TEL-AVIV UNIVERSITY
RAYMOND AND BEVERLY SACKLER
FACULTY OF EXACT SCIENCES
THE BLAVATNIK SCHOOL OF COMPUTER SCIENCE

Learning Decision Trees with Stochastic Linear Classifiers

Thesis submitted in partial fulfillment of the requirements
for the M.Sc. degree of Tel-Aviv University
by

Tom Jurgenson

The research work for this thesis has been carried out at
Tel-Aviv University under the supervision of

Prof. Yishay Mansour

June 2017

Acknowledgments

First and foremost, I would like to thank my supervisor, Prof. Yishay Mansour, who was extremely supportive all throughout this thesis, who was always available for consultation and provided me with superb guidance and insights.

I would also like to thank my partner Alona and my supportive family (Summer included) who consistently helped me throughout this process - your support certainly made it a lot smoother than what it could have otherwise been.

Abstract

We consider learning decision trees in the boosting framework, where we assume that the classifiers in each internal node come from a hypothesis class H_I which satisfies the weak learning assumption.

In this work we consider the class of stochastic linear classifiers for H_I, and derive efficient algorithms for minimizing the Gini index for this class, although the problem is non-convex. This implies an efficient decision tree learning algorithm, which also has a theoretical guarantee under the weak stochastic hypothesis assumption.

Contents

1 Introduction **5**

2 Related Work **8**

3 Model **10**

4 Decision Tree Learning Algorithms **12**

5 Weak Learning and Stochastic Decision Tree Algorithms **16**

6 Stochastic Linear Classifier **18**

7 Approximately minimizing the Gini Index efficiently **21**

7.1 Algorithms for the linearly independent case: overview 24

7.2 Algorithm `FixedPQ` - Iterating over pairs of P_w and Q_w 25

7.3 Algorithm `FixedPSearchQ` - Iterating over values P_w 27

7.4 Algorithm `DependentWGI` - Dependent case 27

7.5 Summary 28

8 Empirical Evaluation **29**

8.1 Validating Strong Learnability 29

8.2 Comparison to other classifiers 33

8.3 Real Data-sets 35

9 Conclusion **37**

A Decision tree learning algorithm: pseudo code **40**

B Proof of Theorem 5.0.1 **43**

C Bounding the WGI by regions **46**

C.1 Partitioning the WGI to monotone regions 46

C.2 Overview of the partition to cells 48

C.3 First region: $R_{b,l}$ 49

C.4 Second region: $R_{b,r}$ 50

C.5 Third region $R_{a,l}$ 52

C.6 Fourth region $R_{a,r}$ 53

C.7 WGI is continuous in $p = 0$ and $p = 1$ 54

C.8 Proof conclusion 56

C.9 Solution for the dependent case 56

D Maximizing Information Gain for P_w and Q_w **58**

D.1 Values of P_w and Q_w are given 59

D.2 Values of P_w and Q_w are within specific ranges 60

D.3 Given value P_w search over Q_w 63

D.4 Feasible Ranges for P_w and Q_w 66

E Approximation Algorithms **67**

E.1 `FixedPQ` Approximate minimal Gini Index in cells of P_w, Q_w . . 68

E.2 `FixedPSearchQ` Approximate minimal Gini Index in range of P_w 71

E.3 `DependentWGI` Dependent constraints case 74

F WGI not convex **76**

G Approximate γ Margin With a Stochastic Tree **78**

H Splitting criteria bounds the classification error **81**

List of Figures

1.1 Example of decision tree with stochastic linear classifier for learning an XOR like function. 7

7.1 Partition P_w to slices for $\rho \geq 0.5$ 25
7.2 Partition (P_w, Q_w) to trapezoid cells for $\rho \geq 0.5$ 25

8.1 Single Hyperplane Decisions 30
8.2 Hyperplane with noise. 35
8.3 Multi-dimensional XOR 35

C.1 The domain R and the sub-domains R_a and R_b (for $\rho \geq 0.5$) . . . 48
C.2 WGI: partitioned into regions for $\rho \geq 0.5$ 48

Chapter 1

Introduction

Decision trees have been an influential paradigm from the early days of machine learning. On the one hand, they offer a "humanly understandable" model, and on the other hand, the ability to achieve high performance using a non-convex model. Decision trees provide competitive accuracy to other methodologies, such as linear classifiers or boosting algorithms. In particular, decision forests provide state of the art performance in many tasks.

Most decision tree algorithms used in practice have a fairly similar structure. They provide a top-down approach to building the decision tree, where each internal node is assigned a hypothesis from some hypothesis class H_I. The most notable class is that of *decision stumps*, namely, a threshold over a single attribute, e.g., $x_i \geq \theta$. The decision tree algorithm decides which hypothesis to assign in each internal tree node, to the most part they perform it by minimizing locally a convex function called *splitting criteria*. Decision tree algorithms differ in the splitting criteria they use, for example, the Gini index is used in CART [BFSO84] and the binary entropy is used in ID3 [Qui86] and C4.5 [Qui93].

From a computational perspective, proper learning of decision tree is hard [ABF+08]. However, assuming the weak learner hypothesis, as is done in boosting, [KM99] show that decision tree algorithms do indeed boost the accuracy of the basic hypotheses. Specifically, they show that for a variety of splitting indices, in-

cluding Gini index and binary entropy, if one assumes the weak learner assumption, the resulting decision tree will achieve boosting and guarantee strong-learning.

This work looks at the variation of extending the decision tree algorithms by allowing a linear classifier at each node. Such decision trees are named *oblique decision trees*, originally proposed in CART-LC and developed by [MKS94] and [HKS93], and extended in a variety of directions [BB98, FAB08, RLSCH12, GMD13, WRR+16]. However, a clear downside of oblique decision trees is that even minimizing the splitting criteria in an single internal node becomes a non-convex problem, and the various algorithms resort to methods which converge to a local minima. From a complexity point of view, oblique decision trees with three internal nodes already encodes three-node neural networks, which is computationally hard to learn [BR93].

Our Contributions: Our main goal is to perform an efficient computation for selecting a hypothesis in each internal tree node, namely, being able to efficiently minimize the splitting criteria. We are able to perform this task efficiently for the related class of *stochastic linear classifiers*. A stochastic linear classifier has a weight vector w and given an input x it branches right with probability $(x^\top w + 1)/2$, where we assume that x and w are unit vectors. We show how to efficiently minimize the Gini index for stochastic linear classifiers. We use our efficient Gini index minimization as part of an overall decision tree learning algorithm. We show, similar to [KM99], that under the weak stochastic hypothesis assumption our algorithm achieves boosting and strong-learning.

The stochastic linear classifier has a few advantages over other models:

1. Unlike decision trees with decision stumps, it can produce more general decision boundaries which may depend on many attributes.

2. Unlike linear classifiers, it produces a hierarchal structure that allows to fit non-separable data. (See Figure 1.1 for an illustration of how it fits a non-separable input.)

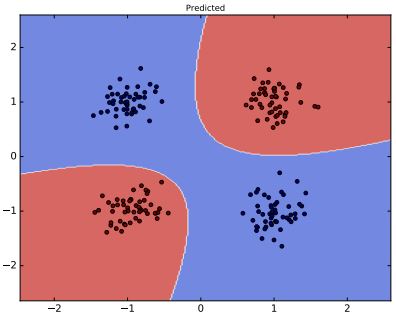

Figure 1.1: Example of decision tree with stochastic linear classifier for learning an XOR like function.

3. Unlike oblique tree, it can efficiently minimize the splitting criteria for a single node.

On the downside, since the model becomes more complicated it also becomes less interpretable than either linear classifiers or decision trees which are based on decision stumps.

Chapter 2

Related Work

Decision tree algorithms are one of the popular methods in machine learning. Most decision tree learning algorithms have a natural top-down approach. Namely, start with the entire sample, select a leaf and a hypothesis and replace the leaf by an internal node that splits using this hypothesis. A main difference between different decision tree algorithms is in the class of hypotheses H_I used and how they select the splitting hypothesis, e.g., the splitting criteria used.

The two most popular decision trees algorithms are C4.5 by [Qui93] and CART by [BFSO84]. Both use *decision stump* as hypotheses in internal nodes. (A decision stump compares an attribute to a threshold value, e.g., $x_i \geq \theta$.) The difference between the two algorithms is the specific splitting criteria $G(x)$, where x is the probability of positive examples within the node. CART uses the Gini index, i.e., $G(x) = 4x(1-x)$, while C4.5 uses the binary entropy, i.e., $G(x) = -x \log x - (1-x) \log (1-x)$. One significant benefit of decision stumps is that the size of the hypothesis class, given a sample of size m, is only dm, where d is the number if attributes. Since for each attribute we have at most m distinct values in a sample of size m.

The idea of using linear classifiers for H_I, the hypotheses class of the internal nodes , date back to CART-LC. A clear drawback is that the size of this hypothesis class is no longer linear in the dimension, but infinite, and minimizing the splitting

criteria is potentially a hard computational task. The original proposal of CART-LC suggested to do a simple gradient decent, which deterministically reaches a local minima. One of the successful implementation of oblique decision trees is OC1 by [MKS94], which uses a combination of hill climbing and randomization to search for a linear classifier of each internal node. Alternative approaches use simulated annealing by [HKS93] or evolutionary algorithms by [CK03], to generate a linear classifier in each internal node. The work of [BB98] build a three internal-nodes decision tree using a non-convex optimization which also maximizes the margins of the linear classifiers in the three nodes. The work of [WRR+16] uses the eigenvector basis to transform the data, and uses decision stumps in the transformed basis (which are linear classifiers in the original basis).

Even though decision tree algorithm are tremendously popular, due to their efficiency and simplicity, they are challenging to analyze theoretically. [KM99] introduced a framework to analyze popular decision tree algorithms based on the Weak Hypothesis Assumption of [Sch90]. The Weak Hypothesis Assumption states, that for any distribution, one can find a hypothesis in the class that achieves better than random guessing. The main result of their work shows that decision trees can be used to transform weak learners to strong ones. Qualitatively, assuming the weak learners always have bias at least γ, decision trees provably achieve an error of ϵ with size of $(1/\epsilon)^{O\left((\gamma\epsilon)^{-2}\right)}$, for the Gini index, and $(1/\epsilon)^{O\left((\gamma)^{-2}\log\left(\epsilon^{-1}\right)\right)}$, for the binary entropy. They also introduce an new splitting criteria which guarantees a bound $(1/\epsilon)^{O\left((\gamma)^{-2}\right)}$ which is polynomial in $1/\epsilon$. We extend the framework of [KM99] to encompass stochastic linear classifiers, which we use as our hypothesis class for internal decision tree nodes.

Chapter 3

Model

Let $X \subset \mathbb{R}^d$ be the domain and $Y = \{0, 1\}$ be the labels. There is an unknown distribution D over $X \times Y$ and samples are drawn i.i.d. from D. Given a hypothesis class H the error of $h \in H$ is $\epsilon(h) = \Pr[h(x) \neq y]$ where the distribution is both over the selection of $(x, y) \sim D$ and any randomization of h.

Given two hypothesis classes H_I and H_L mapping X to $\{0, 1\}$ we define a class of decision tree $\mathcal{T}(H_I, H_L)$ as follows. A decision tree classifier $T \in \mathcal{T}(H_I, H_L)$ is tree structure such that each internal node v contains a splitting function $h_v \in H_I$ and each leaf u contains a labeling function $h_u \in H_L$. In order to classify an input $x \in X$ using the decision tree T, we start at the root r and evaluate $h_r(x)$. Given that $h_r(x) = b \in \{0, 1\}$ we continue recursively with the subtree rooted at the child node v_b. When we reach a leaf l we output $h_l(x)$ as the prediction $T(x)$. In case that H_I includes randomized hypotheses we take the expectation over their outcomes. We denote by $\epsilon(T)$ the error of the tree, and the set of leaves in the tree as $Leaves(T)$. We denote the event that input $x \in X$ reaches node $v \in T$ with the predicate $Reach(x, v, T)$, and when clear from the context we will use $Reach(x, v)$.

Splitting criterion Intuitively, the splitting criterion G, is a function that measures how "pure" and large a node is. This function is used to rank various possible

splits of a leaf v using a hypothesis from H_I. Formally G is a *permissible splitting criterion*: if $G : [0, 1] \to [0, 1]$ and has the following three properties:

1. G is symmetric about 0.5. Thus, $\forall x \in [0, 1] : G(x) = G(1 - x)$.
2. G is normalized: $G(0.5) = 1$ and $G(0) = G(1) = 0$.
3. G is strictly concave.

Three well known such functions are:

1. Gini index: $G(q) = 4q(1 - q)$
2. The binary Entropy: $G(q) = -q \log(q) - (1 - q) \log(1 - q)$
3. sqrt criterion: $G(q) = 2\sqrt{q(1 - q)}$.

One can verify that any permissible splitting criteria G has the property that $G(x) \geq 2 \min\{x, 1 - x\}$ and therefore upper bounds twice the classification error (see appendix H).

Chapter 4

Decision Tree Learning Algorithms

The most common decision tree learning algorithms have a top-down approach, which continuously split leaves in a greedy manner. The algorithms work in iterations, where in each iteration one leaf is split into two leaves using a hypothesis from H_I. The decision which leaf to split and which hypothesis from H_I to use to split it is governed by the splitting criteria G.

Let us start with a few definitions which will help clarify the role of the splitting criteria G. For simplicity we will describe it using the distribution over examples, and later we will explain how it works on a sample. Fix a given decision tree T. We define the probability of reaching node $v \in T$ as:

$$\texttt{weight}(v) = \Pr_{(x,y)\sim D}(Reach(x, v))$$

and its purity as:

$$\texttt{purity}(v) = \Pr_{(x,y)\sim D}(y = 1 | Reach(x, l))$$

which is the probability of a positive label conditioned on reaching node v.

The score of the splitting criterion G with respect to a decision tree T is defined

as follows,

$$G(T) = \sum_{l \in Leaves(T)} \texttt{weight}(l) \cdot G(\texttt{purity}(l)). \tag{4.1}$$

This is essentially the expected value of $G(\texttt{purity}(l))$ when we sample a leaf l using the probability distribution D over the tree T. Recall that G is an upper bound on the error, and therefore $G(T)$ is an upper bound on the error using T. This explains the motivation of minimizing $G(T)$.

The decision tree algorithms iteratively split leaves, with the goal of minimizing the score of the tree. Let $Split(T, l, h)$ be the operation that splits leaf l using hypothesis h. The inputs of $Split(T, l, h)$ are: T a tree, $l \in Leaves(T)$ a leaf and a hypothesis $h \in H_I$. The output of $Split(T, l, h)$ is a tree T' where leaf l is replaced by an inner node v which has a hypothesis h and two children v_0 and v_1, where v_0 and v_1 are leaves. When using the resulting tree, and input x that reaches v then $h(x) = b$ implies that x continues to node v_b. Notice that the split is local, it influences only inputs that reached leaf l in T.

The selection of which leaf and hypothesis to use is based on greedily minimizing $G(T)$. Consider the change in $G(T)$ following $Split(T, l, h)$

$$\begin{aligned} \Delta_{l,h} =& G(T) - G(Split(T, l, h)) \\ =& \texttt{weight}(l) \cdot G(\texttt{purity}(l)) - \sum_{b \in \{0,1\}} (\texttt{weight}(v_b) \cdot G(\texttt{purity}(v_b))) \end{aligned} \tag{4.2}$$

where v_0 and v_1 are the leaves that result from split leaf l. The algorithm would need to find for each leaf l the hypothesis $h \in H_I$ which maximizes $\Delta_{l,h}$. This can be done by considering explicitly all the hypothesis in H_I, in the case where H_I is small, or by calling some optimization oracle. For now, we will abstract away this issue and assume that there is an oracle that given l returns $h \in H_I$ which maximizes $\Delta_{l,h}$, hence minimize $G(T)$.

We now describe the iterative process of building the decision tree. Initially, the tree T is comprised of just the root node r. We split node r using the hypothesis

h which maximizes $\Delta_{r,h}$ (thus minimizing $G(T)$). In iteration t, when we have a current tree T_t, we consider each leaf l of T_t and compute the hypothesis $h_l \in H_I$ which maximizes $\Delta_{l,h}$ using the oracle. We select the leaf that can guarantee the largest decrease, i.e., $l_t = \max \arg_{l,h} \Delta_{l,h}$, and set $T_{t+1} = Split(T, l_t, h_{l_t})$.

When the decision algorithm terminates it assigns a hypothesis from H_L to each leaf. Many decision tree learning algorithms simply use a constant hypothesis which is a label, i.e., $\{0, 1\}$. Our framework allows for a more elaborate hypothesis at each leaf, we only assume that the error of the hypothesis at leaf l is at most $\min\{x, 1-x\}$ where $x = \texttt{purity}(l)$.

Using a sample: Clearly the decision tree algorithms do not have access to the true distribution D but only to a sample S. Let S be a sample of examples sampled from D (S is a multiset) and let D_S be the empirical distribution induced by S. We simply redefine the previous quantities using the empirical distribution D_S. Specifically,

$$\texttt{weight}(v) = \Pr_{(x,y)\sim D_S}(Reach(x,v)) = \frac{|\{(x,y) \in S : Reach(x,v) = TRUE\}|}{|S|}$$

$$\texttt{purity}(v) =$$

$$\Pr_{(x,y)\sim D_S}(y = 1 | Reach(x,l)) = \frac{|\{(x,y) \in S : Reach(x,v) = TRUE, y = 1\}|}{|\{(x,y) \in S : Reach(x,v) = TRUE\}|}$$

As a result of this change the expressions for $G(T)$ and $\Delta_{l,h}$ in equations (4.1) and (4.2) are maintained. This allows the use of the Top-Down approach as specified without any further modifications.

Stochastic Hypothesis: We would like to extend the setting to allow for stochastic hypothesis. A major difference between the stochastic and deterministic case is that in the deterministic case the tree partitions the inputs (according to the leaves) while in the stochastic case we have only a modified distribution over the input induced by the leaves. This implies that now, given a tree T, each input x is associated with a distribution over the leaves of T. This requires modifications to the

basic decision tree learning algorithm, most notably maintaining the distribution over the leaves, per input x. The probability of reaching a node v, whose a path in the tree is $\langle (h_1, b_1)...(h_k, b_k) \rangle$ where $h_i \in H_I$ and $b_i \in \{0, 1\}$, is

$$\begin{aligned} \Pr_{(x,y)\sim D}(Reach(x, v)) &= \Pr\left(\forall i \in [1, k] :\ h_i(x) = b_i\right) \\ &= \prod_{i=1}^{k} \Pr\left(h_i(x) = b_i | h_j(x) = b_j, j < i\right) \end{aligned}$$

For this reason the modified decision tree learning algorithm keeps for each input x a distribution over the current leaves of T (In contrast, in the deterministic case, each input x has a unique leaf). When evaluating the reduction in the splitting criteria G, at a leaf v, we compute the distribution over the sample, which is associated with the current node v (in contrast, in the deterministic case, each node v is associated with a subset of the inputs).

A detailed description of the top-down decision tree learning algorithm `DTL` for a stochastic hypotheses class is provided in Appendix A.

Chapter 5

Weak Learning and Stochastic Decision Tree Algorithms

In this section we extend the results about the boosting ability of deterministic decision trees to include stochastic classifiers in the internal nodes. Since we have stochastic hypothesis we need that the weak learner assumption would hold for the expected error. Next, we slightly modify the existing analysis and show that the same boosting bounds that hold in the deterministic case hold also in the stochastic case. We start by defining the weak learning assumption.

Assumption 5.0.1 (Weak Stochastic Hypothesis Assumption)**.** *Let f be any boolean function over the input space X. Let H be a class of stochastic boolean functions over X. Let $\gamma \in (0, 0.5]$. We say H γ-satisfies the Weak Stochastic Hypothesis Assumption with respect to f if for every distribution $\bar{D}$ over X, there exists an hypothesis $h \in H$ such that:*

$$\Pr_{x \sim \bar{D}} \left(f(x) \neq h(x) \right) \leq 0.5 - \gamma$$

where the probability is both over the distribution $\bar{D}$, and the randomness of h.

Similar to [KM99], we show the following theorem,

Theorem 5.0.1. Let H be any class of stochastic boolean functions over input space X. Let $\gamma \in (0, 0.5]$, and let f be any target function such that H γ-satisfies the Weak Stochastic Hypothesis Assumption with respect to f. Let D be any target distribution, and let T be the tree generated by $DTL(D, f, G, t)$ with t iterations. Then for any target error ϵ, the error $\epsilon(T)$ is less than ϵ provided that:

$$t \geq \left(\frac{1}{\epsilon}\right)^{c/(\gamma^2\epsilon^2 \log(1/\epsilon))} \qquad \text{if } G(q) = 4q(1-q)$$

$$t \geq \left(\frac{1}{\epsilon}\right)^{c \log(1/\epsilon)/\gamma^2} \qquad \text{if } G(q) = H(q)$$

$$t \geq \left(\frac{1}{\epsilon}\right)^{c/\gamma^2} \qquad \text{if } G(q) = 2\sqrt{q(1-q)}$$

The above theorem establishes a tradeoff between the desired error ϵ, the bias parameter γ of the Weak Stochastic Hypothesis Assumption, the splitting criteria G and the decision tree size t. The proof is provided in Appendix B.

Chapter 6

Stochastic Linear Classifier

In this section, we describe the stochastic linear classifier which we use for the hypothesis class H_I, which are the hypotheses used in the internal nodes of the tree. We will also derive some basic properties which will be useful for our derivations.

Let $S = \{\langle x_i, y_i \rangle\}$ be a sample of size N drawn from D. Let D_v be a distribution over S, i.e., $D_v(x_i) \geq 0$ and $\sum_{i=1}^{N} D_v(x_i) = 1$. (We will have various distributions over the sample S, when we consider different nodes of the tree.) We assume that the inputs x_i are normalized to norm 1, i.e., $\|x_i\|_2 = 1$. We denote the weight of the positive examples according to D_v by ρ, i.e., $\rho = \sum_{i=1}^{N} y_i D_v(x_i)$.

For the *Stochastic Linear Classifier* we will have a weight vector w, which we assume is also normalized to norm at most 1, i.e., $\|w\|_2 \leq 1$. Our classification would induce a probability

$$\Pr[y = 1|x, w] = \frac{w \cdot x + 1}{2}$$

Note that since $\|w\| \leq 1$ and $\|x\| = 1$ then $|w \cdot x| \leq 1$ and therefore $0 \leq \Pr[y = 1|x, w] \leq 1$.

We like to define two measures that w induces. The first is P_w, the weight of samples classified as positive by w. Formally,

$$P_w = \sum_{i=1}^{N} D_v(x_i) \frac{w \cdot x_i + 1}{2}. \tag{6.1}$$

Note that $0 \leq P_w \leq 1$, since $\sum_{i=1}^{N} D_v(x_i) = 1$ and $0 \leq \frac{w \cdot x_i + 1}{2} \leq 1$.

We define by Q_w the weight of positive labels in the samples classified by w as positive. Formally,

$$Q_w = \sum_{i=1}^{N} D_v(x_i) \frac{w \cdot x_i + 1}{2} y_i. \tag{6.2}$$

Clearly $0 \leq Q_w$ since all the terms are non-negative. We can upper bound Q_w by P_w, since $y_i \leq 1$:

$$Q_w = \sum_{i=1}^{N} D_v(x_i) \frac{w \cdot x_i + 1}{2} y_i \leq \sum_{i=1}^{N} D_v(x_i) \frac{w \cdot x_i + 1}{2} \cdot 1 = P_w$$

and we can bound Q_w by ρ since $\frac{w \cdot x_i + 1}{2} \leq 1$:

$$Q_w = \sum_{i=1}^{N} D_v(x_i) \frac{w \cdot x_i + 1}{2} y_i = \sum_{i=1 | y_i = 1}^{N} D_v(x_i) \frac{w \cdot x_i + 1}{2} \leq \sum_{i=1 | y_i = 1}^{N} D_v(x_i) = \rho$$

Finally, we can also lower bound Q_w by $P_w - (1 - \rho)$ since:

$$P_w - Q_w = \sum_{i=1}^{N} D_v(x_i) \frac{w \cdot x_i + 1}{2} (1 - y_i) \leq \sum_{i=1}^{N} D_v(x_i)(1 - y_i) = 1 - \rho$$

Therefore,

$$\max\{0, P_w - 1 + \rho\} \leq Q_w \leq \min\{P_w, \rho\} \ . \tag{6.3}$$

Some intuition about the above bounds: since Q_w is a subset of positive labeled samples it is bounded by ρ, and since it is a subset of the samples the classifier classifies as positive, it is bounded by P_w. Also, clearly it is lower bounded by 0 and also lower bounded by the fraction of positives samples minus the fraction of the negative predictions of the classifier, i.e., $1 - P_w$.

We can also write P_w and Q_w in vector notation, which sometimes would be more convenient. For this purpose we denote the distribution $D_v \in R^N$ as a vector over the sample inputs, $X \in R^{N \times d}$ the examples matrix and $y \in \{0, 1\}^N$ the labels of the examples. Using this notation we have:

$$P_w = \frac{1}{2} D_v^\top X w + \frac{1}{2} \tag{6.4}$$

$$Q_w = \frac{1}{2} \left(D_v \odot y \right)^\top Xw + \frac{\rho}{2}, \tag{6.5}$$

where the $\odot$ signal is element-by-element multiplication.

The values of P_w and Q_w play an important role in evaluating candidate classifiers w by plugging them into the splitting criteria, as explained in more details in the next section.

Remark. One may wonder what will be the stochastic effect on the linear classifier behavior. Note that if w is a deterministic linear classifier with margin γ, then it implies that when we use w as a stochastic linear classifier we "err" with probability $(1 - \gamma)/2$. We can amplify the success probability by having a tree of depth $O(\gamma^{-2} \log \epsilon^{-1})$ and reducing the error probability to ϵ. For more details please refer to appendix G.

Chapter 7

Approximately minimizing the Gini Index efficiently

In the previous section we presented the stochastic linear classifiers, which we will use to split internal nodes in the tree, i.e., the class H_I is the set of all stochastic linear classifiers with norm at most 1. We would like to use the generic top-down decision tree learning approach in order to greedily minimize the value of the splitting criteria, and thus minimizing indirectly the prediction error.

In this section we will discuss how we select a hypothesis from H_I to minimize the value splitting criteria. Recall that we like to maximize $\Delta_{l,w}$ for a given leaf l over all stochastic linear classifiers w. Given a leaf l, the distribution D_l over the samples that reach it, and the probability of the positive examples in D_l is ρ, we would like to find a w that maximizes the drop in the splitting criteria, namely

$$\begin{aligned}
&\arg\max_w \Delta_{l,w} \\
&= \arg\max_w \texttt{weight}(l) \left(G(\rho) - \left(P_w \cdot G\left(\frac{Q_w}{P_w}\right) + (1-P_w)\, G\left(\frac{\rho - Q_w}{1-P_w}\right)\right)\right) \\
&= \arg\min_w \left(P_w \cdot G\left(\frac{Q_w}{P_w}\right) + (1-P_w)\, G\left(\frac{\rho - Q_w}{1-P_w}\right)\right)
\end{aligned}$$

We notice that the splitting criteria value is comprised of two parts: the first, $G(\rho)$ is the current purity measure and it has the same value for every w, and

therefore does not influences the optimization (therefore it is dropped in the last line). The second part, is the purity induced by the split w, which we like to minimize.

In this work we will focus on the case Gini Index, GI, as the splitting criteria. Recall that $GI(x) = 4x(1-x)$, where x is the probability of positive example. We denote by WGI, Weighted Gini Index, the value of the Gini index after a split using w, namely $WGI_w = WGI(P_w, Q_w)$ where

$$\begin{aligned} WGI(P_w, Q_w) =& P_w GI\left(\frac{Q_w}{P_w}\right) + (1-P_w) GI\left(\frac{\rho - Q_w}{1-P_w}\right) \\ =& 4\left(\rho - \frac{Q_w^2}{P_w} - \frac{(\rho - Q_w)^2}{1-P_w}\right) \end{aligned} \tag{7.1}$$

Note that for $P_w = 0$ or $P_w = 1$, all the examples reach the same leaf, and therefore there is no change in the Gini index, i.e., $WGI(0, Q_w) = WGI(1, Q_w) = GI(\rho)$.

The function $WGI(p, q)$ is concave[1] (rather than convex). However, note that both inputs are a function of the weight vector w, we show in Appendix F that WGI is not convex in w. This implies that one cannot simply plug-in the Gini index and minimize over the weights w.

Our first step is to characterize the the structure of the optimal weight vector w.

Theorem 7.0.1. For any distribution D_l let $\vec{a} = D_v^\top X$, $\vec{b} = (D_v \odot y)^\top X$ and $w_l^* = \arg\min_{\{w:\|w\|\leq 1\}} WGI(P_w, Q_w)$. There exist constants α and β such that for $w = \alpha\vec{a} + \beta\vec{b}$, we have $\|w\| \leq \|w_l^*\|$ and both $P_{w_l^*} = P_w$ and $Q_{w_l^*} = Q_w$.

Proof. Let $P_{w_l^*} = p$ and $Q_{w_l^*} = q$. Let w' be the solution for the following optimization problem:

$$\begin{aligned} \arg\min \ & \|w\|_2^2 \\ & P_w = p \\ & Q_w = q \end{aligned}$$

[1] $\frac{\partial^2 WGI}{\partial^2 p} = -8\left(\frac{q^2}{p^3} + \frac{(\rho-q)^2}{(1-p)^3}\right) \leq 0.$

Clearly, $\|w'\| \leq \|w_l^*\|$, $P_{w'} = p, Q_{w'} = q$ and therefore w' is also an optimal feasible solution. Lemma D.1.1 in Appendix D.1 shows that the solution for this optimization problem is:

$$w' = \frac{(2q-\rho)(\vec{a}\cdot\vec{b}) - (2p-1)\|\vec{b}\|^2}{\|\vec{a}\|^2\|\vec{b}\|^2 - (\vec{a}\cdot\vec{b})^2}\vec{a} + \frac{(2p-1)(\vec{a}\cdot\vec{b}) - (2q-\rho)\|\vec{a}\|^2}{\|\vec{a}\|^2\|\vec{b}\|^2 - (\vec{a}\cdot\vec{b})^2}\vec{b}$$

which completes the proof. □

Note that $\vec{a} = D_v^\top X$ is the weighted average of the data (average over each feature) and $\vec{b} = (D_v \odot y)^\top X$ the weighted average of the positive examples (average over each feature). The above characterization suggests that we can search of an approximate optimal solution by using only linear combinations of $\vec{a}$ and $\vec{b}$. Our main goal is to find a weight vector w such that $WGI_w - WGI_{w_l^*} \leq \epsilon$. At a high level we perform the search over the possible values of P_w and Q_w. Our various algorithms perform the search in different ways, and get different (incomparable) running times. We start with defining the approximation criteria.

Definition 7.0.1. *Let S be a sample, D_l be a distribution over S, and ϵ be an approximation parameter. An algorithm guarantees an ϵ-approximation for WGI using a stochastic linear classifier w, such that $\|w\|_2 \leq 1$, if $WGI_w - WGI_{w^*} \leq \epsilon$, where $w^* = \arg\min_{\{w:\|w\|\leq 1\}} WGI_w$.*

The following theorem summarizes the running time of our various algorithms.

Theorem 7.0.2. Let S be a sample, D_v be a distribution over S, and ϵ be an approximation parameter. Let $\vec{a} = D_v^\top X$ and $\vec{b} = (D_v \odot y)^\top X$. Then:

1. For the case where $\vec{a}$ and $\vec{b}$ are linearly independent, algorithms `FixedPQ` and `FixedPSearchQ` guarantee an ϵ-approximation for WGI using a stochastic linear classifier. Algorithm `FixedPQ` runs in time $O(Nd) + O\left(\epsilon^{-2}\log\left(\epsilon^{-1}\right)\right)$, and algorithm `FixedPSearchQ` runs in time $O(Nd) + O\left(\frac{d}{\epsilon}\right)$.

2. For the case where $\vec{a}$ and $\vec{b}$ are linearly dependent, i.e., $\vec{b} = \lambda\vec{a}$, algorithm `DependentWGI` runs in time $O(Nd)$, and achieves the optimal result for WGI using a stochastic linear classifier.

The rest of this section is organized as follows. In section 7.1 we will give an overview common to the two algorithms for the linearly independent case, while sections 7.2 and 7.3, discuss each algorithm (full details can be found in Appendices E.1 and E.2). The differences in the running times between the two algorithms depend on the relation between d and ϵ. Section 7.4 provides an outline of algorithm `DependentWGI`, for the the linearly dependent case, i.e., $\vec{b} = \lambda\vec{a}$ (full details in Appendix E.3).

7.1 Algorithms for the linearly independent case: overview

The two algorithms we present, combine a search over the domain of WGI, where we discritize it. The domain includes pairs (p, q), where $p = P_w$ and $q = Q_w$. As we showed before, the domain includes only the subset $\{(p, q)|0 \leq p \leq 1, \max(0, \rho + p - 1) \leq q \leq \min(\rho, p)\})$. We note that the class of stochastic linear classifier also limits the domain, as noted in Appendix D.4, and we consider the intersection of these two domains. Each of our algorithms partitions the domain into cells such that each cell is represented by a finite (constant) number of candidate points. We will need to verify that the candidate points can indeed be generated by a feasible w (i.e., $\|w\|_2 \leq 1$). The algorithm returns the best results found, i.e., with the lowest WGI. We note that these algorithms return a weight vector of the form $w = \alpha\vec{a} + \beta\vec{b}$, which is similar to the characterization of the optimal weight vector, given in Theorem 7.0.1.

The two algorithms differ in the way they define the cells and the way they extract candidate points from these cells. However, both use the following theorem (proved in Appendix C):

Theorem 7.1.1. Given data distribution D_l which has a positive label probability of ρ and a parameter $\epsilon \in (0, 1)$ it is possible to partition the domain of WGI: $\{(p, q)|0 \leq p \leq 1, \max(0, \rho + p - 1) \leq q \leq \min(\rho, p)\}$ into $O\left(\epsilon^{-2}\right)$ cells, such

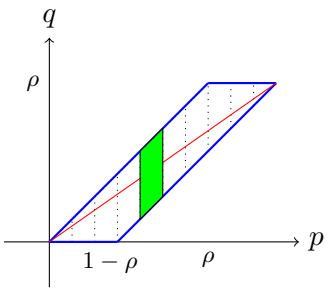

Figure 7.1: Partition P_w to slices for $\rho \geq 0.5$

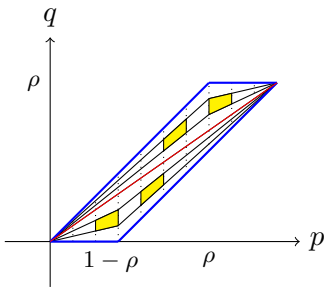

Figure 7.2: Partition (P_w, Q_w) to trapezoid cells for $\rho \geq 0.5$

that the if (p_1, q_1) and (p_2, q_2) belong to the same cell:

$$|WGI(p_1, q_1) - WGI(p_2, q_2)| \leq \epsilon \ .$$

Using the cell partition, we are able to search for solutions in the domain of the WGI function directly, and thus we derive an approximation of the optimal solution w^*. The different cell partitions are illustrated in Figures 7.2 and 7.1.

The main challenge is that the case of extreme values of p and their influence on the value of $WGI(p, q)$. If we ignore the dependence between the p and q then the derivatives of $WGI(p, q)$ are unbounded when $p \approx 0$ or $p \approx 1$. We overcome this issue by relating the value $p = P_w$ and $q = Q_w$ through their common parameter w, and the structure of the cells. This is what enables us to derive the approximation bound.

7.2 Algorithm `FixedPQ` - Iterating over pairs of P_w and Q_w

Algorithm `FixedPQ` partitions the domain of WGI to trapezoid cells. Note that the domain of WGI is a Parallelogram, and we partition it using a discritization of the p values. We like the derivatives of WGI to be monotone in each cell, which leads us to partition the entire domain to four sub-domains. Figure 7.2 illustrates the cells. (Details in Appendix C.)

Algorithm `FixedPQ` iterates over pairs of (p, q) which are the vertices of the trapezoid cells. For each such pair its $WGI(p, q)$ is evaluated, and the candidate (p, q) pairs are sorted from the lowest $WGI(p, q)$ to the highest. The sorted list is then scanned until a feasible stochastic linear classifier with those parameters is found, i.e., a weight vector w such that $\|w\| \leq 1$ and $p = P_w$ and $q = Q_w$.

Notice that computing the value of WGI and testing the feasibility of a pair (p, q) does not require the computation of the classifier w and can be done in $O(1)$ time. This follows since $w = \alpha\vec{a} + \beta\vec{b}$, and given the values of α, β, $\|\vec{a}\|$, $\|\vec{b}\|$ and $(\vec{a} \cdot \vec{b})$ we can compute the norm of w directly from those values in $O(1)$ time without computing the weight vector w explicitly. Computing the weight vector w requires $O(d)$ time. In `FixedPQ` a weight vector w is computed only once - for the first feasible candidate point (p, q) encountered in the scan.

The complexity Algorithm `FixedPQ` is $O(Nd) + O\left(\epsilon^{-2} \log\left(\epsilon^{-1}\right)\right)$, where $O(Nd)$ is for computing $\vec{a}$ and $\vec{b}$ and the $O\left(\epsilon^{-2} \log\left(\epsilon^{-1}\right)\right)$ is for sorting $O(\epsilon^{-2})$ candidate points.

Candidate generation: A list of (p, q) pairs is created by first selecting $O\left(\epsilon^{-1}\right)$ evenly spaced points that cover the feasible interval for P_w. The feasible interval is a subsection of $[0, 1]$ that is determined by the data (see Appendix D.4). We also force points at $p = \rho$ and $p = 1 - \rho$ if these coincide the feasible interval. This ensures that the cells generated are contained within a single sub-domain (more in Appendix C).

Next, for each discretized value of p we define $O\left(\epsilon^{-1}\right)$ evenly spaced values of q in the following two intervals: from $\max(0, \rho + p - 1)$ to ρp, and from ρp to $\min(p, \rho)$. Finally, the pairs are filtered to include only feasible values (as described in Appendix D.4). This process produces $O\left(\epsilon^{-2}\right)$ pairs.

Appendix E.1 provides pseudo-code, a correctness proof and complexity analysis for `FixedPQ`.

7.3 Algorithm `FixedPSearchQ` - Iterating over values P_w

Algorithm `FixedPSearchQ` iterates over values of p which are the endpoints of mutually exclusive slices of trapezoid cells (more in Appendix C). A slice of trapezoid cells, is the set of trapezoid cells which share the same p range (see Figure 7.1). These endpoints are the candidates in this case, and for each candidate at most two weight vector are computed. We filter the non-feasible points, and return the classifier w for the point that achieves the lowest WGI.

Notice that since we do not have the q values in the candidate stage we cannot compute the value WGI before computing a weight vector w or sort according to the WGI values as we did in `FixedPQ`. Instead we must compute a weight vector w for each candidate and return the best weight vector w. However, since we are only interested in the minimum WGI there is no need to sort. The complexity of this algorithm is $O(Nd) + O\left(\frac{d}{\epsilon}\right)$.

Candidate generation: The candidates are p values that are generated by selecting $O\left(\epsilon^{-1}\right)$ evenly spaced points that cover the feasible interval for P_w as in `FixedPQ`. The main difference is that we do not discretize the q values, but maintain a complete slice of all feasible q values for a given range of p values. (More in Appendix C).

Appendix E.2 provides pseudo-code, a correctness proof and complexity analysis for `FixedPSearchQ`.

7.4 Algorithm `DependentWGI` - Dependent case

Algorithm `DependentWGI` describes the algorithm for the case where the constraints over P_w and Q_w are dependent, i.e., $\vec{b} = \lambda\vec{a}$. Appendix C.9, shows that the optimal weight vector is $w = \pm\frac{\vec{a}}{\|\vec{a}\|}$, both attaining equal values of WGI. In order to compute this weight vector, we compute $\vec{a}$ and normalize it in $O(Nd)$ time. We

also compute the WGI value in $O(1)$ time. The total time complexity is $O(Nd)$. Appendix E.3 provides pseudo-code, a correctness proof and complexity analysis for `DependentWGI`.

7.5 Summary

We distinguish between the case that $\vec{a}$ and $\vec{b}$ are linearly independent and the case when they are linearly dependent. Algorithm `DependentWGI` finds the optimal solution for the case where $\vec{a}$ and $\vec{b}$ are linearly dependent, while the remaining two algorithms `FixedPQ` and `FixedPSearchQ` are for the linearly independent case, each using a different search method. Algorithm `FixedPQ` iterates over different possible values of (p, q) pairs and its complexity is: $O(Nd) + O(\epsilon^{-2} \log(\epsilon^{-1}))$. Algorithm `FixedPSearchQ` iterates only over a p candidates, and for each computes the optimal q value. The complexity of `FixedPSearchQ` is: $O(Nd) + O\left(\frac{d}{\epsilon}\right)$. The better running time depends on whether d is larger than $O(\epsilon^{-1} \log(\epsilon^{-1}))$ or not.

Finally, if we assume algorithms `DependentWGI`, `FixedPQ` and `FixedPSearchQ` find a weak learner in H_I as described by the Weak Stochastic Hypothesis Assumption 5.0.1 then we are able to boost the results using the `DTL` algorithm using Theorem 5.0.1. Formally the following theorem holds:

Theorem 7.5.1. Let H_I be the feasible stochastic linear classifiers over input space X. Let $\gamma \in (0, 0.5]$, and let f be any target function such that H_I γ-satisfies the Weak Stochastic Hypothesis Assumption with respect to f. Let D be any target distribution, and let T be the tree generated by $DTL(D, f, G, t)$ by using algorithms `DependentWGI`, `FixedPQ` and `FixedPSearchQ` as the local oracles for t iterations. Then for any target error ϵ, the error $\epsilon(T)$ is less than ϵ provided that:

$$t \geq \left(\frac{1}{\epsilon}\right)^{c/(\gamma^2 \epsilon^2 \log(1/\epsilon))} \qquad \text{if } G(q) = 4q(1-q)$$

Chapter 8

Empirical Evaluation

8.1 Validating Strong Learnability

In this section we test our classifier in learning scenarios to verify it captures the target function well. We demonstrate this ability with targets with increasing difficulty as well as unbalanced classes. All experiments were carried in the setting of a 2-dimensional space on random unit size vectors. The reported results are the average of 10 repetitions of each experiment, and the number of internal nodes in the tree is 15 unless otherwise noted.

Single Hyperplane: In the first experiment the concept class is a hyperplane which intersects the origin and both classes have equal weight. As figure 8.1 shows the algorithm constantly selects hyperplanes which are very close to the target function, and indeed as a result the accuracy of the trained model is around 0.99 which shows that the target is captured well.

Single Hyperplane and Artificial Bias: Next, we investigate the behavior when we add a bias term to the classifier. We model the bias term by adding a constant-valued feature to each example. We note that since the decision boundary intersects the origin there is no reason to add this bias term, however, we want to examine the

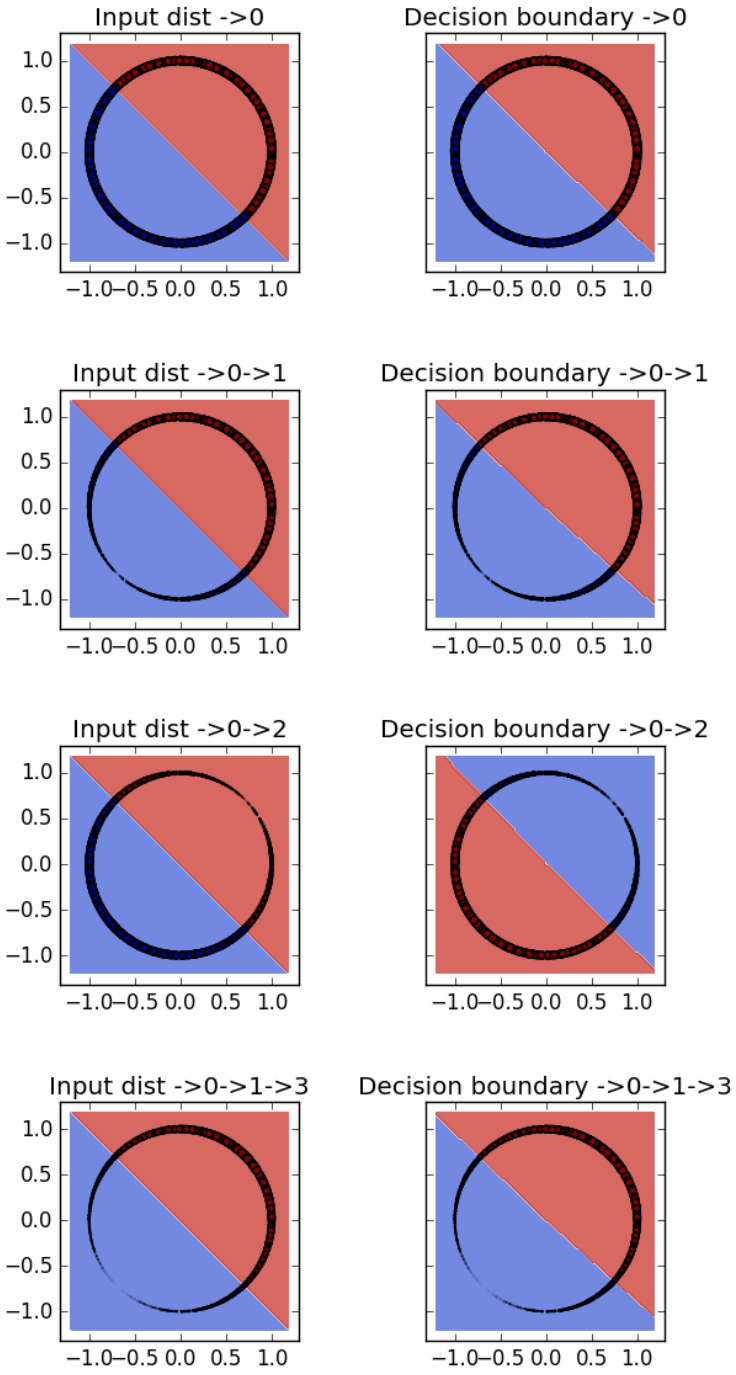

Figure 8.1: Single Hyperplane Decisions

The decisions taken by the proposed classifier. The title explains the position in the tree hierarchy (root is index 0). On the left the input distribution s.t the circle size is relative to sample weight and the color is based on the true label. On the right the decision of the internal node.

case where the classifier and the target are mismatched (in practice when the target function is unknown it may be beneficial to use a bias in different scenarios).

Now, the data contains an artificial constant that the classifier should theoretically ignore. A-priori, we know it is possible to cancel the effect of the bias: both $\vec{a}$ and $\vec{b}$ will have their last coordinate to be the value of the bias, the decision in each node of the tree - w is a linear combination of the form $w = \alpha\vec{a} + \beta\vec{b}$. Therefore, for the bias to be canceled out, the classifier needs to enforce $\alpha - \beta = 0$.

In practice, we indeed see a trend of the classifier to reduce the last coordinate to 0. However as shown in table 8.1, we also see a drop in the confidence when the bias is increased. This phenomena is explained if we consider the limitations we imposed on the stochastic linear classifier:

When the bias (denoted as b) is zero, the data vector x is exactly norm one i.e $1 = \sum x_i^2$. Since the stochastic linear classifier is restricted to consider input vectors of norm ≤ 1, once a bias $b > 0$ is introduced, all the true data coordinates (x_i) are normalized by a factor θ_b which is:

$$1 = \sum (\theta_b \cdot x_i)^2 + b^2 \rightarrow \theta_b = \frac{1}{\sqrt{1+b^2}}$$

Denote x_b and w_b the data and the classifier if the data is modeled with bias b. Even if we assume that $\forall b : w_0 = w_b$ (the algorithm completely ignores the bias) we still need to consider how the classifier output changes because of this added bias. Denote by $c = \frac{w_0 \cdot x_0 + 1}{2}$ (the result of the prediction without bias) thus $w_0 \cdot x_0 = 2c - 1$, the new prediction for bias $b > 0$ is

$$c_b = \frac{w_b \cdot x_b + 1}{2} = \frac{w_0 \cdot x_b + 1}{2} = \frac{\theta_b \cdot w_0 \cdot x_0 + 1}{2} = \theta_b(c - 0.5) + 0.5$$

The first transition is because we assumed that $w_0 = w_b$.

The confidence is the distance $c_b - 0.5$ and therefore for b is $\theta_b(c - 0.5)$ and since $\theta_b < 1$ we have that c_b is always smaller than the confidence for bias 0 (which is $c - 0.5$). In other words, even if we have the same stochastic linear classifier, we lose a factor of θ_b in confidence due to the restriction $\|x\| \leq 1$. We notice that by weakening the restriction to $\|w \cdot x\| \leq 1$ we could have avoided the reduction

Table 8.1: Bias effects on confidence

b bias	c_b positive mean	θ_b normalization coefficient
0	0.8	1
1	0.71	0.707
1.5	0.67	0.555
2	0.63	0.447
3	0.6	0.316
5	0.55	0.196
10	0.51	0.099

in confidence however it is unclear what effect it would have on our solution since the optimization solved in Appendix D.1 will no longer hold.

Table 8.1 shows the drop in prediction confidence in the first internal node as the bias changes. We note that in our tests the weight in the predictor that matched the bias was reduced to almost 0. The middle column shows that the prediction confidence was indeed reduced according to the relation described above.

Single Hyperplane with Unbalalnced Labels In the next test, we show strong learnability even for unbalanced classes. For this experiment we changed the ratio between classes from 1:1 we had in the previous experiments to 2:1 ratio between the classes. We observed that our classifier's accuracy dropped to 0.94. In order to understand this drop we considered the first split of the tree. We noticed that even though the accuracy dropped, when considering the soft accuracy i.e the average of $\|y_i - \hat{y}_i\|$ we see that both the "basic" balanced case and the unbalanced case both consistently reach to scores around 0.3. We expected the soft accuracy value to be comparatively low compared to the accuracy since we now care about the confidence of the classification and not the sign. We know that the stochastic linear classifier could potentially reach perfect confidence but it is rarely achieved in practice. Therefore the drop in accuracy is attributed to the fact that our classifier

tries to optimize the distances to the separating hyperplane and not the sign, but it still preforms well in terms of accuracy (as it still scores accuracy of 0.94).

Multiple Hyperplanes - XOR In the next set of experiments the target function is a XOR of the input hyperplanes (XOR is the product of the signs of $w \cdot x$ for every w in the target function). We chose this type of targets, since our classifier is an aggregation of linear decisions and should therefore capture this structure well.

In the first experiment we have two fixed hyperplanes. The reason the hyperplanes are fixed is to make sure the classes are balanced. In the second experiment, we again have two hyperplanes but now these are chosen randomly, which may result in very unbalanced classes (for instance if the two hyperplanes lie very very close to each other). In the last experiment we raise the number of hyperplanes to three.

As expected, the accuracy goes down as the target becomes more complex: in the first experiment the classifier has an accuracy of 0.94 while just changing to unbalanced classes in the second experiment causes the accuracy to drop to 0.9. The third experiment which has a more complex target achieves an initial accuracy of 0.81. For the last experiment we also tested adding more internal splits, raising the number of internal nodes from 15 (which we use as a default parameters) to 30 raises the accuracy to 0.9. Raising the number of nodes to 60 gives just 0.03 more to a total of 0.93. In general we see deceasing accuracy gains as we add more nodes.

Finally, considering all the above experiments we can conclude that the proposed classifier is able to capture these target concepts well (even non linear ones).

8.2 Comparison to other classifiers

This section discuss empirical evaluations of our classifier on synthetic data-sets we created with the motivation of understanding the benefit of our decision tree using stochastic linear classifiers, compared to decision tree with decision stumps, e.g.,

CART, and a standard linear classifier, e.g., linear SVM. We selected the simulated data, with the goal of highlighting the similarities and differences between the classifiers. In the following, we described the results of two simulated data sets.

In the first experiment we demonstrate that unlike standard decision tree algorithms, our method fits well a data-set which is labeled according a random linear boundary with noise in **a high dimension**. Specifically, we select a random hyperplane w and each sample x are random unit vectors, where the the label is $y = sign(w \cdot x)$ and dimension is $d = 100$. We add random classification noise, i.e., flip the label with probability of θ. We show the accuracy of our model compared to both CART and linear SVM as a function of the noise rate θ in Figure 8.2. Clearly, SVM is ideal for this task, and indeed it outperforms the other two classifiers. We ran both CART and our method to build decision trees of depth 10. The results show that our method achieves competitive performance to SVM while outperforming CART.

The second synthetic data set demonstrates the ability of our classifier to handle non-linearly separable examples, where the boundaries are not axis aligned. Again, all samples are random unit vectors of dimension d. The labels correspond to the XOR of the samples with two hyperplanes, which are perpendicular to each other and are not axis aligned. Specifically we take $w_1 = [\vec{1}_{d/2}, \vec{1}_{d/2}]$ and $w_2 = [\vec{-1}_{d/2}, \vec{1}_{d/2}]$ and the label is $y = sign(x \cdot w_1) \oplus sign(x \cdot w_2)$. Both decision trees were allowed to reach depth of 10 and the sample size is $10,000$. We used various dimension sizes from $d = 6$ to $d = 30$, and the results are plotted in Figure 8.3.

The results were in line with out intuition. First, Linear SVM achieves the worst performance since it does not have the expressive power required for representing this target function. Second, a decision tree algorithm, such as CART, which considers only a single attribute in each split, requires a significant depth to approximate this high-dimensional XOR. Finally, our method achieves a good accuracy since it is not limited to a single hyperplane, as is Linear SVM, nor is it limited to axis aligned decisions, as is CART. We do note, that since our method

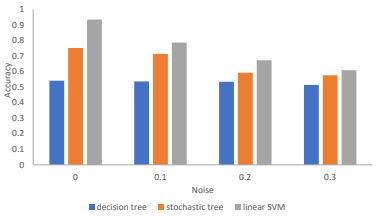

Figure 8.2: Hyperplane with noise.

Accuracy of the different classifiers for linearly separable points. In orange, our stochastic classifier, blue is CART and grey is linear SVM.

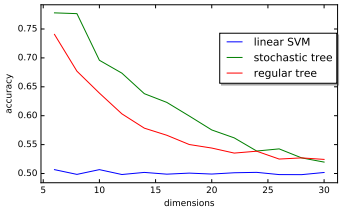

Figure 8.3: Multi-dimensional XOR

Unit vectors with norm 1, are labeled according to a multi dimensional non-axis aligned XOR. Our stochastic classifier (green) achieves much higher accuracy than both a linear SVM (blue) and CART (red).

is stochastic, we need to continuously repeat a hyperplane in order to make a split more significant. This explains why, due to the limited depth of the decision tree, our performance deteriorate as the dimension increases. An alternative is to limit the number of nodes in the decision tree. Further experiments show that our method can achieve the above performance with only 50 internal nodes (rather than a depth of size 10, which implies 1024 internal nodes).

8.3 Real Data-sets

We show the performance of our classifier compared to Linear SVM and CART on three data-sets that were taken from UCI dataset repository [Lic13]. The first dataset *Promoters* contains 106 examples with 57 categorical features and two classes. Since our algorithm runs on numeric data we encode each feature to it's one-hot representation for a total of 228 numerical features. The second dataset is the *Heart Disease Data Set* which contains 303 samples with 13 numeric features. In the original data there are 74 features, but it is common to use the above subset

of features. The original data contains one negative class to indicate no disease in the patient, and 4 positive classes each corresponding to a certain disease. We reduced the problem to a positive vs. negative classification, by combining all the diseases to one class. The third dataset *Statlog (Heart) Data Set*, has the same format as the previous only the target variable is already in binary format. There are 13 features and 270 samples in this dataset.

The following table shows the accuracy of all three classifiers used in our experiments, and demonstrates the advantage of our proposed method:

Dataset	CART	Linear SVM	Stochastic Tree
Promoters	0.8	0.87	**0.91**
Heart Disease Data Set	0.7	0.73	**0.77**
Statlog (Heart) Data Set	0.83	0.78	**0.85**

We note that we gave the same depth restriction to CART and our Stochastic Decision Tree, however, our method seems sometimes to require a smaller depth to converge. For the *Promoters* data-set, while CART was still improving at depth 10, while the stochastic tree only needed only a depth of 3 to converge. This along with the comparatively strong result of the Linear SVM may hint that the true decision boundary can be approximated well with a hyperplane which is not axis aligned. For the other two data-sets, *Heart Disease Data Set* and *Statlog (Heart) Data Set* both methods converged at the same depth of 3 and 5 correspondingly.

Chapter 9

Conclusion

In this work we introduced a stochastic decision tree learning algorithm based on *stochastic linear classifiers*. The main advantage of our approach is that it allows the internal splits in the node to have a global effect, on the one hand, and we can efficiently minimize the Gini index, on the other hand. From a theoretical perspective, we are able to maintain the basic decision tree learning result under the assumption of weak learning.

Our empirical finding are encouraging, and suggest doing a more comprehensive empirical analysis of the strengths and weaknesses of our method. Another possible research direction would be to explore the benefits of the stochastic linear classifier in other boosting frameworks such as AdaBoost.

From a more theoretical perspective we hope to extend our method to work efficiently for a large class splitting functions, under minimal assumptions on this class.

Another challenging research direction, is to change the linear modeling for $\Pr(y|x)$ to a more complex representation, such as a log linear model. This gives rise to an immediate computational challenge of efficiently minimizing the splitting criteria.

Bibliography

[ABF+08] Michael Alekhnovich, Mark Braverman, Vitaly Feldman, Adam R. Klivans, and Toniann Pitassi. The complexity of properly learning simple concept classes. *J. Comput. Syst. Sci.*, 74(1):16–34, 2008.

[BB98] Kristin P Bennett and Jennifer A Blue. A support vector machine approach to decision trees. In *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*, volume 3, pages 2396–2401. IEEE, 1998.

[BFSO84] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[BR93] Avrim Blum and Ronald L. Rivest. Training a 3-node neural network is NP-Complete. In *Machine Learning: From Theory to Applications - Cooperative Research at Siemens and MIT*, pages 9–28, 1993.

[CK03] Erick Cantú-Paz and Chandrika Kamath. Inducing oblique decision trees with evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 7(1):54–68, 2003.

[FAB08] Janis Fehr, Karina Zapién Arreola, and Hans Burkhardt. Fast support vector machine classification of very large datasets. In *Data Analysis, Machine Learning and Applications*, pages 11–18. Springer, 2008.

[GMD13] Dejan Gjorgjevikj, Gjorgji Madjarov, and Sašo Džeroski. Hybrid decision tree architecture utilizing local svms for efficient multi-label learning. *International Journal of Pattern Recognition and Artificial Intelligence*, 27(07):1351004, 2013.

[HKS93] David Heath, Simon Kasif, and Steven Salzberg. Induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2(2):1–32, 1993.

[KM99] Michael J. Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. *J. Comput. Syst. Sci.*, 58(1):109–128, 1999.

[Lic13] M. Lichman. UCI machine learning repository, 2013.

[MKS94] Sreerama K. Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *Journal of artificial intelligence research*, 1994.

[Qui86] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Qui93] J Ross Quinlan. C4. 5: Programming for machine learning. *Morgan Kauffmann*, page 38, 1993.

[RLSCH12] Irene Rodriguez-Lujan, Carlos Santa Cruz, and Ramon Huerta. Hierarchical linear support vector machine. *Pattern Recognition*, 45(12):4414–4427, 2012.

[Sch90] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[WRR+16] D.C. Wickramarachchi, B.L. Robertson, M. Reale, C.J. Price, and J. Brown. Hhcart. *Comput. Stat. Data Anal.*, 96(C):12–23, April 2016.

Appendix A

Decision tree learning algorithm: pseudo code

The pseudo code of Algorithm 1 encapsulates most existing decision tree learning (DTL) algorithm, including algorithms such as CART by [BFSO84] and C4.5 by [Qui93]. Our exposition of the decision tree algorithms tries to abstract the specific splitting function G and the way to minimize it. We also generalize the standard framework to handle stochastic classifiers. We start with describing the oracles we assume.

The first oracle minimizes the splitting criteria for a hypothesis in an internal node. The function $FitH_I(S, D, G)$ receives as input a sample S, and a distribution D over the sample S and a splitting criteria G. It returns (h^*, g^*) where $h^* \in H_I$ is the hypothesis that minimizes the splitting criteria G over the distribution D on S and g^* is its reduction in the value of the splitting criteria G.

The second oracle $FitH_L(S, D)$ receives as input a sample S, a distribution D over the sample S and returns a hypothesis $h \in H_L$ which minimizes the error (and not the splitting criteria G).

We have a function $SplitLeaf(T, l, h)$ which receives a decision tree T, a leaf $l \in Leaves(T)$, a hypothesis $h \in H_I$ and returns a new tree where we split leaf l using hypothesis h.

The decision tree learning algorithm runs t iterations. In each iteration, for each leaf $l \in Leaves(T)$ the function $FitH_I$ evaluates the optimal predictor h_l^* for the leaf l along with g_l^*, the local reduction in G induced by h_l^*. The local reduction is weighted by w_l to obtain the global reduction in G, i.e., $w_l g_l^*$, and we select the leaf that minimizes G globally. We update the tree by splitting this leaf, and for the new leaves update the probabilities of inputs to reach each of them. Once t iterations are done and the leaves have been determined, $FitH_L$ selects for each leaf a predictor from H_L by minimizing the error (and not the splitting criteria G).

One can observe that our framework encapsulates common decision tree methodologies algorithms, such as C4.5 and CART, as follows:

1. H_I is the family of all decision stumps and $H_L = \{0, 1\}$.

2. The function $FitH_I$ is implemented by evaluating all the decision stumps and selecting the best according to G.

3. The function $FitH_L$ sets $y = 0$ or $y = 1$ according to the majority class in the leaf, minimizing the error given a constant label predictor

We presented a general basic scheme of the *DTL* algorithm. We did not describe other hyperparameters (such as minimal split size) or methods to refine the model (such as pruning) which could be applied to the generic algorithm in order to improve results. Our goal is to focus of the core properties of the DTL which we believe are well represented in our framework.

Algorithm `DTL` (S - *sample,* G *splitting criteria,* t *number of iterations*)

Initialize T to be a single-leaf tree with node r, for $x \in S$: $p_r(x) = \frac{1}{|S|}$, $w_r = 1$, and $q_r = \frac{|\{x \in S : f(x) = 1\}|}{|S|}$.
while $|Leaves(T)| < t$ **do**
  $\Delta_{best} \leftarrow 0$; $best_l \leftarrow \perp$; $best_h \leftarrow \perp$
  for *each* $l \in Leaves(T)$*:* **do**
    $h_l^*, g_l^* \leftarrow FitH_I(S, \frac{1}{w_l} p_l(x), G)$
    $\Delta_l \leftarrow w_l G(q_l) - w_l \cdot g_l^*$
    if $\Delta_l \geq \Delta_{best}$ **then**
      $\Delta_{best} \leftarrow \Delta_l$; $best_l \leftarrow l$; $best_h \leftarrow h_l^*$
    end
  end
  for $x \in S$ **do**
    $p_{l_1}(x) \leftarrow p_l(x) \cdot \Pr(best_h(x))$
    $p_{l_0}(x) \leftarrow p_l(x) \cdot (1 - \Pr(best_h(x)))$
  end
  $w_{l_1} \leftarrow \sum_{x \in S} p_{l_1}(x)$; $w_{l_0} \leftarrow \sum_{x \in S} p_{l_0}(x)$
  $q_{l_1} \leftarrow \sum_{x \in S : f(x) = 1} p_{l_1}(x)$; $q_{l_0} \leftarrow \sum_{x \in S : f(x) = 1} p_{l_0}(x)$
  $T \leftarrow SplitLeaf(T, best_l, best_h)$
end
for *each* $l \in leaves(T)$*:* **do**
  $h_l \leftarrow FitH_L(S, \frac{1}{w_l} p_l(x))$
end
return T

Algorithm 1: Decision tree learning algorithm

Appendix B

Proof of Theorem 5.0.1

In this section we provide a sketch of the proof for Theorem 5.0.1 under the Stochastic Weak Learning Assumption. The proof is based on the proof for the deterministic case by [KM99], however it is slightly modified to account for the stochastic hypothesis we introduce in this work. We will only prove the place where the proofs differ, namely Lemma 2 of [KM99] is substitute by Lemma B.0.1.

We start with a few notations. Let f be the target function, let T be a decision tree, a leaf $l \in Leaves(T)$, and let $h \in H_I$. Denote the leaves introduced by $Split(T, l, h)$ by l_1 and l_0. Recall that $\texttt{purity}(l)$ is the fraction of positive examples that reach leaf l from all of the examples that reach it. We use the following shorthand notations for reoccurring expressions:

$$\begin{aligned} q =& \texttt{purity}(l), p = \texttt{purity}(l_0), r = \texttt{purity}(l_1) \\ \tau =& \Pr_{(x,y)\sim D}(Reach(x, l_1)|Reach(x, l)) \\ =& \Pr_{(x,y)\sim D}(h(x) = 1|Reach(x, l)) \\ =& \frac{\texttt{weight}(l_1)}{\texttt{weight}(l)} \end{aligned}$$

Since $q = (1 - \tau)p + \tau r$ and $\tau \in [0, 1]$ than without loss of generality we have: $p \leq q \leq r$. Let:

$$\delta = r - p \ .$$

Let D_l be the distribution over input at leaf l. We also define D'_l the balanced distribution over inputs in leaf l in which the probability positive (or negative) weights is $1/2$. Formally,

$$D_l(x) = \frac{Reach(x,l)}{\texttt{weight}(l)} D(x)$$

$$D'_l(x) = \begin{cases} \frac{D_l(x)}{2q} & \text{if } f(x) = 1 \\ \frac{D_l(x)}{2(1-q)} & \text{if } f(x) = 0 \end{cases}$$

Recall that the information gain is $\Delta = G(q) - (1-\tau)G(p) - \tau G(r)$. Note that if either δ is small or τ is near 0 or 1 then the information gain is small. Lemma B.0.1 shows that if $h \in H_I$ is used to split at l, and h satisfies the Weak Stochastic Hypothesis Assumption for D'_l, then both δ cannot be too small, and τ cannot be too close to either 0 or 1.

Lemma B.0.1. *Let p, τ and δ be as defined above for the split $h \in H_I$ at leaf l. Let D_l and D'_l also be defined as above for l. If the function h satisfies* $\Pr_{x \sim D'_l}(h(x) \neq f(x)) \leq 0.5 - \gamma$, *Then:*

$$\tau(1-\tau)\delta \geq 2\gamma q(1-q) \ .$$

Proof. We calculate the following expressions in terms of τ, r and q:

$$\Pr_{x \sim D_l}(f(x) = 1, h(x) = 1) = \tau r$$

Therefore,

$$\Pr_{x \sim D_l}(f(x) = 0, h(x) = 1) = \tau(1-r)$$

and

$$\Pr_{x \sim D_l}(f(x) = 1, h(x) = 0) = (1-\tau)p = q - \tau r$$

Next we consider the error terms under the distribution D'_l. When changing from the distribution D_l to D'_l we need to scale the expression according to the labels:

$$\Pr_{x \sim D'_l}(f(x) = 0, h(x) = 1) = \frac{\tau(1-r)}{2(1-q)}$$

and

$$\Pr_{x\sim D'_l}(f(x)=1,h(x)=0)=\frac{q-r\tau}{2q}$$

Finally the error terms are combined to achieve the total error:

$$\begin{aligned}\Pr_{x\sim D'_l}(f(x)\neq h(x)) &= \Pr_{x\sim D'_l}(f(x)=0,h(x)=1)+\Pr_{x\sim D'_l}(f(x)=1,h(x)=0)\\ &=\frac{\tau(1-r)}{2(1-q)}+\frac{q-r\tau}{2q}=\frac{1}{2}+\frac{\tau}{2}\left(\frac{1-r}{1-q}-\frac{r}{q}\right)=\frac{1}{2}+\frac{\tau}{2}\cdot\frac{q-r}{q(1-q)}\end{aligned}$$

By our assumption on h we have,

$$\Pr_{x\sim D'_l}(f(x)\neq h(x))=\frac{1}{2}+\frac{\tau}{2}\cdot\frac{q-r}{q(1-q)}\leq\frac{1}{2}-\gamma$$

which implies that

$$\frac{\tau}{2}\cdot\frac{r-q}{q(1-q)}\geq\gamma$$

Substituting $r=q+(1-\tau)\delta$ we get:

$$\frac{\tau}{2}\cdot\frac{(1-\tau)\delta}{q(1-q)}\geq\gamma\rightarrow\tau(1-\tau)\delta\geq2\gamma q(1-q)$$

as required. □

Appendix C

Bounding the WGI by regions

In this appendix, we show how to partition the domain of WGI into cells such that the difference between values of the WGI of points in the same cell is bounded by ϵ. Namely, we prove the following theorem.

Theorem 7.1.1. Given data distribution D_l which has a positive label probability of ρ and a parameter $\epsilon \in (0,1)$ it is possible to partition the domain of WGI: $\{(p,q)|0 \leq p \leq 1, \max(0, \rho+p-1) \leq q \leq \min(\rho,p)\}$ into $O\left(\epsilon^{-2}\right)$ cells, such that the if (p_1,q_1) and (p_2,q_2) belong to the same cell:

$$|WGI(p_1,q_1) - WGI(p_2,q_2)| \leq \epsilon \ .$$

C.1 Partitioning the WGI to monotone regions

Consider the domain of WGI which is

$$R = \{(p,q)|p \in (0,1), q \in [\max(0, p+\rho-1), \min(\rho,p)]\} \ .$$

Note that the values of $p=0$ and $p=1$ are not included. We discuss and add them in Section C.7. We partition R to two sub-domains using the line $q = \rho p$, namely,

$$R_a = \{(p,q)|p \in (0,1), q \in [\rho p, \min(\rho,p)]\}$$

and

$$R_b = \{(p,q)|p \in (0,1), q \in [\max(0, p+\rho-1), \rho p]\} \ ,$$

as illustrated in Figure C.1.

We show that the function WGI is monotone in each of those domains.

Lemma C.1.1. *Given data distribution D_l which has a positive class weight of ρ, the WGI function:*

$$WGI(p,q) = 4\left(\rho - \frac{q^2}{p} - \frac{(\rho-q)^2}{1-p}\right)$$

is monotone in both p and q in both R_a and R_b. Specifically, in R_a it is increasing in p and decreasing in q and in R_b it is decreasing in p and increasing in q.

Proof. Since we are interested only in monotonicity, we can consider simply the function $g(p,q) = -\frac{q^2}{p} - \frac{(\rho-q)^2}{1-p}$. In order to show that $g(p,q)$ is monotone, we first consider where its derivatives vanishes. The derivative with respect to q is

$$\frac{\partial g(p,q)}{\partial q} = -\frac{2q}{p} + \frac{2(\rho-q)}{(1-p)} = \frac{2(\rho p - q)}{p(1-p)}$$

This implies that the derivative vanishes at $q = \rho p$ which is the boundary of the sub-domains R_a and R_b, and hence $g(p,q)$ is monotone in q in each of the domains. Also, in R_a, since $q > p\rho$, the derivative is negative and in R_b, since $q \leq p\rho$, it is positive.

Next we consider the derivative with respect to p,

$$\frac{\partial g(p,q)}{\partial p} = \frac{q^2}{p^2} - \frac{(\rho-q)^2}{(1-p)^2} = \frac{(q-\rho p)(q(1-2p)+\rho p)}{p^2(1-p)^2}$$

This implies that the derivative vanishes at $q = \rho p$ and $q = \frac{\rho p}{2p-1}$. The line $q = \rho p$ is a boundary of our regions and so we consider only the other curve.

For $p \in (0, 0.5)$ we have $1 - 2p > 0$. This implies that $q(1-2p) + \rho p > 0$. Since $q > \rho p$ in R_a it implies that $g(p,q)$ is increasing there, and similarly, since $q \leq p\rho$, it is decreasing in R_b.

For $p \in (0.5, 1)$ we have $-1 < 1 - 2p < 0$. For R_b, since $q \leq \rho p$ we have that $q(1-2p) + \rho p > 0$ and hence $g(p,q)$ is decreasing in p. For R_a, we have

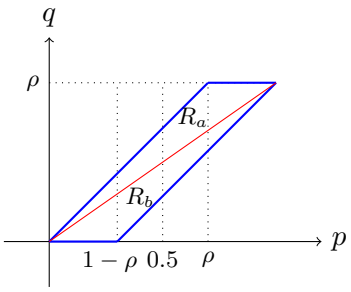

Figure C.1: The domain R and the sub-domains R_a and R_b (for $\rho \geq 0.5$)

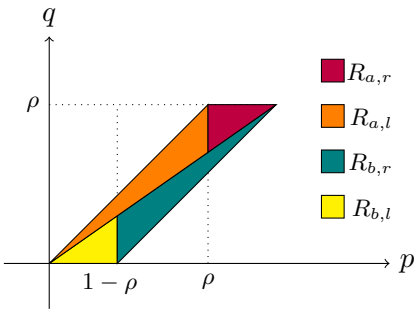

Figure C.2: WGI: partitioned into regions for $\rho \geq 0.5$

$q \leq p + \rho - 1$. This implies that $q(1-2p) + \rho p \leq (1-p)(2p-1+\rho)$ which is non-negative since $p \in (0.5, 1)$. Therefore $q(p,q)$ is increasing in R_a.

Finally, for $p = 0.5$ we have that $\frac{\partial g(p,q)}{\partial p} = \frac{(q-\rho/2)(\rho/2)}{1/16}$ which is positive in R_a and negative in R_b as required. □

Figure C.1 show pictorially the regions R_a and R_b.

C.2 Overview of the partition to cells

We partitioned the domain R of WGI to two sub-domains R_a and R_b. According to Theorem C.1.1 in each of the sub-domains the WGI is monotone in p and q. We will partition each sub-domain in two thus creating four regions with linear boundaries which are monotone in p and q. Finally we partition each of the four regions further into cells in order to bound the WGI difference as required.

We first consider the difference in the value of WGI between two points:

$$\begin{aligned}|WGI(p_1,q_1) - WGI(p_2,q_2)| &= 4\left|\frac{q_2^2}{p_2} + \frac{(\rho-q_2)^2}{1-p_2} - \frac{q_1^2}{p_1} - \frac{(\rho-q_1)^2}{1-p_1}\right| \\ &\leq 4\left|\frac{q_2^2}{p_2} - \frac{q_1^2}{p_1}\right| + 4\left|\frac{(\rho-q_2)^2}{1-p_2} - \frac{(\rho-q_1)^2}{1-p_1}\right| \\ &= 4\Delta_1 + 4\Delta_2 \qquad \text{(C.1)}\end{aligned}$$

where $\Delta_1 = \left|\frac{q_2^2}{p_2} - \frac{q_1^2}{p_1}\right|$ and $\Delta_2 = \left|\frac{(\rho-q_2)^2}{1-p_2} - \frac{(\rho-q_1)^2}{1-p_1}\right|$.

We are going to consider the case where $\rho \geq 0.5$. The symmetric case of $\rho < 0.5$ can be reduced to this case by switching the labels and deriving the same bound. For the definition of the cells we use the parameters $\epsilon \in (0,1)$ which controls the cell's size.

We next present an analysis for each of four sub-domains, and show that in each both $\Delta_1 = O(\epsilon)$ and $\Delta_2 = O(\epsilon)$. In addition each sub-domain will be partitioned in to $O(\epsilon^{-2})$ cells.

C.3 First region: $R_{b,l}$

Let $R_{b,l} = \{(p,q)|0 < p \leq 1-\rho, 0 \leq q \leq \rho p\}$. Clearly $R_{b,l} \subseteq R_b$, and by Lemma C.1.1 we know that WGI is decreasing in p and increasing in q in $R_{b,l}$.

We are now ready to define the cells. The cells will have two parameters $p_c \in (0, 1-\rho-\epsilon]$ and $\alpha \in [\epsilon, 1]$. Let

$$cell_{R_{b,l}}(p_c,\alpha) = \{(p,q)|p_c \leq p \leq p_c+\epsilon, (\alpha-\epsilon)\rho p \leq q \leq \alpha\rho p\}\,.$$

The cells would have $p_c = i\epsilon$, for $0 \leq i \leq \frac{1-\rho}{\epsilon} - 1$ and $\alpha = j\epsilon$, for $1 \leq j \leq 1/\epsilon$. This implies that there are $O(\epsilon^{-2})$ cells in $R_{b,l}$. Since in $R_{b,l}$ we have $\frac{\partial WGI}{\partial q} \geq 0$, there are $p_1, p_2 \in (0, 1-\rho]$, such that the maximum WGI value in $cell_{R_{b,l}}(p_c,\alpha)$ is obtained at a point $(p_1,q_1) = (p_1, \alpha\rho p_1)$ and the minimum value at $(p_2,q_2) = (p_2, (\alpha-\epsilon)\rho p_2)$.

We bound Δ_1 as follows:

$$\begin{aligned}\Delta_1 = &\left|\frac{q_2^2}{p_2} - \frac{q_1^2}{p_1}\right| = \left|\frac{(\alpha-\epsilon)^2\rho^2 p_2^2}{p_2} - \frac{\alpha^2\rho^2 p_1^2}{p_1}\right| \\ \leq &\rho^2(\alpha^2 \overbrace{|p_2 - p_1|}^{\leq \epsilon} + \epsilon p_2 \overbrace{|\epsilon - 1|}^{\leq 1}) \leq \rho^2(\alpha^2\epsilon + \epsilon p_2) \leq 2\epsilon \end{aligned} \tag{C.2}$$

and Δ_2 as follows,

$$\begin{aligned}\Delta_2 = &\left|\frac{(\rho-q_2)^2}{1-p_2} - \frac{(\rho-q_1)^2}{1-p_1}\right| = \left|\frac{(\rho-(\alpha-\epsilon)\rho p_2)^2}{1-p_2} - \frac{(\rho-\alpha\rho p_1)^2}{1-p_1}\right| \\ = &\rho^2\left|\frac{(1-(\alpha-\epsilon)p_2)^2}{1-p_2} - \frac{(1-\alpha p_1)^2}{1-p_1}\right| \\ = &\rho^2\left|\frac{((1-p_2)+((1-\alpha)+\epsilon)p_2)^2}{1-p_2} - \frac{((1-p_1)+(1-\alpha)p_1)^2}{1-p_1}\right| \\ \leq &\rho^2\left(|p_2-p_1| + 2(1-\alpha)|p_2-p_1| + 2p_2\epsilon + (1-\alpha)^2\left|\frac{p_2^2}{1-p_2} - \frac{p_1^2}{1-p_1}\right| + \frac{p_2^2\epsilon(\overbrace{1-\alpha+\epsilon}^{\leq 2})}{1-p_2}\right) \\ \leq &5\epsilon + 2\epsilon\frac{p_2}{1-p_2} + \rho^2\left|\frac{p_2^2 - p_1^2 + p_1^2p_2 - p_1p_2^2}{(1-p_1)(1-p_2)}\right| \\ \leq &7\epsilon + \rho^2\frac{\overbrace{(p_1+p_2)}^{\leq 2}|p_2-p_1|}{(1-p_1)(1-p_2)} + \rho^2\frac{\overbrace{p_1p_2}^{\leq 1}|p_2-p_1|}{(1-p_1)(1-p_2)} \\ \leq &7\epsilon + 3\epsilon\frac{\rho^2}{(1-p_1)(1-p_2)} \\ = &10\epsilon \end{aligned} \tag{C.3}$$

where we used the fact that $\rho \leq 1$ and $p \leq 1-\rho \leq \rho \leq 1-p$ (or $\frac{p}{1-p} \leq 1$ or $\frac{\rho}{1-p} \leq 1$).

Combining the bounds in (C.1), (C.2) and (C.3) the error for in each cell in $R_{b,l}$ is bounded by $4(2\epsilon + 10\epsilon) = 48\epsilon$.

C.4 Second region: $R_{b,r}$

Let $R_{b,r} = \{(p,q)|1-\rho \leq p < 1, p+\rho-1 \leq q \leq \rho p\}$. Clearly $R_{b,r} \subseteq R_b$, and by Lemma C.1.1 we know that WGI is decreasing in p and increasing in q in $R_{b,r}$.

We are now ready to define the cells. The cells will have two parameters $p_c \in [1-\rho, 1-\epsilon)$ and $\alpha \in [\epsilon, 1]$. Let

$$cell_{R_{b,r}}(p_c,\alpha) = \{(p,q) : p_c \le p \le p_c+\epsilon, \\ (\alpha-\epsilon)\rho p + (1-\alpha+\epsilon)(p+\rho-1) \le q \le \alpha\rho p + (1-\alpha)(p+\rho-1)\}\ .$$

The cells would have $p_c = i\epsilon$, for $(1-\rho)/\epsilon \le i \le 1/\epsilon - 1$ and $\alpha = j\epsilon$, for $1 \le j \le 1/\epsilon$. This implies that there are $O(\epsilon^{-2})$ cells in $R_{b,r}$. Since in $R_{b,r}$ we have $\frac{\partial WGI}{\partial q} \ge 0$, there are $p_1, p_2 \in [1-\rho, 1)$, such that the maximum WGI value in $cell_{R_{b,r}}(p_c,\alpha)$ is obtained at a point $(p_1, q_1) = (p_1, \alpha\rho p_1 + (1-\alpha)(p_1+\rho-1))$ and the minimum value at $(p_2, q_2) = (p_2, (\alpha-\epsilon)\rho p_2 + (1-\alpha+\epsilon)(p_2+\rho-1))$

We bound Δ_1 as follows:

$$\begin{aligned}
\Delta_1 =& \left|\frac{q_2^2}{p_2} - \frac{q_1^2}{p_1}\right| \\
=& \left|\frac{((\alpha-\epsilon)\rho p_2 + (1-\alpha+\epsilon)(p_2+\rho-1))^2}{p_2} - \frac{(\alpha\rho p_1 + (1-\alpha)(p_1+\rho-1))^2}{p_1}\right| \\
\le& \alpha^2\rho^2\,|p_2-p_1| + \epsilon\overbrace{|\epsilon-2\alpha|}^{\le 2}\rho^2 p_2 + 2\overbrace{\alpha(1-\alpha)}^{\le 0.25}\rho\,|p_2-p_1| \\
&+ 2\epsilon\overbrace{|2\alpha-\epsilon-1|}^{\le 1}\rho\overbrace{|p_2+\rho-1|}^{\le 1} + \\
&(1-\alpha)^2\left|\frac{(p_2+\rho-1)^2}{p_2} - \frac{(p_1+\rho-1)^2}{p_1}\right| + \frac{\epsilon\overbrace{|2-2\alpha+\epsilon|}^{\le 2}(p_2+\rho-1)^2}{p_2} \\
\le& 5.5\epsilon + \overbrace{\left|\frac{(p_2+\rho-1)^2}{p_2} - \frac{(p_1+\rho-1)^2}{p_1}\right|}^{C.5} + \overbrace{\frac{2\epsilon(p_2+\rho-1)^2}{p_2}}^{C.6} \le 12.5\epsilon
\end{aligned} \tag{C.4}$$

Where we used the following expressions:

$$\left|\frac{(p_2+\rho-1)^2}{p_2} - \frac{(p_1+\rho-1)^2}{p_1}\right| \le |p_2-p_1| + (\rho-1)^2\frac{|p_1-p_2|}{p_1p_2} \le 2\epsilon \tag{C.5}$$

$$\begin{aligned}\frac{2\epsilon(p_2+\rho-1)^2}{p_2} \leq & 2\epsilon\left(p_2+2\overbrace{|\rho-1|}^{\leq 0.5}+\frac{(\rho-1)^2}{p_2}\right)\\ \leq & 2\epsilon\left(2+\overbrace{(1-\rho)}^{\leq 0.5}\frac{1-\rho}{p_2}\right)\\ \leq & 5\epsilon \end{aligned} \tag{C.6}$$

where we used the fact that $p \geq 1-\rho$ (or $\frac{1-\rho}{p} \leq 1$).

For Δ_2 we have

$$\begin{aligned}\Delta_2 = & \left|\frac{(\rho-q_2)^2}{1-p_2}-\frac{(\rho-q_1)^2}{1-p_1}\right|\\ = & \left|\frac{(\rho-(\alpha-\epsilon)\rho p_2-(1-\alpha+\epsilon)(p_2+\rho-1))^2}{1-p_2}-\frac{(\rho-\alpha\rho p_1-(1-\alpha)(p_1+\rho-1))^2}{1-p_1}\right|\\ \leq & |p_1-p_2|\left(\overbrace{\rho^2\alpha^2}^{\leq 1}+2\rho\overbrace{\alpha(1-\alpha)}^{\leq 0.25}+\overbrace{(1-\alpha)^2}^{\leq 1}\right)\\ & +\overbrace{(1-p_2)}^{\leq 1}\left(\rho^2\epsilon\overbrace{|\epsilon-2\alpha|}^{\leq 2}+2\rho\epsilon\overbrace{|2\alpha-1-\epsilon|}^{\leq 1}+\epsilon\overbrace{|2-2\alpha+\epsilon|}^{\leq 2}\right)\\ \leq & 8.5\epsilon \end{aligned} \tag{C.7}$$

Combining the bounds in (C.1), (C.4) and (C.7) the error for in each cell in $R_{b,l}$ is bounded by $4(12.5\epsilon+8.5\epsilon)=84\epsilon$.

C.5 Third region $R_{a,l}$

Let $R_{a,l}=\{(p,q)|0<p\leq\rho,\rho p\leq q\leq p\}$. Clearly $R_{a,l}\subseteq R_a$, and by Lemma C.1.1 we know that WGI is increasing in p and decreasing in q in $R_{a,l}$.

We are now ready to define the cells. The cells will have two parameters $p_c \in (0,\rho-\epsilon]$ and $\alpha\in[\epsilon,1]$. Let

$$cell_{R_{a,l}}(p_c,\alpha)=\{(p,q)|p_c\leq p\leq p_c+\epsilon,(\alpha-\epsilon)p+(1-\alpha+\epsilon)\rho p\leq q\leq \alpha p+(1-\alpha)\rho p\}\ .$$

The cells would have $p_c=i\epsilon$, for $0\leq i\leq \rho/\epsilon-1$ and $\alpha=j\epsilon$, for $1\leq j\leq 1/\epsilon$. This implies that there are $O(\epsilon^{-2})$ cells in $R_{a,l}$. Since in $R_{a,l}$ we have $\frac{\partial WGI}{\partial q}\leq 0$,

there are $p_1, p_2 \in (0, \rho]$, such that the maximum WGI value in $cell_{R_{a,l}}(p_c, \alpha)$ is obtained at a point $(p_1, q_1) = (p_1, (\alpha - \epsilon)p_1 + (1 - \alpha + \epsilon)\rho p_1)$ and the minimum value at $(p_2, q_2) = (p_2, \alpha p_2 + (1 - \alpha)\rho p_2)$

We bound Δ_1 as follows,

$$\begin{aligned}\Delta_1 =& \left|\frac{q_2^2}{p_2} - \frac{q_1^2}{p_1}\right| = \left|\frac{(\alpha p_2 + (1-\alpha)\rho p_2)^2}{p_2} - \frac{((\alpha-\epsilon)p_1 + (1-\alpha+\epsilon)\rho p_1)^2}{p_1}\right| \\ \leq& |p_2 - p_1| \left|\rho + (1-\rho)\alpha\right|^2 + 2\epsilon p_1 \left|\rho + (1-\rho)\alpha\right| |\rho - 1| + \epsilon^2 p_1 (\rho - 1)^2 \\ \leq& 4\epsilon \end{aligned} \tag{C.8}$$

Where we used:

$$|\rho + \alpha(1-\rho)| \leq |\rho| + \alpha |1-\rho| \overset{\alpha \leq 1}{\leq} |\rho| + |1-\rho| \overset{0 \leq \rho \leq 1}{=} \rho + 1 - \rho = 1$$

and bound Δ_2 as follows,

$$\begin{aligned}\Delta_2 =& \left|\frac{(\rho - q_2)^2}{1-p_2} - \frac{(\rho - q_1)^2}{1-p_1}\right| \\ =& \left|\frac{(\rho - \alpha p_2 - (1-\alpha)\rho p_2)^2}{1-p_2} - \frac{(\rho - (\alpha-\epsilon)p_1 - (1-(\alpha-\epsilon))\rho p_1)^2}{1-p_1}\right| \\ \leq& \rho^2 |p_1 - p_2| + 2\alpha |\rho(\rho-1)| |p_2 - p_1| + \alpha^2 (1-\rho)^2 \left|\frac{p_2^2}{1-p_2} - \frac{p_1^2}{1-p_1}\right| + \\ & \frac{2\epsilon p_1 (1-\rho) |\rho - \alpha p_1 - (1-\alpha)\rho p_1|}{1-p_1} + \frac{\epsilon^2 p_1^2 (1-\rho)^2}{1-p_1} \\ \leq& 8.5\epsilon \end{aligned} \tag{C.9}$$

Combining the bounds in (C.1), (C.8) and (C.9) the error for in each cell in $R_{a,l}$ is bounded by $4(4\epsilon + 8.5\epsilon) = 50\epsilon$.

C.6 Fourth region $R_{a,r}$

Let $R_{a,r} = \{(p, q) | \rho \leq p < 1, \rho p \leq q \leq \rho\}$. Clearly $R_{a,r} \subseteq R_a$, and by Lemma C.1.1 we know that WGI is increasing in p and decreasing in q in $R_{a,r}$.

We are now ready to define the cells. The cells will have two parameters $p_c \in [1 - \rho, 1 - \epsilon)$ and $\alpha \in [\epsilon, 1]$. Let

$$cell_{R_{a,r}}(p_c, \alpha) = \{(p, q) | p_c \leq p \leq p_c + \epsilon, (\alpha - \epsilon)\rho + (1 - \alpha + \epsilon)\rho p \leq q \leq \alpha\rho + (1-\alpha)\rho p\} \ .$$

The cells would have $p_c = i\epsilon$, for $\rho/\epsilon \leq i \leq 1/\epsilon - 1$ and $\alpha = j\epsilon$, for $1 \leq j \leq 1/\epsilon$. This implies that there are $O(\epsilon^{-2})$ cells in $R_{a,r}$. Since in $R_{a,r}$ we have $\frac{\partial WGI}{\partial q} \leq 0$, there are $p_1, p_2 \in [1-\rho, 1)$, such that the maximum WGI value in $cell_{R_{a,r}}(p_c, \alpha)$ is obtained at a point $(p_1, q_1) = (p_1, (\alpha-\epsilon)\rho + (1-\alpha+\epsilon)\rho p_1)$ and the minimum value at $(p_2, q_2) = (p_2, \alpha\rho + (1-\alpha)\rho p_2)$

We bound Δ_1 as follows,

$$\begin{aligned}
\Delta_1 =& \left|\frac{q_2^2}{p_2} - \frac{q_1^2}{p_1}\right| = \left|\frac{(\alpha\rho + (1-\alpha)\rho p_2)^2}{p_2} - \frac{((\alpha-\epsilon)\rho + (1-\alpha+\epsilon)\rho p_1)^2}{p_1}\right| \\
\leq& \rho^2 \left(|p_2 - p_1| + 2\alpha\,|p_1 - p_2| + 2\epsilon\,|1 - p_1|\right) \\
&+ \rho^2 \left|\frac{\alpha^2(1-p_2)^2}{p_2} - \frac{\alpha^2(1-p_1)^2}{p_1} - \frac{\epsilon(\epsilon-2\alpha)(1-p_1)^2}{p_1}\right| \\
\leq& 8\epsilon
\end{aligned} \tag{C.10}$$

and Δ_2 as follows,

$$\begin{aligned}
\Delta_2 =& \left|\frac{(\rho - q_2)^2}{1-p_2} - \frac{(\rho - q_1)^2}{1-p_1}\right| \\
=& \left|\frac{(\rho - \alpha\rho - (1-\alpha)\rho p_2)^2}{1-p_2} - \frac{(\rho - (\alpha-\epsilon)\rho - (1-(\alpha-\epsilon))\rho p_1)^2}{1-p_1}\right| \\
\leq& \rho^2(1-\alpha)^2\,|p_1 - p_2| + \rho^2\epsilon\,|2 - 2\alpha + \epsilon|\,(1-p_1) \\
\leq& 3\epsilon
\end{aligned} \tag{C.11}$$

Combining the bounds in (C.1), (C.10) and (C.11) the error for in each cell in $R_{a,r}$ is bounded by $4(8\epsilon + 3\epsilon) = 44\epsilon$.

C.7 WGI is continuous in $p = 0$ and $p = 1$

We show that the value of WGI is continuous at both extreme points $p = 0$ and $p = 1$ for point $(p, q) \in R$. Namely, we show that its value is $4\rho(1-\rho)$. This will allow us to extend the sub-domains from $p \in (0, 1)$ to $p \in [0, 1]$. Recall that,

$$WGI(p, q) = 4\left(\rho - \frac{q^2}{p} - \frac{(\rho - q)^2}{1-p}\right)$$

Lemma C.7.1. *$WGI(p, q)$ (7.1) is continuous in $p = 0$.*

Proof. By definition of WGI 7.1, we know that $WGI(0, q) = 4\rho(1-\rho)$. We show that this is also the limit when p approaches 0.

$$\lim_{p\to 0+} 4\left(\rho - \frac{q^2}{p} - \frac{(\rho-q)^2}{1-p}\right) = 4\left(\rho - \lim_{p\to 0+}\frac{q^2}{p} - \lim_{p\to 0+}\frac{(\rho-q)^2}{1-p}\right)$$

First, since p and q are non negative and $q \leq p$ we have $0 \leq \frac{q^2}{p} \leq q$. Also if p is going to 0, so is q therefore $\lim_{p\to 0} q = 0$. According to the squeeze theorem, we get $\lim_{p\to 0+}\frac{q^2}{p} = 0$. Second, we expand the term $\frac{(\rho-q)^2}{1-p} = \frac{\rho^2}{1-p} - \frac{2\rho q}{1-p} + \frac{q^2}{1-p}$ and calculate the limit for every sub-term separately (ρ^2,0 and 0 correspondingly). Therefore:

$$\lim_{p\to 0+}\frac{\rho^2}{1-p} - \frac{2\rho q}{1-p} + \frac{q^2}{1-p} = \rho^2$$

Finally the limit at $p \to 0+$ is $4\rho(1-\rho)$. □

Lemma C.7.2. *$WGI(p, q)$ (7.1) is continuous in $p = 1$.*

Proof. By definition of WGI 7.1, we know that $WGI(1, q) = 4\rho(1-\rho)$. We show that this is also the limit when p approaches 1.

$$\lim_{p\to 1-} 4\left(\rho - \frac{q^2}{p} - \frac{(\rho-q)^2}{1-p}\right) = 4\left(\rho - \lim_{p\to 1-}\frac{q^2}{p} - \lim_{p\to 1-}\frac{(\rho-q)^2}{1-p}\right)$$

First we notice that $\lim_{p\to 1-} p + \rho - 1 = \rho$ and $\lim_{p\to 1-} min(p, \rho) = \rho$. According to the squeeze theorem and $p + \rho - 1 \leq q \leq min(p, \rho)$ we get $\lim_{p\to 1-} q = \rho$.

Next, since all terms are positive $\frac{(\rho-q)^2}{1-p} \geq 0$. Since $p + \rho - 1 \leq q \to \rho - q \leq 1 - p$ and thus $\frac{(\rho-q)^2}{1-p} \leq \rho - q$. Since $0 \leq \frac{(\rho-q)^2}{1-p}$ and $\lim_{p\to 1-} \rho - q = \rho - \rho = 0$ we apply the squeeze theorem and get that: $\lim_{p\to 1-}\frac{(\rho-q)^2}{1-p} = 0$. Finally, $4\left(\rho - \lim_{p\to 1-}\frac{q^2}{p} - \lim_{p\to 1-}\frac{(\rho-q)^2}{1-p}\right) = 4(\rho - \rho^2)$ as required. □

C.8 Proof conclusion

We finish proving the following Theorem:

Theorem 7.1.1. Given data distribution D_l which has a positive label probability of ρ and a parameter $\epsilon \in (0,1)$ it is possible to partition the domain of WGI: $\{(p,q)|0 \leq p \leq 1, \max(0, \rho+p-1) \leq q \leq \min(\rho,p)\}$ into $O\left(\epsilon^{-2}\right)$ cells, such that the if (p_1,q_1) and (p_2,q_2) belong to the same cell:

$$|WGI(p_1,q_1) - WGI(p_2,q_2)| \leq \epsilon \ .$$

In section C.1, we described 4 mutually exclusive regions that cover the domain of WGI. Let $\bar{\epsilon} \in (0,1)$, then for each region we showed in sections C.3,C.4,C.5 and C.6 a partition of each region to $O(\bar{\epsilon}^{-2})$ mutually exclusive cells that cover the domain of the region, such that if (p_1,q_1) and (p_2,q_2) are within the cell then $\|WGI(p_1,q_1) - WGI(p_2,q_2)\| \leq 100\bar{\epsilon}$.

If combined, these partitions partition the domain of WGI into $4 \cdot O(\bar{\epsilon}^{-2}) = O(\bar{\epsilon}^{-2})$ cells. Next, according to the continuity of WGI for $p=0$ and $p=1$, as showed in section C.7, the partition can extended from $p \in (0,1)$ to $p=[0,1]$ and cover the full domain of WGI. Finally, we set $\bar{\epsilon} = 0.01\epsilon$, clearly we $O(\epsilon^{-2}) = O(\bar{\epsilon}^{-2})$ which concludes the proof.

C.9 Solution for the dependent case

In this section we are going to provide a minimal solution for the WGI in the dependent case.

Lemma C.9.1. *Let $\vec{a} = D_v^\top X$ and $\vec{b} = (D_v \odot y)^\top X$, and assume $\vec{b} = \lambda\vec{a}$. The solutions for $\min(WGI_w)$ are $w_{1,2} = \pm\frac{\vec{a}}{\|\vec{a}\|}$*

Proof. Recall that according to 7.1:

$$WGI_w = WGI(P_w, Q_w) = 4\left(\rho - \frac{Q_w^2}{P_w} - \frac{(\rho - Q_w)^2}{1-P_w}\right)$$

and that $P_w = \frac{\vec{a}^\top w+1}{2}$ and $Q_w = \frac{\vec{b}^\top w+\rho}{2}$. Denote $x = \vec{a}^\top w$ and therefore $\vec{b}^\top w = \lambda \vec{a}^\top w = \lambda x$ we find the solution for:

$$\begin{aligned} g(x) =& 4\rho - \frac{4\left(\frac{\lambda x+\rho}{2}\right)^2}{\frac{x+1}{2}} - \frac{4\left(\rho - \frac{\lambda x+\rho}{2}\right)^2}{1-\frac{x+1}{2}} = 4\rho - \frac{2\left(\rho+\lambda x\right)^2}{1+x} - \frac{2\left(\rho-\lambda x\right)^2}{1-x} \\ =& 4\rho - \frac{4\left(\rho^2 + \lambda(\lambda-2\rho)x^2\right)}{1-x^2} \end{aligned}$$

We substitute $z = x^2$ and derive:

$$\frac{\partial g}{\partial z} = -4 \cdot \frac{\lambda(\lambda-2\rho)(1-z) + (\rho^2+\lambda(\lambda-2\rho)z)}{(1-z)^2} = -4 \cdot \frac{(\lambda-\rho)^2}{(1-z)^2}$$

by the chain rule:

$$\frac{\partial g}{\partial x} = \frac{\partial g}{\partial z} \cdot \frac{\partial z}{\partial x} = \frac{-8x(\lambda-\rho)^2}{\left(1-x^2\right)^2}$$

First we note that $x = 0$ is an extreme point. Next, since $\frac{(\lambda-\rho)^2}{(1-x^2)^2} > 0$, we can see that if $x > 0$ we see that g is decreasing, and if $x < 0$ we see that g is increasing. Therefore, the minimal points are at the edges of the feasible interval of x.

Since $x = \vec{a}^\top w$ and $\|w\| \leq 1$, according to $\cdot$ operator we have: $-\|\vec{a}\| \leq x \leq \|\vec{a}\|$. If $w = \frac{\vec{a}}{\|\vec{a}\|}$ then $x = \|a\|$, and if $w = -\frac{\vec{a}}{\|\vec{a}\|}$ then $x = -\|a\|$. Therefore, $w_{1,2} = \pm\frac{\vec{a}}{\|\vec{a}\|}$ are the solutions for this case. Clearly, both solutions attain the same value of WGI. □

Appendix D

Maximizing Information Gain for P_w and Q_w

In this appendix we characterize various optimization problems that search for a feasible stochastic linear classifiers. We have a few optimization problems depending on the information we assume about P_w and Q_w. The solutions of these of optimizations are used in our algorithms. We will assume that the vectors $\vec{a}$ and $\vec{b}$ are linearly independent throughout the appendix (since the dependent case can be directly found without using an optimization as was shown in Appendix C.9).

We first discuss the basic problem where the value of both P_w and Q_w are given (Appendix D.1). In Appendix D.2 we then show, that the optimal solution within a cell (as define in Appendix C) is found in one of the corners of that cell. The solution is described in Appendix D.1 and D.2 is used in algorithm `FixedPQ`. Next, in Appendix D.3 we solve the optimization used in `FindPSearchQ` for finding a feasible stochastic linear classifier where only the value of P_w is given.

Finally, in Appendix D.4, we show that the set of feasible stochastic linear classifiers restricts the domain of WGI where solutions can be found.

D.1 Values of P_w and Q_w are given

In this section we assume that the values of both P_w and Q_w are given, and we would like to compute whether there exists a feasible w with that give rise to those values, namely, $\|w\| \leq 1$. Such a w implies a feasible stochastic linear classifier. Let the values be: $P_w = p$ and $Q_w = q$. The following lemma characterizes the solution.

Lemma D.1.1. *Let $\vec{a} = D_v^\top X$ and $\vec{b} = (D_v \odot y)^\top X$, and assume that they are linearly independent. For the quadratic optimization problem:*

$$\begin{aligned} w^* = \arg\min \|w\|_2^2 \\ P_w = p \\ Q_w = q \end{aligned}$$

We have that,

$$w^* = \frac{(2q-\rho)(\vec{a}\cdot\vec{b}) - (2p-1)\|\vec{b}\|^2}{\|\vec{a}\|^2\|\vec{b}\|^2 - (\vec{a}\cdot\vec{b})^2}\vec{a} + \frac{(2p-1)(\vec{a}\cdot\vec{b}) - (2q-\rho)\|\vec{a}\|^2}{\|\vec{a}\|^2\|\vec{b}\|^2 - (\vec{a}\cdot\vec{b})^2}\vec{b}$$

We will use Lemma D.1.1 in the following way. If for a given (p, q) we have that resulting w^* has $\|w^*\| \leq 1$, then we found a feasible stochastic linear classifier. Otherwise, we can conclude that there is no feasible stochastic linear classifier for (p, q), since w^* minimizes the norm.

Proof. Recall that

$$P_w = 0.5 \cdot D_v^\top X w + 0.5 = p \quad \text{and} \quad Q_w = 0.5 \cdot (D_v \odot y)^\top X w + 0.5\rho = q$$

We define $\vec{a}^\top w = 2p - 1 = \bar{p}$ and $\vec{b}^\top w = 2q - \rho = \bar{q}$. With this notation we have

$$\begin{aligned} \min 0.5 \sum_{i=1}^{d} w_i^2 \\ \vec{a}^\top \cdot w - \bar{p} = 0 \\ \vec{b}^\top \cdot w - \bar{q} = 0 \end{aligned}$$

We solve the optimization using the Lagrange multipliers:

$$L(w, \lambda_1, \lambda_2) = 0.5 \sum_{i=1}^{d} w_i^2 + \lambda_1(\vec{a}^\top w - \bar{p}) + \lambda_2(\vec{b}^\top w - \bar{q})$$

Considering the derivatives of the Lagrangian, we have

$$\frac{\partial L}{\partial w_i} = w_i + \lambda_1 \vec{a}_i + \lambda_2 \vec{b}_i = 0 \tag{D.1}$$

which implies that $\vec{w} = -\lambda_1 \vec{a} - \lambda_2 \vec{b}$. Also,

$$\frac{\partial L}{\partial \lambda_1} = \vec{a}^\top w - \bar{p} = 0 \tag{D.2}$$

Using that $\vec{w} = -\lambda_1 \vec{a} - \lambda_2 \vec{b}$ we have that $-\lambda_1 \|\vec{a}\|^2 - \lambda_2(\vec{a} \cdot \vec{b}) - \bar{p} = 0$ which implies that $\lambda_1 = -\frac{\lambda_2(\vec{a} \cdot \vec{b}) + \bar{p}}{\|\vec{a}\|^2}$.[1] We consider the other derivative,

$$\frac{\partial L}{\partial \lambda_2} = \vec{b}^\top w - \bar{q} = 0 \tag{D.3}$$

Similarly, this implies that $-\lambda_1(\vec{a} \cdot \vec{b}) - \lambda_2 \|\vec{b}\|^2 - \bar{q} = 0$. Solving for λ_1 and λ_2 and w^* we get,

$$\begin{aligned}
\lambda_2 =& \frac{\bar{p}(\vec{a} \cdot \vec{b}) - \bar{q}\|\vec{a}\|^2}{\|\vec{a}\|^2 \|\vec{b}\|^2 - (\vec{a} \cdot \vec{b})^2} \\
\lambda_1 =& \frac{\bar{q}(\vec{a} \cdot \vec{b}) - \bar{p}\|\vec{b}\|^2}{\|\vec{a}\|^2 \|\vec{b}\|^2 - (\vec{a} \cdot \vec{b})^2} \\
w^* =& \frac{\bar{q}(\vec{a} \cdot \vec{b}) - \bar{p}\|\vec{b}\|^2}{\|\vec{a}\|^2 \|\vec{b}\|^2 - (\vec{a} \cdot \vec{b})^2} \vec{a} + \frac{\bar{p}(\vec{a} \cdot \vec{b}) - \bar{q}\|\vec{a}\|^2}{\|\vec{a}\|^2 \|\vec{b}\|^2 - (\vec{a} \cdot \vec{b})^2} \vec{b}
\end{aligned} \tag{D.4}$$

Notice that this solution is valid only when the denominator is not 0. Therefore, it is valid if the $\vec{a}$ and $\vec{b}$ vectors are not in the same direction, i.e., $\forall \lambda : \vec{a} \neq \lambda \vec{b}$. □

D.2 Values of P_w and Q_w are within specific ranges

In this section we consider finding a feasible stochastic linear classifier inside a certain cell of values of P_w and Q_w (see Appendix C). The cell has the parameters

[1] We note that $\|\vec{a}\| > 0$ since we assume a coordinate with a fixed positive value for every sample, to allow for a bias coordinate.

p_c and α, in addition to $\epsilon \in (0,1)$. Let $U(P_w, \rho)$ and $L(P_w, \rho)$ be linear functions which bound the values of Q_w in the cell from above and below, respectively. The exact linear function depend on the sub-domain we are considering, namely the upper and lower boundaries of the region. Formally we solve the following quadratic optimization problem:

$$\begin{aligned}
&\min \|w\|_2^2 \\
&\quad p \le P_w \le p + \epsilon \\
&\quad (\alpha - \epsilon)U(P_w, \rho) + (1 - \alpha + \epsilon)L(P_w, \rho) \le Q_w \le \alpha U(P_w, \rho) + (1 - \alpha)L(P_w, \rho)
\end{aligned}$$

Since we minimize the norm of w, if there exists a feasible stochastic linear classifier in the cell, the result of this optimization would be feasible. If there is no feasible stochastic linear classifier in this cell, the result of this optimization would be a non feasible w (either $\|w\| > 1$ or no solution at all).

Lemma D.2.1. *Let $\vec{a} = D_v^\top X$ and $\vec{b} = (D_v \odot y)^\top X$, and assume they are linearly independent. Let $\epsilon \in (0,1)$, $\alpha \in [\epsilon, 1]$ and $p \in [0, 1-\epsilon]$. Let $U(P_w, \rho)$ and $L(P_w, \rho)$ be linear functions, such that:*

$$\forall P_w \in [p, p+\epsilon] : \begin{cases} L(P_w, \rho) < U(P_w, \rho) & p \in (0,1) \\ L(P_w, \rho) = U(P_w, \rho) & p = 0 \text{ or } p = 1 \end{cases}$$

Then the solution for:

$$\begin{aligned}
&\min \|w\|_2^2 \\
&\quad p \le P_w \le p + \epsilon \\
&\quad (\alpha - \epsilon)U(P_w, \rho) + (1 - \alpha + \epsilon)L(P_w, \rho) \le Q_w \le \alpha U(P_w, \rho) + (1 - \alpha)L(P_w, \rho)
\end{aligned}$$

coincides with a solution of

$$\begin{aligned}
&\min \|w\|_2^2 \\
&\quad P_w = p' \\
&\quad Q_w = q'
\end{aligned}$$

for $p' \in \{p, p+\epsilon\}$ *and* $q' \in \{(\alpha-\epsilon)U(P_w,\rho)+(1-\alpha+\epsilon)L(P_w,\rho), \alpha U(P_w,\rho)+(1-\alpha)L(P_w,\rho)\}$

We note that in order to compute the actual value, one could use the solution of Lemma D.1.1 in order to evaluate each of the four combinations for (p', q'). The above lemma guarantees that optimal solution is indeed one of them.

Proof. The above optimization is equivalent to the following optimization problem:

$$\begin{aligned}
&\min 0.5 \sum_{i=1}^{d} w_i^2 \\
&\overbrace{p - P_w}^{h_I} \leq 0 \\
&\overbrace{P_w - (p+\epsilon)}^{h_{II}} \leq 0 \\
&\overbrace{(\alpha-\epsilon)U(P_w,\rho) + (1-\alpha+\epsilon)L(P_w,\rho) - Q_w}^{h_{III}} \leq 0 \\
&\overbrace{Q_w - \alpha U(P_w,\rho) - (1-\alpha)L(P_w,\rho)}^{h_{IV}} \leq 0
\end{aligned}$$

Since P_w, Q_w, U and L are all linear functions of w with the constants X, y, D_l and ρ, this formulation corresponds to an optimization problem with a quadratic target and linear constraints. We solve using the Lagrange multipliers:

$$L(w, \lambda_1, \lambda_2, \mu_1, \mu_2) = 0.5 \cdot \sum_{i=1}^{d} w_i^2 + \lambda_1 h_I + \lambda_2 h_{II} + \mu_1 h_{III} + \mu_2 h_{IV}$$

We want to minimize L s.t $\lambda_1, \lambda_2, \mu_1, \mu_2 \geq 0$.

First we consider the pair h_I and h_{II}; if both constraints are summed the result is,

$$p - P_w + P_w - (p+\epsilon) = -\epsilon < 0$$

Therefore at least one of the constraints is strictly negative (otherwise their sum cannot be negative). Since L is to be minimized, for every w the solution $(w, \lambda_1, \lambda_2 ...)$,

is going to select at least one of $\lambda_1 > 0$ or $\lambda_2 > 0$ in order to use the negative term and reach a smaller value of L. According to the complementary slackness this means that either $P_w = p$ or $P_w = p + \epsilon$ (of course we cannot have both).

Next, we consider constraints h_{III} and h_{IV}; we denote with Δ the sum between these two constraints:

$$\begin{aligned}\Delta =&(\alpha - \epsilon)U(P_w, \rho) + (1 - \alpha + \epsilon)L(P_w, \rho) - Q_w + Q_w - \alpha U(P_w, \rho) - (1 - \alpha)L(P_w, \rho)\\ =&\epsilon\left(L(P_w, \rho) - U(P_w, \rho)\right)\end{aligned}$$

Since $\epsilon > 0$ we only need to consider the term $(L(P_w, \rho) - U(P_w, \rho))$. We have three cases: $P_w = 0$, $P_w = 1$ and $0 < P_w < 1$.

For the cases $P_w = 0$ and $P_w = 1$, according to the definition of U and L $(L(0, \rho) - U(0, \rho)) = 0$ or $(L(1, \rho) - U(1, \rho)) = 0$ correspondingly. For both $\Delta = 0$, and both h_{III} and h_{IV} collapse to $Q_w = 0$ or ρ. Therefore the solution is attained either at $(0, 0)$ for when $P_w = 0$ or at $(1, \rho)$ for when $P_w = 1$.

Next, since for $0 < P_w < 1$, $L(P_w, \rho) - U(P_w, \rho) < 0$ and $\epsilon > 0$ therefore $\Delta < 0$. Similarly to the P_w constraints, according to the complementary slackness, this means that either h_{III} or h_{IV} is tight.

Therefore each minimal norm solution occurs when exactly one of h_I or h_{II} is tight, and when one of h_{III} or h_{IV} is tight. In each of those four cases, the solution occurs in one of the corners of the cell, which in turn is the same as solving the optimization problem for constant values of P_w and Q_w for each of the corners of the cell and taking the best solution. □

D.3 Given value P_w search over Q_w

This section takes a different approach then the previous ones. We search for a feasible stochastic linear classifier given a value for only P_w. Instead of minimizing the norm of the classifier, we search for Q_w values that minimize the WGI under the constraint that the solution is feasible.

The following lemma characterizes the solution when we minimize directly WGI for a given P_w. (Note that when we fix the value of P_w the function WGI is convex.)

Lemma D.3.1. *Let $\vec{a} = D_v^\top X$ and $\vec{b} = (D_v \odot y)^\top X$, and assume that they are linearly independent. Let $\vec{\bar{b}} = \vec{b} - \frac{(\vec{a}\cdot\vec{b})}{\|\vec{a}\|^2}\vec{a}$. Then for:*

$$w^* = \arg\min WGI(p, Q_w)$$
$$P_w = p$$

then the two solutions are,

$$w^* = \frac{2p-1}{\|\vec{a}\|^2}\vec{a} \pm \frac{\sqrt{\|\vec{a}\|^2 - (2p-1)^2}}{\|\vec{a}\|\|\vec{\bar{b}}\|}\vec{\bar{b}}$$

Proof. The first observation we have is that for a fixed $P_w = p$, $q = \rho p$ is a maximum for WGI, and since the second derivative of WGI is negative, we get that WGI attains a minimal value when Q_w is either maximized or minimized. Therefore we substitute the original problem with the following two optimization problems

$$Q^{\max} = \arg\max Q_w \quad \text{and} \quad Q^{\min} = \arg\min Q_w$$
$$P_w = p \qquad\qquad P_w = p$$
$$\|w\|^2 \leq 1 \qquad\qquad \|w\|^2 \leq 1$$

where the first constraint is used to fix P_w and the second constraint is used to bound the norm of w. Now, we will translate the maximization while using the previous expressions for $\vec{a}$, $\vec{b}$ and set $\bar{p} = 2p - 1$.

$$\max \vec{\bar{b}}^\top w$$
$$\vec{a}^\top w - \bar{p} = 0$$
$$\|w\|^2 \leq 1$$

We rewrite $\vec{b} = \alpha\vec{a} + \vec{\bar{b}}$ as a sum of:

1. $\alpha \cdot \vec{a}$ which is $\vec{b}$'s projection over $\vec{a}$.
2. $\vec{\bar{b}}$ which is a vector that is orthogonal to a, i.e., $(a \cdot \bar{b}) = 0$. Since a and b are linearly independent we have $\vec{\bar{b}} \neq 0$.

We write w as:

$$w = \mu_1 \vec{a} + \mu_2 \vec{\bar{b}} + \vec{w}$$

where $\vec{w}$ is orthogonal to $\vec{a}$ and $\vec{\bar{b}}$. The optimization becomes:

$$\begin{aligned} \max\ & \alpha\mu_1 \|\vec{a}\|^2 + \mu_2 \|\vec{\bar{b}}\|^2 \\ & \mu_1 \|\vec{a}\|^2 - \bar{p} = 0 \\ & \mu_1^2 \|\vec{a}\|^2 + \mu_2^2 \|\vec{\bar{b}}\|^2 + \|\vec{w}\|^2 \leq 1 \end{aligned}$$

When we substitute $\mu_1 = \frac{\bar{p}}{\|\vec{a}\|^2}$ we get:

$$\begin{aligned} \max\ & \alpha\bar{p} + \mu_2 \|\vec{\bar{b}}\|^2 \\ & \frac{\bar{p}^2}{\|\vec{a}\|^2} + \mu_2^2 \|\vec{\bar{b}}\|^2 + \|\vec{w}\|^2 \leq 1 \end{aligned}$$

Since $\alpha\bar{p}$ is a constant and $\|\vec{\bar{b}}\|^2$ is positive, the optimization is equivalent to:

$$\begin{aligned} \max\ & \mu_2 \\ & \mu_2^2 \|\vec{\bar{b}}\|^2 + \|\vec{w}\|^2 \leq 1 - \frac{\bar{p}^2}{\|\vec{a}\|^2} \end{aligned}$$

Notice that since only μ_2 is a variable that contributes to the max, we would like to make it as large as possible under the constraint. Therefore the optimal solution would be attained when the $\bar{w} = 0$, and the constraint is strictly equal. Thus the expression of μ_2 would be:

$$\mu_2^2 \|\vec{\bar{b}}\|^2 = 1 - \frac{\bar{p}^2}{\|\vec{a}\|^2} \quad \Rightarrow \quad \mu_2 = \pm \frac{\sqrt{\|\vec{a}\|^2 - \bar{p}^2}}{\|\vec{a}\| \|\vec{\bar{b}}\|}$$

The maximal solution for the optimization is achieved with a positive μ_2 in the above expression:

$$\mu_2 = \frac{\sqrt{\|\vec{a}\|^2 - \bar{p}^2}}{\|\vec{a}\| \|\vec{\bar{b}}\|}$$

Finally the solution for w is,

$$w = \frac{\bar{p}}{\|\vec{a}\|^2}\vec{a} + \frac{\sqrt{\|\vec{a}\|^2 - \bar{p}^2}}{\|\vec{a}\|\|\vec{b}\|}\vec{b}$$

Notice that for the minimization problem $\min Q_w$ we simply select the negative value of μ_2, and get,

$$w = \frac{\bar{p}}{\|\vec{a}\|^2}\vec{a} - \frac{\sqrt{\|\vec{a}\|^2 - \bar{p}^2}}{\|\vec{a}\|\|\vec{b}\|}\vec{b}$$

This implies that the solution is either one of:

$$w^*_{1,2} = \frac{2p-1}{\|\vec{a}\|^2}\vec{a} \pm \frac{\sqrt{\|\vec{a}\|^2 - (2p-1)^2}}{\|\vec{a}\|\|\vec{b}\|}\vec{b}$$

□

D.4 Feasible Ranges for P_w and Q_w

In all of our problems we have constraints such as $P_w = p$ and $Q_w = q$ (or their range variants). However we should note that not all possible values of p and q are applicable for a given data:

Lemma D.4.1. *Let $\vec{a} = D_v^\top X$ and $\vec{b} = (D_v \odot y)^\top X$ and let w be a feasible stochastic linear classifier. Then:*

$$0.5 - 0.5\|\vec{a}\| \leq P_w \leq 0.5 + 0.5\|\vec{a}\|$$
$$0.5\rho - 0.5\|\vec{b}\| \leq Q_w \leq 0.5\rho + 0.5\|\vec{b}\|$$

Proof. First:

$$-\|\vec{a}\|\|w\| \leq \vec{a}^\top \cdot w \leq \|\vec{a}\|\|w\| \rightarrow -\|\vec{a}\| \leq \vec{a}^\top \cdot w \leq \|\vec{a}\|$$
$$-\|\vec{b}\|\|w\| \leq \vec{b}^\top \cdot w \leq \|\vec{b}\|\|w\| \rightarrow -\|\vec{b}\| \leq \vec{b}^\top \cdot w \leq \|\vec{b}\|$$

where the left-hand side is a property of $\cdot$, and the right-hand is because w is feasible. Applying the above inequalities in eq 6.4: $P_w = 0.5\vec{a}^\top \cdot w + 0.5$ and eq 6.5: $Q_w = 0.5\vec{b}^\top \cdot w + 0.5\rho$ concludes the proof.

□

Appendix E

Approximation Algorithms

This section details the three algorithms discussed in theorem 7.0.2:

Theorem 7.0.2. Let S be a sample, D_v be a distribution over S, and ϵ be an approximation parameter. Let $\vec{a} = D_v^\top X$ and $\vec{b} = (D_v \odot y)^\top X$. Then:

1. For the case where $\vec{a}$ and $\vec{b}$ are linearly independent, algorithms `FixedPQ` and `FixedPSearchQ` guarantee an ϵ-approximation for WGI using a stochastic linear classifier. Algorithm `FixedPQ` runs in time $O(Nd)+O\left(\epsilon^{-2}\log\left(\epsilon^{-1}\right)\right)$, and algorithm `FixedPSearchQ` runs in time $O(Nd) + O\left(\frac{d}{\epsilon}\right)$.

2. For the case where $\vec{a}$ and $\vec{b}$ are linearly dependent, i.e., $\vec{b} = \lambda\vec{a}$, algorithm `DependentWGI` runs in time $O(Nd)$, and achieves the optimal result for WGI using a stochastic linear classifier.

For each algorithm we provide pseudo code, correctness proof and a complexity analysis: Algorithm `FixedPQ` is in Appendix E.1, Algorithm `FixedPSearchQ` is in Appendix E.2 and Algorithm `DependentWGI` is in Appendix E.3.

As mentioned all three algorithms depend on $\vec{a}$ the weighted average of the data, and $\vec{b}$ the weighted average of the positive class. We note here that both $\vec{a}$ and $\vec{b}$, along with other static related computations such as $\|\vec{a}\|, \|\vec{b}\|$ are treated as global variables which are computed once in the beginning of each algorithm. We

also note that the constant $C = 100$ used in the pseudo code corresponds to the bound from Appendix C.

E.1 `FixedPQ` Approximate minimal Gini Index in cells of P_w, Q_w

In this section we describe Algorithm `FixedPQ`. After computing the expressions as described earlier, we generate a list $candidates$ that contains pairs (p, q) which are corners of the trapezoid cells:

Initially we generate a set of equally spaces P_w values from the interval $[\max(0, 0.5(1 - \|\vec{a}\|)),\ \min(1, 0.5(1 + \|\vec{a}\|))]$. This interval is the intersection of the domain of WGI and the feasible range of w as described in Appendix D.4. The set of P_w values also includes the values for ρ and $1 - \rho$ in order to ensure that the candidates partition the domain according to the regions defined in Appendix C.

Next, for each P_w value we define equally spaced values in ranges:

1. $[\max(0, p + \rho - 1, 0.5(\rho - \|\vec{b}\|)), \min(\rho p, 0.5(\rho + \|\vec{b}\|))]$
2. $[\max(\rho p, 0.5(\rho - \|\vec{b}\|)), \min(\rho, p, 0.5(\rho + \|\vec{b}\|))]$

Again, these ranges take into consideration both the sub-domains of the WGI and the feasible ranges of the stochastic linear classifier.

The equally spaced values are generated using the $GetRange$ function, and we always partition into $\frac{C}{\epsilon}$ values in order to achieve an approximation of ϵ (see correctness below). This results in $O(\epsilon^{-1})$ values of P_w, and for each of those we have two times $O(\epsilon^{-1})$ for the Q_w values below and above the line $Q_w = \rho P_w$ which separates the regions. Thus, the total number of candidates is $O(\epsilon^{-2})$.

The list $candidates$ now contains pairs of (P_w, Q_w) which are sorted according to WGI in an ascending order. For each point made of P_w and Q_w values will try to fit a stochastic linear classifier: First we use the function

$GetCandidateWeightsTerms$ to get the scalars α and β. Since we also know $\|\vec{a}\|$, $\|\vec{b}\|$ and $(\vec{a} \cdot \vec{b})$ we measure the feasibility of the classifier w before computing it. Once we find a feasible classifier, we compute it explicitly and return it.

Algorithm correctness Denote w^* the optimal feasible solution with respect to WGI: $WGI_{w^*} = \min_{\{w|\|w\|\leq 1\}} WGI_w$. Let $Cells$ be the union of non-intersecting trapezoid cells parametrized by step size ϵ/C. We denote $c^* \in Cells$ the cell which contains w^*. Denote W_c the feasible solutions proposed to cell $c \in Cells$ by the optimization in Appendix D.2 (since w^* is feasible, we know that at least one vertex is feasible). Specifically for c^*, we denote the best proposed solution as w_1:

$$w_1 = \arg \min_{w \in W_{c^*}} (WGI(w))$$

Since both $w^*, w_1 \in c^*$ according to the bound we get $WGI_{w^*} \leq WGI_{w_1} \leq WGI_{w^*} + \epsilon$.

Next, we define W_{Cells} as the union of the solutions from all the cells: $W_{Cells} = \bigcup_{c \in Cells} W_c$. W_{Cells} is the collection of all the classifiers that would have been created from evaluating every point in candidates. Since $W_{c^*} \subset W_{Cells}$ we have $w_1 \in W_{Cells}$. Let w_2 be the minimal solution from all the solutions in W_{Cells}:

$$w_2 = \arg \min_{w \in W_{Cells}} (WGI_w)$$

In particular, $WGI_{w^*} \leq WGI_{w_2} \leq WGI_{w_1} \leq WGI_{w^*} + \epsilon$, and we notice that w_2 is actually the result returned by the algorithm. We therefore conclude that the algorithm returns an ϵ approximation to the optimal WGI.

Time complexity We start by computing ρ which consists of a dot-product over vectors of dimension N and takes time $O(N)$. Next, we calculate $\vec{a}$, $\vec{b}$, by multiplying an N dimensional vector by an $N \times O(d)$ matrix, which is done in time $O(Nd)$. Computing the expressions that depend on $\vec{a}$ and $\vec{b}$ is done once and takes time $O(d)$.

Algorithm `FixedPQ` (X - *dataset,* D_v - *data distribution,* y - *labels,* ϵ - *approximation range*)

Store globally: $\rho \leftarrow D_v \cdot y, \vec{a}^\top \leftarrow D_v^t X, \vec{b}^\top \leftarrow (D_v \odot y)^\top X$

Compute once and store globally: $\|\vec{a}\|, \|\vec{b}\|, (\vec{a} \cdot \vec{b})$

candidates $\leftarrow$ new List()

foreach $p \in$ `GetRange` $(\max(0, 0.5(1 - \|\vec{a}\|)), \min(1, 0.5(1 + \|\vec{a}\|)), \frac{C}{\epsilon}) \bigcup \{\rho, 1 - \rho\}$ **do**

  foreach $q \in$ `GetRange` $(\max(0, p + \rho - 1, 0.5(\rho - \|\vec{b}\|)), \min(\rho p, 0.5(\rho + \|\vec{b}\|)), \frac{C}{\epsilon})$ **do**

    candidates.Insert((p, q))

  end

  foreach $q \in$ `GetRange` $(\max(\rho p, 0.5(\rho - \|\vec{b}\|)), \min(\rho, p, 0.5(\rho + \|\vec{b}\|)), \frac{C}{\epsilon})$ **do**

    candidates.Insert((p, q))

  end

end

candidates $\leftarrow$ candidates.Sort(ascending=True, by = $WGI(p, q)$)

foreach $p, q \in$ *candidates* **do**

  $(\alpha, \beta) \leftarrow$ `GetCandidateWeightsTerms` (p, q)

  if $\alpha^2 \cdot \|\vec{a}\|^2 + 2\alpha\beta \cdot (\vec{a} \cdot \vec{b}) + \beta^2 \cdot \|\vec{b}\|^2 \leq 1$ **then**

    return $\left(\alpha \cdot \vec{a} + \beta \cdot \vec{b}, WGI(p, q)\right)$

  end

end

return *"Error - Impossible"*;

Procedure `GetCandidateWeightsTerms` (p, q)

$\bar{p} \leftarrow 2p - 1$, $\bar{q} \leftarrow 2q - \rho$

return $\left(\frac{\bar{q} \cdot (\vec{a} \cdot \vec{b}) - \bar{p} \cdot \|\vec{b}\|^2}{\|\vec{a}\|^2 \cdot \|\vec{b}\|^2 - (\vec{a} \cdot \vec{b})^2}, \frac{\bar{p} \cdot (\vec{a} \cdot \vec{b}) - \bar{q} \cdot \|\vec{a}\|^2}{\|\vec{a}\|^2 \cdot \|\vec{b}\|^2 - (\vec{a} \cdot \vec{b})^2}\right)$

Procedure `GetRange` (*lower, upper, steps*)

return $\{p|$ *equally spaced points in* [*lower,upper*] *with step=* $\frac{\text{upper - lower}}{\text{steps}}\}$

Algorithm 2: Approximating optimal WGI with 2 ranges

We create a list $candidates$, consisting of $O(\epsilon^{-2})$ pairs (p, q) in time $O(\epsilon^{-2})$. Next, we compute the WGI for each candidate (p, q) in $O(1)$ time, and we sort them using the WGI values in time $O\left(\epsilon^{-2}\log\left(\epsilon^{-2}\right)\right) = O\left(\epsilon^{-2}\cdot\log\left(\epsilon^{-1}\right)\right)$.

Finally we try to match a classifier for each point: we iterate over (sorted) $candidates$, and we call the function $GetCandidateWeightsTerms$ which runs in time $O(1)$ since it is using the pre-calculated expressions. There are at most $O\left(\epsilon^{-2}\right)$ items in the list. Therefore this step costs $O\left(\epsilon^{-2}\right)$. Once a solution is found to be feasible , only then we compute the weight vector in time $O(d)$.

The final complexity is:

$$O(Nd) + O(d) + O\left(\epsilon^{-2}\right) + O\left(\epsilon^{-2}\cdot\log\left(\epsilon^{-1}\right)\right) + O\left(\epsilon^{-2}\right) + O(d)$$
$$= O(Nd) + O\left(\epsilon^{-2}\cdot\log\left(\epsilon^{-1}\right)\right)$$

E.2 `FixedPSearchQ` Approximate minimal Gini Index in range of P_w

In this section we describe Algorithm `FixedPSearchQ`. After computing the expressions as described earlier, we generate a list $candidates$ that contains values p which are equally spaced values on the feasible range of P_w:

This set is generated from the interval $[\max(0, 0.5(1 - \|\vec{a}\|)), \min(1, 0.5(1 + \|\vec{a}\|))]$. This interval is the intersection of the domain of WGI and the feasible range of w as described in Appendix D.4. The set of P_w values also includes the values for ρ and $1 - \rho$ in order to ensure that the candidates partition the domain according to the regions defined in Appendix C.

The equally spaced values are generated using the $GetRange$ function, and we partition into $\frac{C}{\epsilon}$ values in order to achieve an approximation of ϵ (see correctness below). This results in $O(\epsilon^{-1})$ values of P_w.

Next, we compute $\vec{\bar{b}}$ which is $\vec{b}$ minus $\vec{a}$'s projection over it. This allows for the computation of $\vec{u}$ and $\vec{v}$. These vectors are components of the final solution.

Finally, for each P_w we fit two solutions which are a function of P_w, $\vec{u}$ and $\vec{v}$. We keep the best result $\hat{w}$ and its matching WGI $\hat{g}$ using the function $SetCandidateIfBetter$, and return $\hat{w}$ and $\hat{g}$ at the end of the loop.

Algorithm correctness Given an ϵ parameter for the algorithm, we know that the optimal feasible solution w^* belongs to some range of P_w in the grid, denote that range as $C^* = [\alpha, \beta]$.

Denote $p^* = P_{w^*}, q^* = Q_{w^*}$. First we note that $p^* \in C^*$ (by definition). Next, we consider C_q as the Q range that would have included w^* if we were to partition into trapezoid cells. Since w^* is feasible, at least one of the corners of $C^* \times C_q$ is feasible. There is a feasible solution that we denote as w_c at α or β. Without loss of generality assume it is α. According to the bound on the WGI in Appendix C, this candidate solution is at most ϵ away from w^*. Since our algorithm considers all the possible feasible solutions for α it also considers w_c and therefore the result of the optimization problem as presented in Appendix D.3 denoted as w_1 is at least as good as w_c:

$$WGI_{w_1} - WGI_{w^*} \leq WGI_{w_c} - WGI_{w^*} \leq \epsilon$$

Finally, if the algorithm returns a different solution w_2 it is only because $WGI_{w_2} \leq WGI_{w_1}$, and we have

$$WGI_{w^*} \leq WGI_{w_2} \leq WGI_{w_1} \leq WGI_{w^*} + \epsilon$$

namely, w_2 is also an ϵ approximation.

Time complexity We start by computing ρ which consists of a dot-product over vectors of dimension N and takes time $O(N)$. Next, we calculate $\vec{a}$, $\vec{b}$, by multiplying an N dimensional vector by an $N \times O(d)$ matrix, which is done in time $O(Nd)$. Computing the expressions that depend on $\vec{a}$ and $\vec{b}$ is done once and takes time $O(d)$.

Algorithm `FixedPSearchQ`(X - *dataset,* D_v - *data distribution,* y - *labels,* ϵ - *approximation range*)

Store globally: $\rho \leftarrow D_v \cdot y, \vec{a}^\top \leftarrow D_v^t X, \vec{b^\top} \leftarrow (D_v \odot y)^\top X$

Compute once and store globally: $\|\vec{a}\|, \|\vec{b}\|, (\vec{a} \cdot \vec{b})$

$\vec{\bar{b}} \leftarrow \vec{b} - \frac{(\vec{a} \cdot \vec{b})}{\|\vec{a}\|^2} \cdot \vec{a}$

$\vec{u} = \frac{1}{\|\vec{a}\|^2} \cdot \vec{a}$, $\vec{v} = \frac{1}{\|\vec{a}\| \|\vec{\bar{b}}\|} \cdot \vec{\bar{b}}$

$\hat{w} \leftarrow \perp, \hat{g} \leftarrow \perp$

foreach $p \in$ `GetRange` *(*$\max(0, 0.5(1 - \|\vec{a}\|))$, $\min(1, 0.5(1 + \|\vec{a}\|))$, $\frac{C}{\epsilon}$*)*$\bigcup\{\rho, 1 - \rho\}$ **do**

  $\bar{p} \leftarrow 2p - 1$

  $w_1 \leftarrow \bar{p} \cdot \vec{u} + \sqrt{\|\vec{a}\|^2 - \bar{p}^2} \cdot \vec{v}$

  $\hat{w}, \hat{g} \leftarrow$ `SetCandidateIfBetter`(w_1,$\hat{w}$,$\hat{g}$,p)

  $w_2 \leftarrow \bar{p} \cdot \vec{u} - \sqrt{\|\vec{a}\|^2 - \bar{p}^2} \cdot \vec{v}$

  $\hat{w}, \hat{g} \leftarrow$ `SetCandidateIfBetter`(w_2,$\hat{w}$,$\hat{g}$,p)

end

if $\hat{w} = \perp$ **then**

  return *"Error - Impossible"*

end

return $\hat{w}, \hat{g}$

Procedure `SetCandidateIfBetter`(w,$\hat{w}$,$\hat{g}$,p)

$q \leftarrow \frac{\vec{b}^\top w + \rho}{2}$

if $WGI(p, q) \leq \hat{g}$ **then**

  return $w, WGI(p, q)$

end

else

  return $\hat{w}, \hat{g}$

end

Procedure `GetRange`(*lower, upper, steps*)

return $\{p|$ *equally spaced points in* $[$*lower,upper*$]$ *with* *step*$=\frac{\text{upper - lower}}{\text{steps}}\}$

Algorithm 3: Approximating optimal WGI with one range over P_w

Finally the loop iterates over $O(\epsilon^{-1})$ values of p. For each such value, the algorithm computes a weight vector and uses the method $SetCandidateIfBetter$ which has time complexity $O(d)$. The total cost of all the iterations is $O(\frac{d}{\epsilon})$.

The final complexity is:

$$O(Nd) + O(d) + O\left(\frac{d}{\epsilon}\right) = O(Nd) + O\left(\frac{d}{\epsilon}\right)$$

E.3 `DependentWGI` Dependent constraints case

In this section we describe Algorithm `DependentWGI`. Since we have an analytical solution (Appendix C.9), we simply apply it and return its result as w and the corresponding WGI g.

Algorithm `DependentWGI` (X - *dataset,* D_v - *data distribution,* y - *labels*)

Store globally: $\rho \leftarrow D_v \cdot y, \vec{a}^\top \leftarrow D_v^t X, \vec{b}^\top \leftarrow (D_v \odot y)^\top X$

Compute once and store globally: $\|\vec{a}\|$

Get λ s.t $\vec{b} = \lambda \vec{a}$ ▷ According to our assumption, this always happens

$w \leftarrow \frac{\vec{a}}{\|\vec{a}\|}$

$g \leftarrow 4\left(\rho - \frac{(\rho+\lambda\|\vec{a}\|)^2}{2(1+\|\vec{a}\|)} - \frac{(\rho-\lambda\|\vec{a}\|)^2}{2(1-\|\vec{a}\|)}\right)$

return w, g

Algorithm 4: Approximating optimal WGI dependent case

Algorithm correctness See Appendix C.9.

Algorithm complexity We start by calculating ρ which consists of dot-product over vectors of dimension N in time $O(N)$. Next, we calculate $\vec{a}$ and $\vec{b}$, by multiplying an N vector by an $N \times O(d)$ matrix in time $O(Nd)$. Also, we calculate $\|\vec{a}\|$ in time $O(d)$. And finally, we extract λ which is $O(1)$ without verification

(simply, $\lambda = \frac{\vec{b}_1}{\vec{a}_1}$), or $O(d)$ with verification (make sure that $\forall i \in [1, d] : \lambda = \frac{\vec{b}_i}{\vec{a}_i}$). We compute w in time $O(d)$ and the drop in constant time. The final complexity is:

$$O(Nd) + O(d) + O(d) = O(Nd)$$

Appendix F

WGI not convex

In this section we are going to show that the WGI is non convex in w.

Lemma F.0.1. *The function WGI_w is non convex*

Proof. Let $x_1, x_2 \in R^d$ be samples s.t $x_{1,1} = -x_{2,1} = 2x$ and $\forall i \in [2, d] : x_{1,i} = x_{2,i}$ (x_1 and x_2 are opposite in the first dimension, and the same in all other). Let $y_1 = 1$ be the label of x_1 and let $y_2 = 0$ be the label of x_2. Also let D be the data distribution s.t $D(x_1) = D(x_2) = 0.5$. We note that this entails that $\rho = 0.5$ and that $\vec{a}_1 = 0$ while $\vec{b}_1 = x$.

We define three feasible stochastic linear classifiers:

1. $w_0 = \vec{0}$ all zeros vector
2. w_1 whose first coordinate is 1 and the rest 0: $w_{1,1} = 1, \forall i \in [2, d] : w_{1,i} = 0$
3. w_m whose first coordinate is 0.5 and the rest 0: $w_{1,1} = 0.5, \forall i \in [2, d] : w_{1,i} = 0$.

We see that

$$\vec{a}^\top w_0 = \vec{a}^\top w_1 = \vec{a}^\top w_m = 0$$
$$\vec{b}^\top w_0 = 0, \;\; \vec{b}^\top w_1 = x, \;\; \vec{b}^\top w_m = 0.5x$$

Sice $WGI_w = 4\left(\rho - \frac{Q_w^2}{P_w} - \frac{(\rho - Q_w)^2}{1-P_w}\right)$ by equations (6.4) and (6.5) we get:

$$WGI_w = 4\left(\rho - \frac{(\frac{\vec{b}^\top \cdot w + \rho}{2})^2}{\frac{\vec{a}^\top \cdot w + 1}{2}} - \frac{(\rho - \frac{\vec{b}^\top \cdot w + \rho}{2})^2}{1 - \frac{\vec{a}^\top \cdot w + 1}{2}}\right) = 4\left(\rho - \frac{(\rho + \vec{b}^\top \cdot w)^2}{2(1 + \vec{a}^\top \cdot w)} - \frac{(\rho - \vec{b}^\top \cdot w)^2}{2(1 - \vec{a}^\top \cdot w)}\right)$$

We substitute the values of ρ, $\vec{a}^\top \cdot w$ and $\vec{b}^\top \cdot w$:

$$WGI_{w_0} = 4\left(0.5 - \frac{0.5^2}{2} - \frac{0.5^2}{2}\right) = 1$$
$$WGI_{w_1} = 4\left(0.5 - \frac{(0.5+x)^2}{2} - \frac{(0.5-x)^2}{2}\right) = 4\left(0.25 - x^2\right) = 1 - 4x^2$$
$$WGI_{w_m} = 4\left(0.5 - \frac{(0.5+0.5x)^2}{2} - \frac{(0.5-0.5x)^2}{2}\right) = 1 - x^2$$

Since $w_m = 0.5 \cdot w_0 + 0.5 \cdot w_1$, in order to show non convexity we show that $WGI_{w_m} > 0.5 \cdot WGI_{w_0} + 0.5 \cdot WGI_{w_1}$:

$$1 - x^2 > 0.5 \cdot 1 + 0.5(1 - 4x^2) = 1 - 2x^2$$

which concludes the proof. □

Appendix G

Approximate γ Margin With a Stochastic Tree

In this appendix we show that the class of decision trees with stochastic linear classifiers has an efficient representation with a bounded error for a γ margin feasible linear separator. This is show by the following theorem:

Theorem G.0.1. Let w be a γ margin feasible linear separator, then there exists a stochastic tree T with feasible linear separators in the internal nodes such that the error of the tree is bounded by ϵ and the depth of T is $O\left(\ln\left(\epsilon^{-1}\right)\gamma^{-2}\right)$.

Proof. Let $y \in \{0,1\}$ be a binary label. Let w be a linear separator such that $\forall x : (2y-1)\cdot w \cdot x \geq \gamma$. We denote Z_i as a random variable such that: $\Pr\left(Z_i = 1\right) = \frac{1+w\cdot x}{2}$ and $\Pr\left(Z_i = -1\right) = \frac{1-w\cdot x}{2}$. Let $S = \sum_{i=0}^{h} Z_i$ be a random variable which is the sum of h independent Z_is. We note that $E\left[Z_i\right] = 1\cdot\Pr\left(Z_i = 1\right) - 1\cdot\Pr\left(Z_i = -1\right) = w\cdot x$ and therefore by the expectation of a sum $E\left[S\right] = \sum_{i=0}^{h} E[Z_i] = h(w\cdot x)$.

Let x be a positive sample, meaning $w\cdot x \geq \gamma$. We bound the probability of

the sum S to be negative by using Hoeffding's inequality:

$$\Pr(S \leq 0) = \Pr(S - E[S] \leq -E[S]) = \Pr(S - E[S] \leq -h(w \cdot x))$$
$$\leq \exp \frac{-2(h(w \cdot x))^2}{\sum_{i=0}^{h} 2^2} = \exp \frac{-h(w \cdot x)^2}{2}$$

Since $w \cdot x \geq \gamma$, we get $\Pr(S \leq 0) \leq \exp \frac{-h\gamma^2}{2}$, and if we want to drive the probability below ϵ we have:

$$\Pr(S \leq 0) \leq \exp \frac{-h\gamma^2}{2} \leq \epsilon \rightarrow \frac{-h\gamma^2}{2} \leq \ln \epsilon \rightarrow h \geq 2 \ln(\epsilon^{-1})\gamma^{-2}$$

Similarly, for the negative sample for $\Pr(S \geq 0) \leq \epsilon$ we also require $h \geq 2 \ln(\epsilon^{-1})\gamma^{-2}$.

We now describe a stochastic decision tree T: T is a full tree of depth $h \geq 2 \ln(\epsilon^{-1})\gamma^{-2}$. Each internal node contains the stochastic linear separator w and each leaf $l \in leaves(T)$ is labeled 1 if the path from $root(T)$ to l contains more left turns than right and 0 otherwise.

We notice that when predicting for sample x_i using T, the decision in each internal node is modeled by Z_i. A positive sample x is classified positive according to the expectation over the leaves:

$$\Pr(y_i = 1) = \sum_{l \in leaves(T)} label(l) \Pr(Reach(x, l))$$

We consider all the positive labels, since the negative labels contribute 0 to the expectation:

$$\Pr(y_i = 1) = \sum_{l \in leaves(T) | label(l)=1} \Pr(Reach(x, l))$$

By the construction of T, this probability equals the probability of reaching leaves with more left turns than right (therefore these leaves are labeled as 1). Namely,

$$\Pr(y_i = 1) = \sum_{l \in leaves(T)} label(l) \Pr(Reach(x, l)) = \Pr(S \geq 0) \geq 1 - \epsilon$$

And therefore the error probability $\Pr(y_i = 0) =\leq \epsilon$. Similarly, we can show that the error for a negative sample is also bounded by ϵ. Finally this ensures that the expected error is also smaller than ϵ as required. □

Appendix H

Splitting criteria bounds the classification error

In this appendix we are going to show that a *permissible splitting criterion* G, upper bound twice the classification error. Reminder: a function G over $[0, 1]$ is a permissible splitting criterion if:

1. G is symmetric about 0.5. Thus, $\forall x \in [0, 1] : G(x) = G(1 - x)$.
2. G is normalized: $G(0.5) = 1$ and $G(0) = G(1) = 0$.
3. G is strictly concave.

Next, we prove the following theorem:

Theorem H.0.1. Let G be a permissible splitting criterion, then for $\forall x \in [0, 1]$: $G(x) \geq 2 \min\{x, 1 - x\} = \min\{2x, 2 - 2x\}$.

Proof. First, we consider $x \in [0, 0.5]$, and we notice that $\min\{2x, 2 - 2x\} = 2x$. Next, we notice that $x = (1 - 2x) \cdot 0 + 2x \cdot 0.5$ and according to concavity of G we derive the bound for this interval of x:

$$G(x) = G((1 - 2x) \cdot 0 + 2x \cdot 0.5) \geq (1 - 2x)G(0) + 2xG(0.5) = 2x$$

Where the last transition is due to $G(0.5) = 1$ and $G(0) = G(1) = 0$.

Next, we consider $x \in [0.5, 1]$, and we notice that $\min\{2x, 2 - 2x\} = 2 - 2x$. Again, we notice that $x = (2 - 2x) \cdot 0.5 + (1 - (2 - 2x)) \cdot 1$ and according to concavity of G we derive the bound for this interval of x:

$$G(x) = G((2 - 2x) \cdot 0.5 + (1 - (2 - 2x)) \cdot 1) \geq (2 - 2x)G(0.5) + (1 - (2 - 2x))G(1) = 2 - 2x$$

Where the last transition is due to $G(0.5) = 1$ and $G(0) = G(1) = 0$.

The above two cases cover the interval $x \in [0, 1]$ and therefore we can conclude that $\forall x \in [0, 1] : G(x) \geq 2 \min\{x, 1 - x\} = \min\{2x, 2 - 2x\}$. □

תקציר

בעבודה זו אנחנו בוחנים את בעיית למידת עצי החלטה תחת מסגרת ההגברה (boosting), בה אנחנו מניחים כי המסווגים בצמתים הפנימיים של העץ שייכים למחלקת השערות אשר מספקת את השערת הלמידה החלשה (Weak Learning Assumption).

אנחנו חוקרים את מחלקת ההשערות של מסווגים לינאריים סטוכסטיים, כמחלקת השערות המספקת את השערת הלמידה החלשה. עבור מחלקה זו אנחנו מספקים אלגוריתמים יעילים למזעור הג'יני אינדקס (Gini Index) שהוא חסם על השגיאה אפילו שהבעיה אינה קמורה. לכן, תחת הנחת הלמידה חלשה אנחנו מספקים אלגוריתמים יעילים ובעל חסמי שגיאה ללמידה של עצי החלטה המשתמשים במסווגים לינאריים סטוכטיים.

לבסוף אנחנו בוחנים את האלגוריתם גם עבור מקרים בהם הנתונים הם סינטטיים וגם עבור נתונים אמיתיים. היתרון במחלקה הנ"ל היא היכולת להתחשב ביותר ממאפיין אחד של הנתונים בכל רגע נתון, לכן אנחנו משווים את הביצועים של השיטה המוצעת כנגד הביצועים של מפריד לינארי – Support Vector Machine עם קרנל לינארי וכנגד עץ החלטה המבוסס "גדי החלטות" (decision stumps) ומראים כי עבור נתונים ממימד גבוה עם גבול החלטה מורכב המסווג המוצע עובד טוב יותר מהשניים האחרים.

אוניברסיטת תל אביב

הפקולטה למדעים מדויקים

על שם ריימונד ובברלי סאקלר

בית הספר למדעי המחשב

לימוד עצי החלטות סטוכסטיים עם מסווגים לינאריים

חיבור זה הוגש כחלק מהדרישות לקבלת התואר

"מוסמך אוניברסיטה" – M.Sc.

באוניברסיטת תל אביב

בית הספר למדעי המחשב

על ידי

תום יורגנסון

עבודת המחקר לתזה זו נערכה

באוניברסיטת תל אביב בהנחייתו של

פרופסור ישי מנצור

יוני 2017