}}[(P_t^\gamma)^2] =& E_{B,\vec{V}}[(1-\lambda)^2(P_{t-1}^B)^2 + \lambda^2 V_t^2 + 2\lambda(1-\lambda)V_t P_{t-1}^B] \\
=& (1-\lambda)^2 E_{B,\vec{V}}[(P_{t-1}^B)^2] + \lambda^2 E_{B,\vec{V}}[V_t^2] + 2\lambda(1-\lambda)E_{B,\vec{V}}[V_t P_{t-1}^B]
\end{aligned}$$

We now compute each of the terms above:

$$\begin{aligned}
E_{B,\vec{V}}[V_t^2] =& E_B[b] = \frac{1}{2} \\
E_{B,\vec{V}}[V_t P_{t-1}^B] =& E_{B,\vec{V}}[V_t \frac{1+\sum_{i=1}^{t-1} V_i}{t+1}] = E_B[b\frac{1+b(t-1)}{t+1}] = \frac{\frac{1}{2}+\frac{1}{3}(t-1)}{t+1} = \frac{1}{3} - \frac{1}{6(t+1)} \\
E_{B,\vec{V}}[(P_{t-1}^B)^2] =& E_{B,\vec{V}}[\frac{(1+\sum_{i=1}^{t-1} V_i)^2}{(t+1)^2}] = \frac{1}{(t+1)^2} E_{B,\vec{V}}[1 + 2(\sum_{i=1}^{t-1} V_i) + (\sum_{i=1}^{t-1} V_i)^2] \\
=& \frac{1}{(t+1)^2} E_B[1 + 3b(t-1) + (t-1)(t-2)b^2] = \frac{1 + \frac{3}{2}(t-1) + \frac{1}{3}(t-1)(t-2)}{(t+1)^2} \\
=& \frac{1}{3} - \frac{1}{6(t+1)}
\end{aligned}$$

Substituting the terms and summing over t,

$$\begin{aligned}
\sum_{t=1}^{T} E_{B,\vec{V}}[(P_t^\gamma)^2] =& \sum_{t=1}^{T} \frac{\lambda^2}{2} + \left(\frac{1}{3} - \frac{1}{6(t+1)}\right)\left((1-\lambda)^2 + 2\lambda(1-\lambda)\right) \\
=& \sum_{t=1}^{T} \frac{\lambda^2}{2} + \left(\frac{1}{3} - \frac{1}{6(t+1)}\right)[1-\lambda^2] \\
\approx& \frac{\lambda^2 T}{2} + \left(\frac{T}{3} - \frac{\ln T}{6}\right)(1-\lambda^2)
\end{aligned}$$

We can now compute the expected payoff

$$U = \sum_{t=1}^{T} E_{B,\vec{V}}[2bP_t^\gamma - (P_t^\gamma)^2] = \frac{2T}{3} - \frac{1-\lambda}{3}\ln T - \frac{\lambda^2 T}{2} - \left(\frac{T}{3} - \frac{\ln T}{6}\right)(1-\lambda^2)$$

The derivative is,

$$U' = \frac{\ln T}{3} - \lambda T + 2\lambda\left(\frac{T}{3} - \frac{\ln T}{6}\right)$$

and requiring $U' = 0$ we get that the maximum[9] is achieved at $\lambda = \frac{\ln T}{T + \ln T}$ □

Recall that when none of the agents are informed, the utility maximizing update parameter is $\frac{\ln T}{2T}$ while if all the agents are informed and use Bayes updates then the average update parameter is $\frac{\ln T}{T}$. The update parameter above is a small step from the update of all informed Bayesian traders to all uninformed EMA traders.

[9] Note that $U'' < 0$.

10.4.3 Single Aware Agent

This setting can be seen as the flip-side of the previous setting. Here we consider the case that only a single agent is informed regarding his location. This models the benefit that a trader can gain by being able to access his location. One way of gaining the information is through buying it exclusively, the utility gain bounds the price the agent would be willing to pay for such an information.

Technically, assume a single agent doing the correct (fully informed, Bayesian) update $\hat{\theta}(\cdot)$, and all other agents (not aware of their location) are restricted to use EMA θ_γ strategy and are either unaware or ignore the fact that a single agent is using a different strategy. We define

$$P^\gamma_{t-1} = (1-\gamma)P^\gamma_{t-2} + \gamma V_{t-1} \quad \text{and} \quad P^B_t = \frac{1+\sum_{i=1}^t V_i}{t+2} \ .$$

When the single aware agent is in location t his expected utility is $E_{B,\vec{V}}[S_b(P^B_t) - S_b(P^\gamma_{t-1})]$. We consider the average expected utility of the single aware agent, denoted $u_T(\gamma, \text{Bayes})$, is,

$$u_T(\gamma, \text{Bayes}) = \frac{1}{T}\sum_{t=1}^{T} E_{B,\vec{V}}[S_b(P^B_t) - S_b(P^\gamma_{t-1})].$$

Theorem 30. *For $\gamma = \frac{\ln T}{2T}$ we have $u_T(\frac{\ln T}{2T}, \text{Bayes}) = \Theta(\frac{\ln^2 T}{T})$ and for $\gamma = \frac{1}{\sqrt{2T}}$ we have $u_T(\frac{1}{\sqrt{2T}}, \text{Bayes}) = \Theta(\frac{1}{\sqrt{T}})$.*

Proof. Let $u_T(\text{Bayes})$ be the average expected utility of the agents using Bayesian updates, i.e.,

$$u_T(\text{Bayes}) = \frac{1}{T}\sum_{t=1}^{T} C^B(b, P_0, P_T) = (b-P_0)^2 - \frac{1}{T}\sum_{t=1}^{T} \text{MSE}(b, P^B_t)$$

where the first equality follows since the gain of the Bayesian agents is identical to the cost to the market maker, and the second equality follows from Claim 14.

Similarly, let $u_T(\gamma)$ be the average expected utility of the agents using EMA $\theta\gamma(\vec{V})$ updates, i.e.,

$$u_T(\gamma) = \frac{1}{T}\sum_{t=1}^{T} C^\gamma(b, P_0, P_T) = (b-P_0)^2 - \frac{1}{T}\sum_{t=1}^{T} \text{MSE}(b, P^\gamma_t),$$

where again, the first equality follows since the gain of the Bayesian agents is identical to the cost to the market maker, and the second equality follows from Claim 14.

We first show

$$u_T(\gamma, \text{Bayes}) = u_T(\text{Bayes}) - u_{T-1}(\gamma).$$

Let t be the location of the aware trader.

$$\begin{aligned}
u_T(\gamma, \text{Bayes}) =& \frac{1}{T}\sum_{t=1}^{T} E_{B,\vec{V}}[S_b(P_t^B) - S_b(P_{t-1}^\gamma)] \\
=& \frac{1}{T}\sum_{t=1}^{T} E_{B,\vec{V}}[S_b(P_t^B) - S_b(P_0)] + \frac{1}{T}\sum_{t=1}^{T} E_{B,\vec{V}}[S_b(P_0) - S_b(P_{t-1}^\gamma)] \\
=& u_T(\text{Bayes}) + \frac{1}{T}\sum_{t=1}^{T-1} E_{B,\vec{V}}[S_b(P_0) - S_b(P_t^\gamma)] \\
=& u_T(\text{Bayes}) - u_T(\gamma)
\end{aligned}$$

Using the identities for $u_T(\text{Bayes})$ and $u_T(\gamma)$ we have,

$$u_T(\gamma, \text{Bayes}) = \frac{1}{T}\sum_{t=1}^{T} \text{MSE}(b, P_t^\gamma) - \text{MSE}(b, P_t^B).$$

By Theorem 18 we have $\text{MSE}(b, P_t^B) = \Theta(\frac{1}{t})$, and hence $\frac{1}{T}\sum_{t=1}^{T} \text{MSE}(b, P_t^B) = O(\frac{\ln T}{T})$. By Theorem 21 we have $\text{MSE}(b, P_t^\gamma) = \Theta(\frac{\ln t}{t})$ for $\gamma = \frac{\ln T}{2T}$ and $\frac{1}{T}\sum_{t=1}^{T} \text{MSE}(b, P_t^\gamma) = \Theta(\frac{\ln^2 T}{T})$. Similarly, $\text{MSE}(b, P_t^\gamma) = \theta(\frac{1}{\sqrt{t}})$ for $\gamma = \frac{1}{\sqrt{2T}}$ and $\frac{1}{T}\sum_{t=1}^{T} \text{MSE}(b, P_t^\gamma) = \Theta(\frac{1}{\sqrt{T}})$. Combining the two bounds derives the theorem. □

When all agents are symmetric then the utility of an individual agent is $\Theta(\frac{1}{T})$, since the total utility of all the agents is constant. It follows from the theorem above that the utility to a single aware agent is significantly higher. Thus, the value of knowing the position is (about) $\frac{1}{\sqrt{T}}$.

10.5 Summary and Discussion

The main results are further discussed in this section, followed by a short review of related models and future reserach direction.

Update process	Worst case Mean Square Error: $\max_b \text{MSE}(b, P_T)$	Expected Mean Square Error: $E_B\text{MSE}(b, P_T)$ B uniform in $[0,1]$	Guaranteed accuracy for confidence $1-\delta$: $\epsilon(\delta) = \text{argmin}_\epsilon \max_b$ $\text{Prob}(\lvert P_T - b\rvert < \epsilon)$ $\geq 1-\delta$
Bayes	$\frac{1}{4(T+2)}$	$\frac{1}{6(T+2)}$	$\frac{1}{T+2} + \sqrt{\frac{\ln\frac{1}{\delta}}{2T}}$
Exp. Moving Averages:			
Symmetric Equilibrium $\gamma = 1/\sqrt{2T}$	$\frac{1}{4\sqrt{2T}} + e^{-\sqrt{2T}}$	$\frac{1}{12\sqrt{2T}-6} + O\left(e^{-\sqrt{2T}}\right)$	$O\left(\sqrt{\frac{\log(T)\log(1/\delta)}{\sqrt{T}}}\right)$
Socially Optimal $\gamma = \left(\frac{\ln T}{2T}\right)$	$\frac{\ln T}{8T} + \frac{1}{T}$	$\frac{\ln T}{24T} - \frac{1}{12T} + O\left(\frac{1}{T^2}\right)$	$O\left(\sqrt{\frac{\log(T)\log(1/\delta)}{T}}\right)$
Arbitrary γ	$\frac{\gamma}{4} + (1-\gamma)^{2T}$	$\frac{\gamma}{6(2-\gamma)} + (1-\gamma)^{2T}\frac{(2-3\gamma)}{12(2-\gamma)}$	$\sqrt{\frac{\gamma}{2}\ln\frac{1}{\delta}} + (1-\gamma)^T$

Figure 10.1: Performance metrics for Exponential Moving Average (EMA) method versus Bayes optimal benchmark for estimating (P_T) the unknown bias b of a coin. T is the total number of updates and γ is the averaging constant of EMA. The mean squared error (MSE) in the second column is over realizations of the signals $V_1, \ldots, V_T$. The expectation in the third column is assuming a uniform prior of $b \in [0,1]$ and initial prediction $P_0 = \frac{1}{2}$. The values in this table come from Theorems 18,21,22, and 25.

10.5.1 Main Results

We considered three measures of quality for a predictor:

1. The worst case mean square error (for any choice of $b \in [0,1]$). (Column 2 of Figure 10.1).

2. The expected mean square error (where b is uniformly chosen in $[0,1]$ before the signals are generated). (Column 3 of Figure 10.1).

3. The guaranteed accuracy for a given level of confidence. (Column 4 of Figure 10.1).

The rows of Figure 10.1 are for Bayes updates and various values of γ for exponential moving averages.

- Bayes update, this is equivalent to agents knowing the complete history, namely,

Bayes	$\widehat{\theta}(\vec{V}) = \frac{\sum_{t=1}^{T} V_t + 1}{T+2}$.
Exponential Moving Average	$\theta_\gamma(\vec{V}) = \frac{(1-\gamma)^T}{2} + \gamma \sum_{t=1}^{T} (1-\gamma)^{T-t} V_t$.
Computing Bayes Predictions Incrementally	$P_t = \left(1 - \frac{1}{t+2}\right) P_{t-1} + \frac{1}{t+2} V_t$.
Computing Exponential Moving Averages Incrementally	$P_t = (1-\gamma) P_{t-1} + \gamma V_t$.

Figure 10.2: Bayes Update *vs.* Exponential Moving Average, uniform $[0, 1]$ prior, $p_0 = 1/2$, $\vec{V} = V_1, \ldots, V_T$, where V_t is the private signal to agent t. Agent t is the t'th agent to act. Note the similarity of rows 3 and 4. Agents that do not know their index cannot do incremental Bayes update.

the count and values of previous updates performed.

- Updates performed via exponential moving averages, in two settings:

  1. Agents are strategic, and γ $(= \frac{1}{\sqrt{2T}}$ — see Theorem 25) is a symmetric equilibrium.

  2. Choosing γ $(= \frac{\ln T}{2T}$ — see Theorem 22) so as to maximize the social score. By Claim 14, This is equivalent to choosing γ to minimize the mean square error.

  We also give the general form of the estimators performance metrics for any value of γ (last row of Figure 10.1).

In Lemma 12 we prove that the sum of the agents' net score (that is, the cost of subsidizing the whole process) and the Bregman loss function (e.g., the mean square error, when the quadratic scoring rule is used) always sums to a constant. This implies that an extra expenditure of x as a subsidy to the information market means a reduction of the same amount in the mean square error expected. Thus, one can view such processes as information markets where the market makers pay (in subsidies) in return for quality of information (reduction in the mean square error, or other Bregman loss

function).

Now, comparing Row 1 (Bayes estimator) of Figure 10.1 with Row 3 (max profit possible with exponential moving averages) suggests that agents lose little by restricting their strategy space to exponentially moving averaging. Their total profit cannot exceed that obtained by the Bayes estimator, and they are very close to this upper bound. *I.e.*, the difference is $O(\frac{\ln T}{T})$. This loss (in total agents score) is the consequence of agents being unaware of the historical trades and using an exponential moving average estimator.

Now, comparing Row 2 (symmetric equilibrium using exponential moving averages) of Figure 10.1 with Row 1 (Bayes estimator) shows that the mean square error increases by a $\Theta(\sqrt{T})$ factor and that the error probability for a given confidence increases by a $\Theta(T^{-\frac{1}{4}})$ factor. Note that the mean square error still vanishes at a polynomial rate (in T), and that, for any constant accuracy, the error probability remains exponentially small.

Subsequently, we dealt with several extensions that include the following:

1. The number of traders T is unknown to the agents but is sampled from a known prior distribution. We show that the mean square error, for strategic agents, in equilibrium is $\sqrt{\epsilon}$ where ϵ is the mean square error for the Bayes estimator.

2. We further consider two related settings:

   - All but one agent do Bayesian updates, what should the outlier do? Intuitively, she should not value her signal too highly. The correct choice is $\gamma \approx \frac{\ln T}{T}$.
   - All agents do exponential moving average updates (with $\gamma = \frac{1}{\sqrt{2T}}$), but one agent knows her position in line. What should she do? What is the value to the trader to know her position in line? Here, the correct choice is $\gamma \approx \frac{1}{\sqrt{T}}$.

10.5.2 Related Models

Our setting can be casted[10] in the model of partial information presented by [17]. The issues studied in [17, 58, 87] are how communication leads agents to revise their posteriors until they converge, given that the agents have common priors.

[10] In a somewhat non-standard use of the Aumann's model, because there are aspects of the state of the world that are not interesting in and of themselves, whereas in our setting agents are only interested in the underlying probability of the event occurring.

Other work discussing aspects of information aggregation among agents having private information differ by the nature of the information to get aggregated. In [76], addressing the impact of information on pricing, informed traders (insiders) and "noise" traders are modeled, and prices are set by a market maker according to the aggregated demand. The informed trader, aware of the impact of trades on prices, behaves strategically, thus, trades so that private information is incorporated into the market slowly in order to maximize utility. Eventual convergence in probability of an information market using market scoring rules is investigated in [94]. Their model, however, assumes that all trades are public knowledge, and does not consider uncertainty regarding historical trades. Moreover, the rate of convergence is not quantified. Convergence in information markets with incomplete information is also considered by [54]. They consider a version of Shapley-Shubik market games[11] [112] where, in multiple rounds, traders simultaneously submit bids, and market clearing prices are computed and revealed. This repeats until convergence (if it converges, as characterized by [54]). Agents in this model, however, are not utility maximizers (the only goal is to compute some function), and therefore, contrary to our model, there is no notion of a strategic equilibrium.

10.5.3 Closing Remarks and Future Work

Inherently, in information markets, agents have private signals and trade options (or make predictions) based upon incomplete information. Knowing the history of past actions makes a difference because it reveals much about the signals of other participants, under appropriate assumptions (See [54], [78]). Our setting relaxes this assumption of available history, and therefore also fits scenarios in which this information may simply be unavailable.

Note that an agent having access to the actions history could use her own signal and compute the optimal (Bayes) estimate for the bias b. The Bayes estimator is the best possible in the sense that it minimizes the expected loss of *any* Bregman loss function, see [20].

Let alone the whole history, even just the timing of the action (that is, the "position in line") is critical. consider the first and last agents to act. The first to act associates little value to the initial prediction (based entirely upon the common prior),

[11] In a somewhat unusual setting, where Bayesian traders indicate how much they are willing to spend, but not how much they value the security.

and accordingly values her own signal highly. Contra-wise, the last to trade has good reason to assume that the current prediction (imperfectly) mirrors the wisdom of the crowd and that her signal has much less importance. This reasoning is irrelevant in our setting, where agents don't know how many agents acted before them. Strategic agents would decide on the their action based on their "belief " on the way other agents perform their updates, hence suggesting our analysis of equilibrium strategies for utility maximizing agents.

Our analysis of history-independent social learning settings was mostly based on the quadratic scoring rule. This raises the question regarding the socially optimal and equilibrium updates resulting in settings where different scoring rules are used, and any qualitative and quantitative differences (if at all). Conducting the analysis in more general terms for families of scoring rules (e.g., based on their characterization as Bregman loss functions) may also be the subject of future research.

Finally, the choice of strategy space for the agents is usually key in the analysis of equilibrium in game-like scenarios. Future research allowing for strategies beyond those assumed in our investigated setting (e.g., an update rate γ that also depends on the outstanding prediction) may result in different equilibrium (and socially optimal) strategies, and, as a result, in different properties for the related predictors.

Part IV

Summary

This thesis investigated the usage of machine learning in different settings and related robustness aspects. In the first setting, the Trading Agent Competition, a model-light approach to the design of a competing agent in the TAC-AA game proved to be top performing. Also, evidence of robustness of top performing agents to changes in the synthetic game environment suggests the applicability of their strategies to real scenarios. Furthermore, a new TAC game, AdX, was implemented as a platform for evaluating the mechanisms used in the Ad Exchange setting, and evidence from the first competitions conducted during 2014 provide insights regarding successful advertiser's strategies. In the second setting, Domain Adaptation, the methods of Robust Optimization and the concept of Algorithmic Robustness are combined to derive a generalization bound and a related Domain Adaptation SVM learning algorithm. The last setting researched in this thesis, sequential multi-agent learning, is introduced and shown to be robust to the unavailability of transaction history. That is, interpreting the state of the social computation as a predictor of an unknown underlying random variable, it's performance in equilibrium is quantified and shown to perform comparably to the optimal predictor having access to the full history. In what follows, the main results of the thesis are detailed together with possible related future research direction.

First, a simple model-light competing agent for TAC-AA was implemented, mainly relying on the model-free RWM on-line-learning algorithm. Making it into the final rounds of the TAC-AA 2010 competition proved the approach viable. In subsequent TAC-AA competition, the agent performance was significantly improved by modeling the TAC-AA user population distribution across states using the Particle Filtering method. Contrary to other reported implementation exploiting the game specification, ours was based on the model-free KNN algorithm for estimating the particle filter inputs. Due to the high inherent unpredictability in the TAC-AA scenario and related diminishing benefits of modeling improvements through Machine Learning, our simplifying approach proved to be as successful, eventually winning TAC-AA 2013. To cope with the relatively high estimation errors introduced by using KNN for particle filter inputs, an unorthodox exploration-exploitation particle-advance method was used, in which a small randomly chosen subset of the particles are not advanced and kept for the subsequent time step regardless of the observation. Theoretical analysis of this method is challenging and would be a very interesting future research.

Aiming at an assessment of the applicability of strategies employed by successful TAC agents to real scenarios, an empirical study of the robustness of TAC-AA agents was conducted. A series of experiments that varied different underlying game parameters (the competing agents being unaware) provided surprising evidence for the superior robustness of the top performing TAC-AA agents. This is surprising since one could expect that the high performance is due to fitting the strategies to the game spec, making such agents vulnerable to modeling mismatch. Therefore, the robustness results suggests that the strategies may be universally successful in a sense and applicable to more complex and unpredictable scenarios, such as those taking place in real settings. Another result obtained in this thesis regarding the TAC scenario is the ability to identify agents and predict their profitability by simple machine learning using agents' behavioral attributes (e.g., agent bidding activity, distributions of some events for different query types, and resulting ad position). Principal Component Analysis (with only two principal components sufficing) was used for agent identification, and 3-Nearest Neighbor for profit estimation. Associating the agents robustness and this characterization is left for future research, as also ways to use it as part of a TAC-AA competing agent's strategy.

Finally, for the TAC setting, a new TAC game for the Ad Exchange setting - TAC-AdX - is introduced and implemented, with competitions taking place during 2014. The game simulates key elements of the Ad Exchange setting (e.g., users of different types visits to publishers' web sites, impression opportunities announced and auctioned at the Ad Exchange, campaigns being allocated to advertisers) and competing agents implement that advertisers strategies aiming to acquire and execute targeted marketing campaigns. As with other TAC games, TAC-AdX provides a platform for evaluating in conjunction different competing agents strategies as well as alternative implementations of the underlying mechanisms of the scenario (e.g., the auction performed at the Ad Exchange, the pricing of information regarding users' attributes by dedicated third-parties, and the reserve price setting by the publisher). Reports by teams that implemented agents for the first TAC-AdX competitions suggest the effectiveness of some real-world methods for the TAC-AdX game, although a methodological evaluation of applicability of successful strategies by TAC-AdX agents to real setting (e.g., through robustness, as done for TAC-AA) is left for future research. Another natural follow-up research may include the empirical evaluation (given implementations of com-

peting agents) of alternative schemes for implementing a related TAC-AdX mechanism (e.g., reserve price optimization by the publishers).

Next, the Robust Optimization approach is used to derive a generalization bound and learning algorithm for the Domain Adaptation setting. A measure of discrepancy (λ-shift) between distributions is introduced, allowing its incorporation into a related generalization bound (Theorem 11). Furthermore, interpreting the prior knowledge regarding domain discrepancy and Algorithmic Robustness requirements as constraints on a related optimization program for learning a classifier, fits into the Robust Optimization framework and results in Domain Adaptation variants of SVM for classification and regression that are inherently Algorithmically Robust, with the dual formulation (8.17) suggesting a re-weighting interpretation. Natural extensions of this work as part of future research include the application of similar methods to other Algorithmically Robust learning algorithms (e.g., PCA), and the analysis of optimal space-partition methods (over which the λ-shift metric is defined) which were left out of the current research.

Last, this thesis considers the social learning setting, where a chain of agents (that is, a common computation propagating through a chain of communicating agents) seek an estimate of the probability, b, of some future binary event, based on private independent signals (each a realization of a binary event having the same underlying probability b). This research introduces strategic behavior to current models of influence propagation in social networks and provides a prescription for combining the private signal and the preceding agent's advice, quantifying the resulting performance (in equilibrium) of the Exponential Moving Average (EMA) strategy vs. the optimal achievable using Bayes updates (that is, having access to the full history). The quality of p_T, the prediction of the last agent along a chain of T agents (which may be viewed as an aggregate estimator of b that depends on the private signals of T agents) was studied and shown to be as follows

- (Theorem 18) When agents know their position in the sequence, the expected mean square error of the aggregate estimator is $\Theta(\frac{1}{T})$. Moreover, with probability $1-\delta$, the aggregate estimator's deviation from b is $\Theta\left(\sqrt{\frac{\ln(1/\delta)}{T}}\right)$.

- (Corollary 26) If the position information is not available, and agents are in equilibrium, the aggregate estimator has a mean square error of $O(\frac{1}{\sqrt{T}})$. Furthermore,

with probability $1-\delta$, the aggregate estimator's deviation from b is $\widetilde{O}\left(\sqrt{\frac{\ln(1/\delta)}{\sqrt{T}}}\right)$.

Extensions of the model are also analyzed for the cases where the total number of agents is only available through a common prior (the equilibrium update for this case is given in Theorem 27), for the case where there is only a single unaware agent (the optimal update rule for the single agent is provided in Theorem 29), and for the case where there is only a single aware agent having access to the full history (Theorem 30 quantifying the performance of the related resulting estimators). Future research directions for this setting include dealing with other Bregman-based losses beyond the quadratic loss analyzed in this work, and considering the resulting equilibrium for richer strategy spaces for the collaborating agents (in the most general case, where the agents' update may arbitrarily depend on the outstanding prediction and their signal), beyond the linear updates assumed herein.

Bibliography

[1] 13th international conference on autonomous agents and multiagent systems. http://aamas2014.lip6.fr/. 5.7

[2] 16th international workshop on agent-mediated electronic commerce and trading agents design and analysis. http://users.ecs.soton.ac.uk/vr2/amectada2014/. 5.7

[3] Alexa. http://www.alexa.com/. 5.3.2.1, 5.6.3

[4] The annual loebner competition. http://www.loebner.net/Prizef/loebner-prize.html. 2.2

[5] Marketwatch virtual stock exchange games. http://www.marketwatch.com/game/. 2.2

[6] Quantcast. www.quantcast.com. 5.3.2.1

[7] Tac ad auctions game. http://aa.tradingagents.org/. 2, 5.6

[8] Tac adx git repository. https://code.google.com/p/tac-adx/. 5.6

[9] Tac agents repository. http://tac.sics.se/showagents.php. 2.3

[10] Tau adx game workshop, fall 2013. http://sites.google.com/site/gameadx/agents. 5.7

[11] The trading agent competition - competitive benchmarking for the trading agent community. http://tac.sics.se/. 5.6

[12] J. Abernethy and R. Frongillo. A characterization of scoring rules for linear properties. *JMLR*, 2012. 9.5

[13] Jacob Abernethy and Rafael M. Frongillo. A collaborative mechanism for crowdsourcing prediction problems. *CoRR*, abs/1111.2664, 2011. 9.5

[14] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Trans. Sig. Proc.*, 50(2):174–188, February 2002. 3.4.3.1

[15] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, May 2002. 1.2

[16] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002. 1.2

[17] Robert J. Aumann. Agreeing to disagree. *The Annals of Statistics*, 4(6):pp. 1236–1239, 1976. 10.5.2

[18] Robert Axelrod. The evolution of strategies in the iterated prisoners dilemma. *The dynamics of norms*, pages 1–16, 1987. 2.2

[19] S. Balseiro, J. Feldman, V. Mirrokni, and S. Muthukrishnan. Yield Optimization of Display Advertising with Ad Exchange. *ArXiv e-prints*, February 2011. 5.1, 5.5.3.1

[20] Arindam Banerjee, Xin Guo, and Hui Wang. On the optimality of conditional expectation as a bregman predictor. *IEEE Transactions on Information Theory*, 51(7):2664–2669, 2005. 10.1.3, 10.5.3

[21] Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing closeness of discrete distributions. *J. ACM*, 60(1):4:1–4:25, February 2013. 6.3

[22] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010. 6.3, 4

[23] Shai Ben-David, Tyler Lu, Teresa Luu, and Dávid Pál. Impossibility theorems for domain adaptation. *Journal of Machine Learning Research - Proceedings Track*, 9:129–136, 2010. 6.3, 6.3

[24] Shai Ben-David and Ruth Urner. On the hardness of domain adaptation and the utility of unlabeled target samples. In *ALT*, pages 139–153, 2012. 6.3

[25] Shai Ben-David and Ruth Urner. Domain adaptation—can quantity compensate for quality? *Annals of Mathematics and Artificial Intelligence*, 70(3):185–202, March 2014. 6.3

[26] Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88(3):411–424, 2000. 7.1

[27] J. Berg, A. Greenwald, V. Naroditskiy, and E. Sodomka. A knapsack-based approach to bidding in ad auctions. In *ECAI*, pages 1013–1014, 2010. 4.1

[28] Jordan Berg, Amy Greenwald, Victor Naroditskiy, and Eric Sodomka. A first approach to autonomous bidding in ad auctions. In *In EC 2010 Workshop on Trading Agent Design and Analysis (TADA*, pages 343–348. SpringerVerlag, 2010. 3.1, 3.4, 3.4

[29] Dimitris Bertsimas, David B. Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM Rev.*, 53(3):464–501, August 2011. 7.1

[30] Steffen Bickel, Michael Brckner, and Tobias Scheffer. Discriminative learning for differing training and test distributions. In *In ICML*, pages 81–88. ACM Press, 2007. 6.3

[31] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning under covariate shift. *J. Mach. Learn. Res.*, 10:2137–2155, December 2009. 6.2.1

[32] Darse Billings. The first international RoShamBo programming competition. *ICGA Journal*, 23(1):42–50, 2000. 2.2

[33] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *Association for Computational Linguistics*, Prague, Czech Republic, 2007. 6.3

[34] A. Blum and Y. Mansour. Learning, regret-minimization, and equilibria. In *Algorithmic Game Theory, Chapter 4*. 2007. 3.3.2

[35] G. W. Brier. Verification of forecasts expressed in terms of probability. *Weather Rev*, 78:1–3, 1950. 9.4

[36] Lorenzo Bruzzone and Mattia Marconcini. Domain adaptation problems: A dasvm classification technique and a circular validation strategy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(5):770–787, May 2010. 6.3

[37] Sbastien Bubeck and Nicol Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012. 1.2

[38] U.S. Census Bureau. *Current Population Survey, 2013 Annual Social and Economic Supplemen*. Available at http://www.census.gov/prod/techdoc/cps/cpsmar13.pdf. 5.1, 5.3.1, 5.6.3

[39] Nicolò Cesa-Bianchi, Claudio Gentile, and Yishay Mansour. Regret minimization for reserve prices in second-price auctions. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pages 1190–1204. SIAM, 2013. 5.3.2.2

[40] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006. 1.2, 3.3.2

[41] Yee Seng Chan and Hwee Tou Ng. Word sense disambiguation with distribution estimation. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 1010–1015, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc. 10

[42] Yiling Chen and Jennifer Wortman Vaughan. A new understanding of prediction markets via no-regret learning. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, EC '10, pages 189–198, New York, NY, USA, 2010. ACM. 9.5, 9.5

[43] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *J. Artif. Int. Res.*, 4(1):129–145, March 1996. 6.2.3

[44] Corinna Cortes, Yishay Mansour, and Mehryar Mohri. Learning bounds for importance weighting. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors, *NIPS*, pages 442–450. Curran Associates, Inc., 2010. 6.2.1

[45] Corinna Cortes and Mehryar Mohri. Domain adaptation in regression. In *ALT*, pages 308–323, 2011. 6.3

[46] Corinna Cortes, Mehryar Mohri, Michael Riley, and Afshin Rostamizadeh. Sample selection bias correction theory. In *Proceedings of the 19th International Conference on Algorithmic Learning Theory*, ALT '08, pages 38–53, Berlin, Heidelberg, 2008. Springer-Verlag. 6.2.1, 6.2.3

[47] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 193–200, New York, NY, USA, 2007. ACM. 6.2.3

[48] Easley David and Kleinberg Jon. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*, chapter 16. Cambridge University Press, New York, NY, USA, 2010. 9.3

[49] Bruno De Finetti. La prevision: Ses lois logiques, ses sources subjectives. *Ann. Inst. Henri Poincaré*, 7:1–68, 1937. 9.4

[50] Nikhil R. Devanur and Thomas P. Hayes. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *ACM Conference on Electronic Commerce*. Association for Computing Machinery, Inc., 2009. 5.5.3.1

[51] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, July 2000. 3.4.3.1

[52] Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later, 2011. 3.4

[53] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007. 4.1

[54] Joan Feigenbaum, Lance Fortnow, David M. Pennock, and Rahul Sami. Computation in a distributed information market, 2004. 10.5.2, 10.5.3

[55] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Cliff Stein. Online stochastic packing applied to display ad allocation. In *Proceedings of the 18th Annual European Conference on Algorithms: Part I*, ESA'10, pages 182–194, Berlin, Heidelberg, 2010. Springer-Verlag. 5.5.3.1

[56] Jon Feldman, Vahab Mirrokni, S. Muthukrishnan, and Mallesh M. Pai. Auctions with intermediaries: Extended abstract. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, EC '10, pages 23–32, New York, NY, USA, 2010. ACM. 5.1, 5.3.3

[57] Social Fresh. *How To Use The 15 Facebook Ad Targeting Options*. http://socialfresh.com/facebook-ad-options/. 5.3.1

[58] John Geanakoplos and Heracles M. Polemarchakis. We can't disagree forever. Cowles Foundation Discussion Papers 639, Cowles Foundation for Research in Economics, Yale University, July 1982. 10.5.2

[59] Arpita Ghosh, Mohammad Mahdian, Preston McAfee, and Sergei Vassilvitskii. To match or not to match: Economics of cookie matching in online advertising.

In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, pages 741–753, New York, NY, USA, 2012. ACM. 5.3.3

[60] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. 9.4

[61] I. J. Good. Rational decisions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 14(1):pp. 107–114, 1952. 9.4

[62] Amy Greenwald and Peter Stone. The first international trading agent competition: Autonomous bidding agents. *IEEE Internet Computing*, 5(2):52–60, 2001. 2, 6.2.3

[63] Amaury Habrard, Jean-Philippe Peyrache, and Marc Sebban. Boosting for unsupervised domain adaptation. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Zelezn, editors, *ECML/PKDD (2)*, volume 8189 of *Lecture Notes in Computer Science*, pages 433–448. Springer, 2013. 6.3

[64] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009. 3.4.3.3

[65] Robin Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5(1):107–119, 2003. 9.5

[66] Shai Hertz, Mariano Schain, and Yishay Mansour. An empirical study of trading agent robustness. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 1253–1254, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems. 1.5.4, 4

[67] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schlkopf. Correcting sample selection bias by unlabeled data. In Bernhard Schlkopf, John Platt, and Thomas Hoffman, editors, *NIPS*, pages 601–608. MIT Press, 2006. 6.3

[68] iab. RTB Project. *OpenRTB API Specification Version 2.2*. Available at `http://www.iab.net/media/file/OpenRTBAPISpecificationVersion2_2.pdf`. 5.3.1

[69] Patrick R. Jordan, Michael P. Wellman, and Guha Balakrishnan. Strategy and mechanism lessons from the first ad auctions trading agent competition. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, EC '10, pages 287–296, New York, NY, USA, 2010. ACM. 2.3, 4.1, 4.3.3

[70] PatrickR. Jordan and MichaelP. Wellman. Designing an ad auctions game for the trading agent competition. In Esther David, Enrico Gerding, David Sarne, and Onn Shehory, editors, *Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*, volume 59 of *Lecture Notes in Business Information Processing*, pages 147–162. Springer Berlin Heidelberg, 2010. 2, 3.1, 4.1, 5.2.2

[71] Takafumi Kanamori and Hidetoshi Shimodaira. Active learning algorithm using the maximum weighted log-likelihood estimator. *Journal of Statistical Planning and Inference*, 116(1):149 – 162, 2003. 6.2.3

[72] W. Ketter, J. Collins, P. Reddy, C. Flath, and M. Weerdt. The power trading agent competition. *ERIM Report Series Reference No. ERS-2011-027-LIS*, 2011. 2

[73] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. *Proceedings of the 30th International Conference on Very Large Data Bases*, 2004. 6.3

[74] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Proceedings of the First International Conference on Autonomous Agents*, AGENTS '97, pages 340–347, New York, NY, USA, 1997. ACM. 2.2

[75] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1785–1792, Washington, DC, USA, 2011. IEEE Computer Society. 6.3

[76] Albert S. Kyle. Continuous auctions and insider trading. *Econometrica*, 53(6):pp. 1315–1335, 1985. 10.5.2

[77] S. Lahaie, D. M. Pennock, A. Saberi, and R. V. Vohra. Sponsored search auctions. In N. Nisan, T. roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, chapter 28, pages 699–716. Cambridge University Press, New York, NY, 2007. 4.1

[78] Nicolas S. Lambert, David M. Pennock, and Yoav Shoham. Eliciting properties of probability distributions. In *Proceedings of the 9th ACM Conference on Electronic Commerce*, EC '08, pages 129–138, New York, NY, USA, 2008. ACM. 10.5.3

[79] John Ledyard, Robin Hanson, and Takashi Ishikida. An experimental test of combinatorial information markets. *Journal of Economic Behavior and Organization*, 69(2):182 – 189, 2009. Individual Decision-Making, Bayesian Estimation and Market Design: A Festschrift in honor of David Grether. 9.5

[80] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Inf. Comput.*, 108(2):212–261, February 1994. 1.2, 3.3.2

[81] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *COLT*, 2009. 6.3, 6.3, 5, 6.3

[82] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. *Advances in Neural Information Processing Systems (2008)*, 2009. 6.2.3

[83] Yishay Mansour, S. Muthukrishnan, and Noam Nisan. Doubleclick ad exchange auction. *CoRR*, abs/1204.0535, 2012. 5.1, 5.3.3

[84] Yishay Mansour and Mariano Schain. Robust domain adaptation. *Annals of Mathematics and Artificial Intelligence*, pages 1–16, 2013. 1.5.4, 6.3, 7.1

[85] Anna Margolis. A literature review of domain adaptation with unlabeled data. Technical report, 2011. 6.3

[86] Colin McDiarmid. On the method of bounded differences, surveys in combinatorics. *Math. Soc. Lecture*, pages 148–188, 1989. 15

[87] Richard D. McKelvey and Talbot Page. Common knowledge, consensus, and aggregate information. *Econometrica*, 54(1):pp. 109–127, 1986. 10.5.2

[88] John Blitzer Minmin Chen and Kilian Weinberger. Co-training for domain adaptation. In *Neural Information Processing Systems*, Cambridge, MA, 2011. MIT Press. 6.3

[89] Mehryar Mohri and Andres Muñoz Medina. Learning theory and algorithms for revenue optimization in second-price auctions with reserve. *CoRR*, abs/1310.5665, 2013. 5.3.2.2, 5.7

[90] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. 1.1, 1, 1.1, 1.2, 1.2, 6.3, 6.3

[91] S. Muthukrishnan. Ad exchanges: Research issues. In *WINE*, pages 1–12, 2009. 5.1, 5.3.3

[92] R. Myerson. Optimal auction design. *Mathematics of Operations Research, 6(1)*, pages 58–73, 1981. 5.3.2.2

[93] Jinzhong Niu, Kai Cai, Simon Parsons, Peter McBurney, and Enrico Gerding. What the 2007 TAC market design game tells us about effective auction mechanisms. *Autonomous Agents and Multi-Agent Systems*, 21:172–203, 2010. 2

[94] Michael Ostrovsky. Information aggregation in dynamic markets with strategic traders. Research Papers 2053, Stanford University, Graduate School of Business, March 2009. 10.5.2

[95] Michael Ostrovsky and Michael Schwarz. Reserve prices in internet advertising auctions: A field experiment. In *Proceedings of the 12th ACM Conference on Electronic Commerce*, EC '11, pages 59–60, New York, NY, USA, 2011. ACM. 5.3.2.2

[96] Abraham Othman and Tuomas Sandholm. The gates hillman prediction market. *Review of Economic Design*, 17(2):95–128, 2013. 9.5

[97] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, 22(10):1345–1359, October 2010. 6.2.3

[98] D. Pardoe, D. Chakraborty, and P. Stone. Tactex09: a champion bidding agent for ad auctions. In W. van der Hoek, G. A. Kaminka, Y. Lespérance, M. Luck, and S. Sen, editors, *AAMAS*, pages 1273–1280. IFAAMAS, 2010. 4.1

[99] David Pardoe, Doran Chakraborty, and Peter Stone. Tactex09: A champion bidding agent for ad auctions. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, AAMAS '10, pages 1273–1280, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems. 3.1, 3.4, 3.4.3.2, 3.4.3.3

[100] David Pardoe and Peter Stone. Boosting for regression transfer. In Johannes Frnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 863–870. Omnipress, 2010. 6.2.3

[101] David Pardoe and Peter Stone. A particle filter for bid estimation in ad auctions with periodic ranking observations. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '11, pages 887–894, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems. 2.2, 3.3.3, 4.1

[102] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009. 6.1

[103] Achim Rettinger, Martin Zinkevich, and Michael Bowling. Boosting expert ensembles for rapid concept recall. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 464–469. AAAI Press, 2006. 6.2.3

[104] John Rust, John H. Miller, and Richard Palmer. Characterizing effective trading strategies: Insights from a computerized double auction tournament. *Journal of Economic Dynamics and Control*, 18(1):61 – 96, 1994. Special Issue on Computer Science and Economics. 2.2

[105] Norman Sadeh, Raghu Arunachalam, Joakim Eriksson, Niclas Finne, and Sverker Janson. TAC-03: A supply-chain trading competition. *AI Magazine*, 24(1):92–94, 2003. 2, 6.2.3

[106] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 213–226, Berlin, Heidelberg, 2010. Springer-Verlag. 6.3

[107] F. Sanders. On Subjective Probability Forecasting. *Journal of Applied Meteorology*, 2:191–201, April 1963. 9.4

[108] Leonard J. Savage. Elicitation of personal probabilities and expectations. *Journal of the American Statistical Association*, 66(336):pp. 783–801, 1971. 9.4

[109] Mariano Schain, Shai Hertz, and Yishay Mansour. A model-free approach for a tac-aa trading agent. In Esther David, Christopher Kiekintveld, Valentin Robu, Onn Shehory, and Sebastian Stein, editors, *AMEC/TADA*, volume 136 of *Lecture Notes in Business Information Processing*, pages 119–132. Springer, 2012. 1.5.4, 3

[110] Mariano Schain and Yishay Mansour. Ad exchange - proposal for a new trading agent competition game. In *AMEC/TADA*, pages 133–145, 2012. 1.5.4, 2

[111] Shai Ben-David Shai Shalev-Shwartz. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. 2, 1.4

[112] Lloyd S Shapley and Martin Shubik. Trade using one commodity as a means of payment. *Journal of Political Economy*, 85(5):937–68, October 1977. 10.5.2

[113] Lampros C. Stavrogiannis. Competing demand-side intermediary auctioneers in online advertising exchanges. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pages 1705–1706, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems. 5.3.3

[114] Masashi Sugiyama and Motoaki Kawanabe. *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. The MIT Press, 2012. 6.1

[115] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul Von Bnau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *In NIPS*, 2008. 6.2.1, 6.3

[116] G. Sutcliffe. The cade-17 atp system competition. *J. Autom. Reason.*, 27(3):227–250, October 2001. 2.2

[117] Bingyang Tao, Fan Wu, and Guihai Chen. TAC adx'14: Autonomous agents for realtime ad exchange. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pages 1111–1119, 2015. 5.7

[118] H. R. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, 2007. 4.1

[119] Y Vorobeychik. A game theoretic bidding agent for the ad auction game. In *Third International Conference on Agents and Articial Intelligence, ICAART*, 2011. 3.1

[120] Pengyuan Wang, Yechao Liu, Marsha Meytlis, Han-Yun Tsao, Jian Yang, and Pei Huang. An efficient framework for online advertising effectiveness measurement and comparison. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 163–172, New York, NY, USA, 2014. ACM. 5.3.4.1

[121] Michael P. Wellman, Amy Greenwald, and Peter Stone. *Autonomous Bidding Agents: Strategies and Lessons from the Trading Agent Competition (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2007. 2, 2.3

[122] Michael P. Wellman and Peter R. Wurman. A trading agent competition for the research community. In *IJCAI-99 Workshop on Agent-Mediated Electronic Trading*, Stockholm, August 1999. 2

[123] Robert L. Winkler. Scoring rules and the evaluation of probability assessors. *Journal of the American Statistical Association*, 64(327):pp. 1073–1078, 1969. 9.4

[124] Dikan Xing, Wenyuan Dai, Gui-Rong Xue, and Yong Yu. Bridged refinement for transfer learning. In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, PKDD 2007, pages 324–335, Berlin, Heidelberg, 2007. Springer-Verlag. 6.3

[125] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *J. Mach. Learn. Res.*, 10:1485–1510, December 2009. 7.1

[126] Huan Xu and Shie Mannor. Robustness and generalization. In *COLT*, pages 503–515, 2010. 7.2, 6, 7.2, 7, 6, 8.2, 8.3, 8.4, 4, 8.6

[127] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *IN ICML*, pages 912–919, 2003. 6.3

המחקר מניח מרחב אסטרטגיות פשוט לכל סוכן (עדכון החישוב הנוכחי ע"י שקלול לינארי עם הסיגנל הפרטי) ומעוניינים באסטרטגיה האופטימלית מבחינה חברתית (כלומר, המשיגה בתוחלת את התועלת הכללית הגבוהה ביותר עבור כלל ניקוד נתון) ובאסטרטגיות שווי המשקל (כלומר, מצב בו תיפגע תוחלת הניקוד של סוכן שיסטה ממנה, בעוד שאר הסוכנים ממשיכים לנקוט בה). עבור אסטרטגיות אלו (המנותחות בפרק 10 עבור כללי ניקוד ריבועיים – Quadratic Scoring Rules), מתקבלים תהליכים הניתנים לפרשנות כמשערכים להסתברות המאורע החיצוני. אנו מנתחים את ביצועי משערכים אלו ומשווים אותם לביצועי מהשערך האופטימלי (בייסיאני) האפשרי במידה והיסטורית הפעולות היתה ידועה ונגישה. בהמשך, אנו מרחיבים את המודל למקרים בהם הסוכנים אינם יודעים מראש את מספר הסוכנים הכולל (אלא רק התפלגות – הידועה לכל הסוכנים – לגבי מספר הסוכנים הכולל), למקרים בהם רק אחד הסוכנים נטול היסטוריה (ואת השוני בתועלת כמייצג את הערך של ידיעת ההיסטוריה), ולמקרים בהם ההסטוריה נגישה רק לאחד הסוכנים (ואת השוני בתועלת כמייצג חסם על המחיר שסוכן יסכים לשלם כדי לקבל גישה כאמור). פרק 10 נחתם באזכור של מחקרי עבר של כמה מודלים קשורים (אם כי לא נחקרו בעבר שיוויי משקל ותכונות משערכים במודל המוצג בו אוסף גדול של סוכנים הפועלים סדרתית בצורה אסטרטגית), ובהתייחסות למחקר עתידי אפשרי עבור תרחישים המשתמשים בכללי ניקוד אחרים או המאפשרים לסוכנים מרחבי אסטרטגיות רחבים יותר.

מתוארים בפרק 7. אלגוריתמי ההסתגלות וחסמי ההכללה המוצגים בעבודה זו מבוססים על תוצאות המאמר הבא:

Robust domain adaptation. Mansour, Yishay and Schain, Mariano. Annals of Mathematics and Artificial Intelligence, pages 1-16, 2013. Springer.

חלק שלישי – למידה חברתית

בלמידה חברתית (Social Learning) המתוארת בפרק 9 של העבודה, אוסף סוכנים בעלי מידע פרטי משתפים פעולה להשגת תוצאה משותפת שאמורה לשקף את סך המידע המבוזר בין הסוכנים. המידה בה תוצאת החישוב אכן מייצגת את סך המידע תלויה בתועלת של כל אחד מהסוכנים ובנכונותו לפעול באופן המייצג את המידע הפרטי שברשותו. מבחינים בין שני תרחישים של למידה חברתית השונים מהותית. בראשון, התועלת של כל אחד מהסוכנים תלויה בתוצאת החישוב המשותף (למשל, בעת בחירות). בשני, בו אנו מתמקדים בעבודה זו, התועלת של כל סוכן קשורה במאורע חיצוני, והמידע הפרטי של הסוכן מתייחס להסתברות התרחשות המאורע (לדוגמה, המאורע הוא פשיטת רגל של חברה מסויימת, ולכל סוכן מידע פנים המייצג הסתברות להתרחשות המאורע). תוצאת החישוב המשותף אמורה לייצג את "חכמת ההמון" בקשר להסתברות האמיתית להתרחשות המאורע, והתועלת של כל סוכן נקבעת בדיעבד לאור קרות המאורע או לא (למשל, מחיר אגרת חוב של חברה כמשקף תחזית ההמון לגבי סיכויי פשיטת הרגל של החברה, והתועלת לסוכן בהתאם למחיר בו קנה את אגרת החוב ומצב החברה בעת תשלום החוב). בפרק 9 מתוארים שווקי תחזיות (Prediction Markets) המבוססים על כללי ניקוד (Scoring Rules) כדוגמה לתרחיש למידה חברתית, מוצגת התייחסות למחיר של האופציה הנסחרת בשוק כמשערך של ההסתברות לקרות המאורע העומד בבסיסו, ומפותחת נוסחה הקושרת בין העלות של עושה השוק (מעצם טבעם, בשווקי תחזיות נדרש סבסוד חיצוני) לאיכות המשערך המתקבל.

בפרק 10 אנו חוקרים מודל של למידה חברתית סדרתית נטולת היסטוריה. עבור מאורע חיצוני שסיכויי התרחשותו אינם ידועים לסוכנים, כל סוכן מקבל דגימה אחת (סיגנל) של ניסוי בעל אותם סיכויי התרחשות. הסוכנים חולקים ביניהם את המידע (באופן עקיף) על ידי עדכון סדרתי של חישוב משותף התלוי בסיגנל של כל סוכן כך שבעת העדכון היסטוריית העדכונים אינה נגישה לסוכן (אלא רק תוצאת החישוב עד כה). ניתן להתייחס למודל זה כמשקף שרשרת המלצות בין אנשים לגבי טיב מוצר מסויים על בסיס נסיון שימוש פרטי ואמון בממליץ, בהנחה שלממליצים חשובה תדמיתם בעיני חבריהם ולכן פועלים ביושר על פי הסיגנל האמיתי שבידיהם.

בעלות גבוהה מהתשלום מהמפרסם), והאפשרות לנצל מצב טיפוסי לקראת סוף המשחק (בו נותרים מעט סוכנים עם אפשרות לזכות במסעות פרסום) לצורך הפקת רווח משמעותי. תוצאות אלו מחזקות את הרלוונטיות של המשחק כפלטפורמה לחקר תרחישים דומים במציאות (וגם לצרכים אקדמיים – המשחק ממשיך לשמש כפלטפורמה לסדנאות רלוונטיות באוניברסיטאות שונות). בפרק 5 נמצא תיאור מפורט של המשחק והיישויות השונות, כמו גם הארכיטקטורה והמימוש. תיאור זה מבוסס בחלקו על הצעת המשחק שפורסמה במאמר הבא:

Ad Exchange - Proposal for a New Trading Agent Competition Game. Schain, Mariano and Mansour, Yishay. 14th International Workshop on Agent-Mediated Electronic Commerce and Trading Agents Design and Analysis 2012.

חלק שני – הסתגלות חסינה

בעיית ההסתגלות (Domain Adaptation) היא מושא המחקר בחלק זה של התזה. הבעיה מוצגת בפרק 6, ובה על אלגוריתם למידה חישובית להתמודד עם מצב בו, בניגוד להנחה המקובלת בתיאוריה הקלאסית של למידה חישובית, סביבת האימון שונה מסביבת המבחן. כלומר – התכונות הסטטיסטיות של סביבת המקור (כלומר, של דגימות האימון אשר באמצעותם משיג האלגוריתם יכולת הכללה) שונות מאלו של הסביבה בה על תוצאת אלגוריתם הלמידה לפעול (סביבת היעד). באופן טבעי יש להניח קשר כלשהו בין סביבת המקור והיעד בכדי שתתאפשר יכולת למידה והכללה במצב שכזה. בפרק 6 מוצגים המתודולוגיות שהיו מוכרות להתמודדות עם הבעיה, תוך דגש על חסמי הסתגלות המבוססים על מידת פער בין סביבות המקור והיעד.

בעבודה זו מוצגים אלגוריתמי למידה וחסמי הכללה להסתגלות המבוססים על שיטות אופטימיזציה חסינה (Robust Optimization). באופטימיזציה חסינה (בניגוד לאופטימיזציה *רגילה*) אנו מחפשים פתרון אופטימלי תחת אילוצים המשקפים אי ודאות לגבי המצב. כתוצאה מכך, הפתרון של אופטימיזציה כזו *חסין* לטעויות בהערכת המצב (עד כדי המרווח הנתון ע"י האילוצים). בפרק 8 אנו מציגים פתרון לבעית ההסתגלות באמצעות אופטימיזציה חסינה ע"י אפיון הפער בין סביבת המקור והיעד כאילוצים המייצגים את חוסר הודאות בין הסביבות. בפיתוח האלגוריתם אנו מסתמכים גם על תיאוריית החסינות האלגוריתמית (Algorithmic Robustness), המאפיינת ומכמתת את יכולות ההכללה של אלגוריתמים אשר להם השתנות חסומה של תוצר הלמידה בסביבות איברי מדגם האימון. השימוש בחסינות אלגוריתמית מאפשר לאלגוריתם שלנו לגשר על השוני בין סביבות היעד והמקור ומספק חסמי הכללה התלויים במידת שוני בין סביבות המוצגת גם היא בפרק 8. אופטימיזציה חסינה וחסינות אלגוריתמית

מספר הימים בו הם משתתפים במכרזים, השונות ברווחים היומיים, ותמהיל המוצרים הממוצע) ושימוש באלגוריתם למידה חישובית מסוג PCA (אנליזה של רכיבים ראשיים - Principal Component Analysis) הודגמה האפשרות לסווג סוכנים בדיוק גבוה ע"פ שני הרכיבים הראשיים בלבד. פרק זה מבוסס בעיקר על תוצאות שפורסמו במאמר הבא:

An Empirical Study of Trading Agent Robustness. Hertz, Shai, and Schain, Mariano and Mansour, Yishay. Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems.

בפרק 5 מתואר משחק TAC חדש, TAC-Ad Exchange שאופיין ומומש במסגרת התזה. המשחק מדמה סביבת התמודדות על פרסום למשתמשים הגולשים באתרי אינטרנט. בניגוד לפרסום בחיפוש ממומן, כאן מכרזי הפרסום מבוצעים ע"י רכיב Ad Exchange אליו מתנקזים בזמן אמת הזדמנויות הפרסום (מאתרי אינטרנט בהם מבקרים המשתמשים) וההצעות של המפרסמים (באמצעות סוכנויות פרסום, Ad Networks). המשחק פותח מנקודת המבט של ה Ad Networks במטרה לאפשר בחינה של אסטרטגיות הוצאה לפועל של מסעות פרסום במציאות מורכבת בה משולבים מנגנונים רבים המבוצעים ע"י הישוויות המעורבות: בעיקר קביעת מחירי מינימום לפרסום ע"י האתרים, מכרזי הפרסום ע"י ה Ad Exchange, סחר במידע לגבי המשתמשים (משמש את המפרסמים כדי להתאים את הפרסומות לאוכלוסיית היעד הנדרשת), והקצאת חוזי הפרסום ל Ad Networks ע"י המפרסמים.

במשחק, הסוכנים המתחרים מממשים את האסטרטגיות של ה Ad Networks. בתחילת המשחק לכל סוכן מוקצה מסע פרסום שעליו להוציא לפועל (מסע פרסום מאופיין ע"י אוכלוסיית היעד, כמות הצפיות המבוקשת, והתשלום לסוכנות). בהמשך המשחק מתמודדים הסוכנים על זכיה במסעות פרסום נוספים ע"י הצעת העלות למפרסם. ההצעה ובה העלות הנמוכה ביותר זוכה, אך תוך התחשבות בביצועי העבר (ביכולתו המוכחת להוציא מסעות פרסום לפועל) של כל סוכן מתחרה. על הסוכן המתחרה לאזן לפיכך בין התועלת לטווח הקצר (הוצאה לפועל של מסע הפרסום הנוכחי בעלות נמוכה ככל האפשר, תוך סיכון של אי מימוש כנדרש) לבין ביצוע החוזה "בכל מחיר" והשגת שיפור תדמיתי במטרה לשפר סיכוייו לזכיה במסעות פרסום עתידיים.

עם סיום מימוש המשחק התקיימו מספר תחרויות (אחת במסגרת TAC 2014, והשניה במסגרת סדנא במדעי המחשב באוניברסיטת תל אביב) בהן השתתפו קבוצות מאוניברסיטאות בארה"ב, בריטנה, סין, וישראל. בחינה של אסטרטגיות הסוכנים במשחק ומסקנות הקבוצות המשתתפות מרמזת על דינאמיקה ותופעות שעשויות להתרחש במציאות. בפרט, החשיבות שבביצוע מסעות הפרסום הראשונים (גם

שימוש בשיטות סטטיסטיות – Particle Filtering) והרחבנו את אפשרויות הפעולה שלו כך שיוכל להתמודד במכרזים גם על מקומות נוספים פרט לראשון. בכל מקרה האופי נטול המודל נשמר ע"י ביסוס הקלט למנגנון ה Particle Filter על אלגוריתם Nearest Neighbor הנטול-מודל במהותו, בניגוד לקבוצות אחרות שמימשו מנגנון זה אך ביססו אותו על המבנה (המודל) הספציפי של המשחק. בנוסף, שיפרנו את רכיב האופטימיזציה ע"י הנחה היוריסטית לגבי הקשר בין המחיר הזוכה במכרז לבין כמות המוצרים שיימכרו והרווח הצפוי לסוכן. הנחה זו איפשר לממש אופטימיזציה ע"י חיפוש פשוט אחר הצעת המחיר האופטימלית במכרז (בניגוד לשיטות מורכבות ויקרות חישובית ששימשו לאופטימיזציה, ע"פ פרסומים, את הסוכנים התמחרים). שיטת אופטימיזציה זו הוכחה כיעילה, כמו גם מנגנון המידול, (שאמנם לא השיג דיוק במודל כמו שיטות תלויות-מודל – חוסר דיוק שלא השפיע מהותית על התוצאה הסופית לאור רמת הרעש הגבוהה המובנית במשחק) ובתחרויות של 2011 ו 2012 הסוכן הגיע למקום השלישי ובתוצאה הקרובה עד כדי 3% לזו של הסוכן המנצח. בסופו של דבר, לאחר סבב שיפורים נוסף, הסוכן זכה בתחרות של שנת 2013. פרק 3 בעבודה מכיל תיאור מפורט של הארכיטקטורה והמנגנונים השונים שמומשו בסוכן, כמו גם ניתוח של התועלת השולית היורדת של שיפור הדיוק של אלגוריתמי הלמידה המשמשים במידול. פרק זה מבוסס בעיקר על תוצאות שפורסמו במאמר הבא:

A Model-Free Approach for a TAC-AA Trading Agent. Schain, Mariano and Hertz, Shai and Mansour, Yishay. Lecture Notes in Business Information Processing 2012, and Trading Agents Design and Analysis (TADA) Workshop 2012.

בפרק 4 מתוארים אוסף ניסויים שבוצעו לבחינת הרלוונטיות של מנגנונים המשמשים סוכנים במשחקי TAC (כלומר, במצבים מופשטים יחסית) למצבים מורכבים במציאות. לצורך כך, שינינו את מימוש שרת המשחק (בין השאר, שונה מודל מעבר המשתמשים בין המצבים השונים, מודל ההקלקות, גודל אוכלוסיות המשתמשים, ועוד) ומדדנו את השינויים בביצועים של סוכנים מתחרויות TAC-AA קודמות (הסוכנים לא שונו). התוצאות הראו שדווקא הסוכנים בעלי הביצועים הגבוהים בתחרויות TAC-AA הפגינו חוסן והיו עמידים יותר לשינויים. לאור תוצאה מפתיעה זו (הרי בכדי להשיג תוצאות שיא, צפוי היה שסוכנים יותאמו ככל האפשר לחוקים הספציפיים של המשחק), ניתן להעריך שמנגנונים המשמשים סוכנים בעלי ביצועים גבוהים עשויים להיות רלוונטיים גם למציאות המורכבת האמיתית, ובכך עולה הערך של משחקים דוגמת TAC כפלטפורמות סינטטיות לבחינת שיטות העשויות לשמש במציאות.

כמו כן נבדקה בניסויים האפשרות לזהות סוכנים ע"פ טביעת אצבע התנהגותית (לעתים במצבים כאלו, ובמשחקי TAC בפרט, זהות הסוכנים היריבים לא נתונה במהלך המשחק או בדוחות המתפרסמים תוך כדי ואחרי תום המשחק). ע"י הגדרת מספר מאפיינים קטן (בפרט, המיקום בו הסוכנים זוכים במכרזים,

מציבים לסוכנים המתחרים אתגר מורכב מתחום תורת המשחקים, ובכך הם נבדלים מסביבות משחקים מלאכותיות אחרות אשר מציבות אתגרים מתחום תורת ההחלטות (כלומר, בעיות אופטימיזציה במהותן, שאינן כוללות יריב בעל אסטרטגיה לא ידועה) או בעלות מרחבי אסטרטגיה פשוטים מדי עבור הסוכנים המתחרים (לדוגמה, החלטה בודדת, לעומת אוסף החלטות יומיות ותלויות-הדדית במשחקי TAC) באופן שלא מאפשר להכיל את מורכבות האתגרים הרלוונטיים במסחר אוטונומי באינטרנט. לפיכך, על סוכנים מתחרים במשחקי TAC לממש אסטרטגיות מסתגלות המאזנות בין השפעות של ההחלטות האפשריות בסביבת מסחר דינאמית על בסיס משוב מחזורי ובתנאי אי וודאות. ארכיטקטורת מימוש של סוכן גנרי תכלול לכן רכיב מידול המשערך את מצב המשחק מתוך הדו"חות, ורכיב אופטימיזציה המחשב החלטה מיטבית עבור המצב המשוערך.

במשחק TAC-AA, שרת המשחק מדמה אוכלוסיות רוכשים פוטנציאליים ("משתמשים") באינטרנט (כל אוכלוסיה עם העדפה מובנית למוצרים מסוג מסויים) אשר מבצעים מדי יום חיפוש על פי מצב המייצג את נטייתם לבצע רכישה בסופו של דבר. הסוכנים המתחרים מעוניינים למכור מלאי מוצרים מוגבל, ומתחרים במכרזים על הזכות להציג מודעות פרסום במקביל לתוצאת החיפוש של המשתמשים. משתמש עשוי לבחור (בהקלקה) אחת ממודעות הפרסום שזכו במכרז (פעולה זו גוררת עלות לסוכן, ע"פ תוצאות המכרז, בשיטת מחיר שני מוכלל – Generalized Second Price) ובסיכוי מסויים (התלוי במצבו ובגורמים נוספים) מבצע רכישה (פעולה הגוררת רווח לסוכן). מכירה של מוצרים מעבר למלאי נתון עלולה לגרור הפסדים לסוכן ועליו להעריך לפיכך (על בסיס מידע חלקי לגבי המצב האפשרי של המשתמשים השונים, רמות המלאי הנוכחיות, ופעולות הסוכנים המתחרים) את התועלת השולית של זכיה במכרז להצגת מודעה (וכנגזרת, הצעת המחיר במכרז). מאפייני משחקי TAC ומשחק TAC-AA בפרט מפורטים בפרק 2.

בפרק 3 אנו מתארים תכנון ומימוש סוכנים לתחרויות TAC-AA בגישה נטולת מודל (model-light), ומראים שסוכן כאמור יכול להשיג ביצועים עליונים וגם להנות מיתרונות החוסן והפשטות הנלווים. במימוש הסוכן הראשון, לתחרות TAC-AA 2010, רכיב מידול התעלם כמעט כליל ממבנה המשחק ורכיב האופטימיזציה מימש אלגוריתם מזעור חרטה (Regret Minimization) לבחירת תמהיל המוצרים למכירה (על פי הרווח בעבר כפי שדווח עבור כל מוצר). בהנתן התמהיל הרצוי, הסוכן מציע במכרז תוך נסיון לזכות במיקום הראשון עבור כל מוצר בתמהיל. תוצאות התחרות הראו שהגבלות אלו על הסוכן עדיין אפשרו לו להגיע למקום גבוה – מעל התוצאה החציונית בתחרות ובפכר של כ 30% מתחת לתוצאה המנצחת.

לקראת התחרות הבאה בשנת 2011, ובמטרה לזהות את המרכיבים הדרושים כדי לנצח בתחרות (תוך שמירה על גישה נטולת-מודל ככל האפשר) הוספנו לסוכן יכולת להעריך את מצב המשתמשים (ע"י

תקציר

חלק ראשון – סוכני מסחר אוטונומיים

בחלק זה נחקרות אסטרטגיות של סוכני מסחר אוטונומיים (כלומר, המקבלים החלטות לאורך זמן באופן עצמאי וללא התערבות אדם) המממשים אלגוריתמי למידה לשם הוצאה לפועל מיטבית של מסעות פרסום באינטרנט. סוכנים אלו, אבן הפינה של כלכלת האינטרנט, מתחרים במכרזים בזמן אמת על הזדמנויות פרסום לגולשים באתרי אינטרנט. הצעות המחיר של הסוכנים מושפעות מגורמים רבים, בעיקר מאפייני המשתמשים (אליהם מיועדות הפרסומות) והאתרים, מאפייני מסע הפרסום (סוגי וכמות קהל היעד, אורך זמן, וכדומה), אופי המפרסמים המתחרים ומסעות הפרסום שלהם, ועוד.

המבנה המורכב של הפרסום באינטרנט והקושי להעריך ביצועי אסטרטגיות באופן בלתי תלוי (מעצם טבע המצב, ביצועי אסטרטגיה תלויים באופן הדוק באסטרטגיה המממושת ע"י הסוכנים המתחרים) הובילו לפיתוח סביבות מלאכותיות המדמות את הסביבה האמיתית (אם כי תוך כדי הפשטה מתבקשת) ומהוות תשתית לבחינה מבוקרת של אסטרטגיות כאמור. אחת הסביבות האלו היא TAC Trading - Agent Competition, קהילת מחקר המנהלת מאז שנת 2000 תחרויות שנתיות בהן מתעמתים סוכנים סוחרים המממשים אסטרטגיות מטעם צוותי מחקר שונים. סביבה מבוקרת ושקופה שכזו מאפשרת ניתוח והערכה של ההאסטרטגיות השונות ואלגוריתמי הלמידה הממומשים בהן (ובכלל של מנגונינים המשמשים יישויות המשתתפות בתרחיש – שיטות המכרזים למשל) ומהווה ציר מחקר לקבוצות רבות החוקרות שיטות רלוונטיות. משחק TAC הראשון, בו התחרו בשנת 2000, הציע סביבת מסחר לרכישת רכיבי חופשות (כלומר, ספקי טיסות, מלונות ובידור, מול סוכני נסיעות המתחרים ביניהם על תיאום חופשות לרוכשים פוטנציאליים). משחקים נוספים הוצגו בשנים שלאחר מכן, לסביבת ניהול שרשראות אספקה, תכנון שווקים, ופרסום בחיפוש ממומן (Sponsored Search)(TAC-AA) TAC Ad Auctions, נשוא פרקים 3 ו 4 של עבודה זו, בהם מתואר סוכן מתחרה שמומש כחלק מעבודת התזה, ובחינה אמפירית של חוסן סוכנים לשינויים כאמצעי להערכת אפשרות להעברת הטכנולוגיות לשימוש בעולם האמיתי. לאחרונה הוצעו משחק המדמה מצבי מסחר מבוזר בין ספקי וצרכני אנרגיה, ומשחק לסביבת Ad Exchange – מנגנון ריכוזי למכירה פומבית למפרסמים, שפותח וממומש כחלק מעבודת התזה ומתואר בפרק 5.

למשחקי TAC מאפיינים משותפים רבים. בכולם שרת מרכזי (המסמלץ את הסביבה ומממש את המשחק) המתקשר עם סוכנים מתחרים מרוחקים ובלתי תלויים. משחקי TAC מבוססים על מחזור יומי ובו שרת המשחק מבצע סימולציה על סמך החלטות יומיות המתקבלות מהסוכנים המתחרים (תוצאות חלקיות של הסימולציה מדווחות לסוכנים, לצורך קבלת החלטות עתידיות). במהותם, משחקי TAC

rules) סטנדרטיים, אנו מנתחים אסטרטגיות אופטימליות חברתית, את אסטרטגיות שיווי המשקל, ואת ביצועי החישוב המתקבל עבור אסטרטגיות אלו (כמייצגות משערך) ביחס למשערך הבייסיאני האופטימלי אשר היה מתאפשר לו היתה גישה מלאה למידע הפרטי של כל הסוכנים ולהיסטוריית הפעולות.

לבסוף, בחלק הראשון של התזה, נבחנת סביבת Ad Exchange – מנגנון ריכוזי למכירה פומבית למפרסמים (בזמן אמת) של הזדמנויות פרסום לגולשים באתרי אינטרנט. תוך התמקדות בנקודת המבט והאתגרים של ה Ad Network, יישות האמונה על ההוצאה לפועל של מסעות פרסום ע"י זכיה בהזדמנויות פרסום לאוכלוסיית יעד מוגדרת מראש, הוגדר ומומש משחק TAC חדש TAC-Ad Exchange. סוכנים שממומשו ע"י קבוצות סטודנטים לתחרויות שהתקיימו ב 2014 מאפשרים לגבש תובנות לגבי האסטרטגיות הנדרשות בתרחיש זה – מעמודי התווך של כלכלת האינטרנט.

מופע נוסף של חוסן, הנחקר בחלק השני של התזה, הוא בהקשר לבעיית ההסתגלות (Domain-Adaptation) – התרחיש בו תוצר אלגוריתם למידה חישובית (למשל, תכנית לסווג) מצופה לפעול בסביבת "יעד" בעלת מאפיינים שונים ביחס לסביבת ה"מקור" (בה התרחשה הלמידה - סביבת ה"אימון"). תרחישים כאלו נפוצים מאוד למעשה (למשל בזיהוי דיבור, ראיה ממוחשבת, עיבוד שפה טבעית, ועוד) ומכאן החשיבות הרבה בכימות הפגיעה בביצועים (באמצעות חסמי הכללה) ובשיטות להתאמת אלגוריתמי למידה קיימים לתרחישים אלו.

אנו עושים שימוש בגישת אופטימיזציה-חסינה (Robust Optimization) ובתכונות של חסינות-אלגוריתמית (Algorithmic Robustness) כדי להשיג חסמי הכללה חדשים לבעיית הסתגלות והתאמה של אלגוריתמי למידה קלאסיים לסווג ורגרסיה (תוך שימוש ב-SVM) לתרחישי הסתגלות. אנו מציעים אלגוריתם למידה לבעיית ההסתגלות על ידי עיצוב מובנה של תכונות חסינות אלגוריתמית. גישה גנרית זו עשויה לשמש להתאמה של אלגוריתמי למידה קלאסיים אחרים לבעיית ההסתגלות. באופן טבעי, חסמים והתאמות כאמור תלויים בתכונות סטטיסטיות של ההבדלים בין סביבות המקור והיעד. אנו מציעים אמת מידה לכימות הבדלים אלו.

החלק השלישי והאחרון בתזה עוסק ב*למידה חברתית* (Social Learning) – שיטות לשיתוף פעולה בין סוכנים תחרותיים לצורך חישוב התלוי במידע פרטי המבוזר בין הסוכנים. במיוחד, אנו חוקרים תהליך סדרתי בו כל סוכן פועל (כלומר, מעדכן את החישוב) לאור תוצאת פעולת הסוכן שלפניו והמידע הפרטי שברשותו. תהליכים כאלו מתאימים לתיאור שווקי תחזיות (בהם הסוכנים סוחרים באופציות לסיכויי התרחשות מאורע עתידי, על בסיס מידע פנימי פרטי לכל סוחר) ולתהליכי המלצות המבוססות על אמון (בהם כל סוכן מקבל מקודמו המלצה לגבי איכות מוצר, ומעביר המלצה מעודכנת לבא אחריו לאור נסיונו). ניתן לראות בתוצאת החישוב במקרים אלו כצבירה של המידע הפרטי שבידי הסוכנים ("חכמת ההמון"), למשל כמשערך הציבורי להסתברות התרחשות מאורע עתידי.

אנו מציגים מודל של למידה חברתית נטולת היסטוריה ובו לכל סוכן מרחב אסטרטגיות פשוט המגדיר את פעולת העדכון (של מצב החישוב הנוכחי) על בסיס המידע הפרטי שלו והמצב הנוכחי בלבד (כלומר, ללא נגישות להיסטורית העדכונים, ובפרט לכמות העדכונים שכבר נעשו). עבור כללי ניקוד (scoring

תמצית

התפתחות האינטרנט הביאה עמה הזדמנויות עסקיות ואתגרים נלווים, רבים מהם הניתנים להצגה כבעיות למידה, כלומר, הכללה על בסיס תצפיות: תבניות התנהגות והעדפות הגולשים באתרים, ההצעות של המפרסמים במכרזים לפרסומות, סווג תכני משתמשים, ועוד. אלגוריתמי למידה מניחים מודל – של הסביבה בה הם פועלים ונבחנים, וביצועיהם תלויים במידת ההתאמה בין המודל והמציאות. *החוסן* של אלגוריתם למידה הוא מידת רגישותו לאי התאמות שכאלו. מחקר זה בוחן היבטי חוסן של אלגוריתמי למידה חישובית בהקשרים שונים.

מושא המחקר בחלק הראשון של התזה - תרחישים הדורשים מידול ואופטימיזציה בסביבה מרובת סוכנים (למשל, במצבי תחרות בין סוכנים – בהם נדרש סוכן להערכת מצב על בסיס למידת התנהגות הסוכנים היריבים והשפעתם, וקבלת החלטה לפעולה אופטימלית), מהווים מופע של חוסן נדרש כאמור. בתרחישים כאלו, הערכת מצב המבוססת על מודל מוטעה עלולה להוביל להחלטות אומללות.

המטרה הראשונה בתזה, בהקשר זה, היא מימוש סוכנים לומדים, ובמיוחד סוכנים לתחרות Trading Agent Competition (TAC) – סביבה מבוקרת להערכת אסטרטגיות להוצאה לפועל של מסעות פרסום במנועי חיפוש. סוכנים הממומשים בשיטות למידה חישובית *נטולות מודל* נמנעים מסיכוני אי-התאמת מודל ומצופים להיות עמידים ליריבים עויינים בסביבה מורכבת. סוכנים כאלו גם פשוטים יותר למימוש. בתזה אנו בודקים תכונות וביצועי סוכן שמימשנו בגישה נטולת מודל לעומת אלו של סוכנים מתחרים ב TAC שמומשו בגישות שונות (האמורות להשיג ביצועים משופרים תוך הסתמכות על המבנה והפרמטרים הספציפים בסביבה המסומלצת בתחרות). הוספה הדרגתית של מנגנונים תלויי מודל לסוכן שלנו לאורך שנות ההשתתפות בתחרות (לראשונה בשנת 2010, ועד זכיה במקום הראשון בתחרות TAC-Ad Auctions בשנת 2013) מאפשרת לכמת את הערך של שימוש בשיטות אלו. כמו כן, אנו מבססים את החוסן של שיטות נטולות מודל ע"י הדגמת החשיבות השולית היורדת של שימוש במודלים מדוייקים בסוכן עתיר הביצועים שלנו.

בתזה נבדק החוסן כאמצעי להערכת הרלוונטיות של האסטרטגיות המשמשות סוכנים בסביבה המבוקרת של תחרויות דוגמת TAC לתרחישים במציאות המורכבת הרבה יותר בדרך כלל. ע"י שינוי בפרמטרי ההדמיה בתחרות (ללא יידוע הסוכנים המתחרים) הודגם (במפתיע) שהסוכנים המובילים בתחרות המקורית עמידים יותר לשינויים לא צפויים בסביבה. ממצאים אלו תומכים באפשרות לשימוש בתובנות לגבי אסטרטגיות ומנגנונים מתחרויות בסביבה פשטנית כגון TAC עבור תרחישים מורכבים במציאות.

אוניברסיטת תל-אביב TEL AVIV UNIVERSITY

הפקולטה למדעים מדויקים ע"ש ריימונד ובברלי סאקלר

בית הספר למדעי המחשב ע"ש בלבטניק

אלגוריתמי למידה חישובית

וחוסן

חיבור לשם קבלת תואר "דוקטור לפילוסופיה"

מאת

מריאנו שיין

בהנחייתו של

פרופסור ישי מנצור

הוגש לסנאט של אוניברסיטת תל אביב

ינואר 2015