Statistical Learning, Fall 2022

# Homework exercise 4

Due date: 17 January 2023 before class — no extensions as we will discuss it in class

1. **Playing around with trees**
   Run a variety of tree-based algorithms on our competition data and show their performance. Compare:

   - Small tree without pruning
   - Large tree without pruning
   - Large tree after pruning with 1-SE rule
   - Bagging/RF on small trees (100 iterations)
   - Bagging/RF large trees (100 iterations)

   Do this under five-fold cross validation on our competition training set, and use the results of the five different folds to calculate confidence intervals for performance. Plot all the results in a reasonable way (e.g. using `boxplot()`) and comment on them. Explain your choices of "small" and "large".
   Hints: a. Start early since bagging may take a while to run. b. Use as a basis the code from class which implements much of this.

2. **AdaBoost and $\epsilon$-Adaboost**
   This problem refers to the AdaBoost algorithm (Freund and Schapire, 1997), which is used for binary classification with labels $y_i \in \{\pm 1\}$. Adaboost initializes $\hat{f}_0 = 0, w_i \equiv 1$, then for $t = 1...T$ updates:

   (a) Fit a classification tree with response $y$ and weights $w$ on the observations, getting tree $h_t$
   (b) Denote by $Err_t$ the (weighted) misclassification error of $h_t$
   (c) Set $\alpha_t = 0.5 \log\left((1 - Err_t)/Err_t\right)$
   (d) Update weights: $w_i \leftarrow w_i \exp\left(-\alpha_t (y_i h_t(x_i))\right)$

   The model after step $t$ is $f_t(x) = \sum_{u=1}^{t} \alpha_u h_u(x)$, the final model is $f_T$, and classification if according to the sign of $f_T(x)$.
   As we said in class, from the coordinate descent perspective of boosting, AdaBoost can be viewed as gradient boosting with loss function $L(y, \hat{y}) = \exp(-y\hat{y})$ and line-search steps, explicitly:

   - Fitting a classification tree is minimizing $\sum_i w_i y_i h_k(x_i) = \langle wy, h_k(X) \rangle$ (so $w_i y_i$ is the actual gradient)
   - The calculated $\alpha_t$ is the solution to the line search problem: $\alpha_t = \arg\min_\alpha \sum_i L\left(y_i, (f_{T-1} + \alpha h_t)(x_i)\right)$
   - The updated $w_i$ is indeed (proportional to) the absolute value of the gradient:

   $$w_i \propto \left| \frac{dL\left(y_i, l\right)}{dl}\Big|_{l=\hat{f}_t(x_i)} \right|$$

   (a) Choose one of the three properties above and prove explicitly that it holds (for example, if you choose the first one, show how fitting $h_k$ classification tree minimizing weighted miclassification error is equivalent to choosing a coordinate descent direction in the exponential loss function).

(b) The code in www.tau.ac.il/∼saharon/StatsLearn2022/AdaBoost.r implements AdaBoost on the competition data for the problem of whether $y > 3$ or $y \leq 3$. Read the code carefully to make sure you understand the details. Note especially the parameters "method" and "weights" that rpart takes.

i. With 8000-2000 training-test division as in the code, run the algorithm for 1000 iterations and draw a plot of training and test misclassification as a function of iterations. Explain its form.

ii. Change the algorithm from line-search boosting to $\epsilon$-boosting with $\epsilon = 0.01$, not changing the other parameters. Run this version for 1000 iterations and draw the same plot. Discuss the results and compare to the line-search version.

iii. Now change the loss function from the exponential loss to squared error loss, and the approach to the regular gradient boosting with regression tree. Explain briefly in writing what you did, and run with the same parameters ($\epsilon = 0.01, T = 1000$). Classify according to sign of $\hat{y}$ and repeat the same analysis again. Discuss the relative results.

iv. (* +5) Play with the parameters in one (or all) of the algorithms (tree depth or other stopping criteria, $\epsilon$, $T$) to change the results. Show how you can guarantee much better training results. Can you find settings that give much better testing results?

3. **ESL 7.6 (7.5 in first edition): Degrees of freedom of Nearest Neighbors**
Prove that in the standard i.i.d error model (which the book calls "additive error"), the effective degrees of freedom of $k$-NN with $N$ observations is $N/k$.

4. **Neural networks:**

(a) Assume we are given a modeling problem with $x \in \mathbb{R}^p$ and $y \in \{0, 1\}$, which are can treat as a regression or classification problem (but prediction is always by comparing predictions to 0.5 and predicting either 0 or 1). For the following popular models, describe a neural network that implements them:

- Standard linear regression
- Logistic regression

Explain in what sense the network implements them. Specifically, do we expect to get the same fitted model from the network as from the regular model when applied to data? Why yes or why not?

(b) The code nn.r reads the South African heart dataset, divides it into training and test sets, and uses Keras to apply a NN with one hidden neuron and logit (=sigmoid) activation. It also applies and tests logistic regression. Use this skeleton to:

i. Implement all four models described in the previous part

ii. Prepare 2*2 confusion tables of predicted vs. actual labels

iii. Briefly discuss the results compared to your expectations from the previous section

(c) Implement a more complex architectures (e.g., a hidden layer with three nodes, and then an output layer, see commented code in the file) and apply it to the data. You may play with some of the parameters if necessary. Discuss its test-set performance.

(d) Implement a network with a hidden layer with three nodes and an output layer, where all activations are linear. What form does the final model have? What functions of the original $x$ variables are being fitted?

Resources for this problem:
Keras help
Keras in R

**Note:** You are of course welcome to solve and submit this problem in Python or any other environment, all Keras/Tensorflow models in the example R code should be easily transportable.