CS 461: Machine Learning Principles
Fall 2021

Problem Set #4 (80 Points)

Out: Tuesday, November 9

Due: Tuesday, November 23, 8:00pm

Instructions

How and what to submit? Please submit your solutions electronically via Canvas. Please submit two files:

1. A PDF file with the written component of your solution including derivations, explanations, etc. You should create this PDF in LaTeX. Please name this document ⟨`firstname-lastname`⟩`-sol4.pdf`.

2. The empirical component of the solution (Python code and the documentation of the experiments you are asked to run, including figures) in a Jupyter notebook file. Rename the notebook ⟨`firstname-lastname`⟩`-sol4.ipynb`.

Late submissions: there will be a penalty of 20 points for any solution submitted within 24 hours past the deadline. No submissions will be accepted past then.

What is the required level of detail? See below.

(a) When asked to **derive** something, please clearly state the assumptions, if any, and strive for balance: justify any non-obvious steps, but try to avoid superfluous explanations.

(b) When asked to **plot** something, please include in the `ipynb` file the figure as well as the code used to plot it. If multiple entities appear on a plot, make sure that they are clearly distinguishable (by color or style of lines and markers) and references in a legend or in a caption.

(c) When asked to **provide a brief explanation or description**, try to make your answers concise, but do not omit anything you believe is important. If there is a mathematical answer, provide it precisely (and accompany by succinct wording, if appropriate).

(d) When **submitting code (in Jupyter notebook)**, please make sure it's reasonably well-documented, runs, and produces all the requested results. If **discussion** is required/warranted, you should include it directly in the notebook (using the markdown cell).

Collaboration policy: collaboration is allowed and encouraged, as long as you (1) write your own solution entirely on your own, and (2) specify names of student(s) you collaborated with in your writeup.

1 Boosting

Given training examples $(x_1, y_1) \dots (x_N, y_N) \in \mathcal{X} \times \{\pm 1\}$, a hypothesis class $\mathcal{H}$ of binary classifiers (for which learning is tractable), and the number of boosting rounds T, AdaBoost consecutively finds classifiers $h_1 \dots h_T \in \mathcal{H}$ and weights $\alpha_1 \dots \alpha_T \geq 0$ such that their ensemble $g_T : \mathcal{X} \to \mathbb{R}$

$$g_T(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \qquad \forall x \in \mathcal{X}$$

is a strong classifier. Note that the output space of each h_t is $\{\pm 1\}$ whereas the output space of g_T is $\mathbb{R}$, so the actual prediction (i.e., binary label) is made by $\mathbf{sign}\,(g_T(x))$. A central concept in AdaBoost is a round-specific distribution D_t over training examples. We define D_0 to be uniform (i.e., $D_0(i) = 1/N$ for all i) and then compute, for all $t = 1 \dots T$,

$$D_t(i) = \frac{D_{t-1}(i) \exp(-\alpha_t y_i h_t(x_i))}{\sum_{j=1}^{N} D_{t-1}(j) \exp(-\alpha_t y_j h_t(x_j))} \qquad \forall i \in \{1 \dots N\}$$

In the lecture, we derived this form from a softmax over negative margins of the ensemble. We cannot compute D_t from D_{t-1} without choosing h_t and α_t. AdaBoost dictates

$$h_t = \arg\min_{h \in \mathcal{H}} \hat{\epsilon}_{D_{t-1}}(h) \tag{1}$$

$$\alpha_t = \frac{1}{2} \log \left(\frac{1 - \hat{\epsilon}_{D_{t-1}}(h_t)}{\hat{\epsilon}_{D_{t-1}}(h_t)} \right) \tag{2}$$

where $\hat{\epsilon}_{D_{t-1}}(h_t)$ is the empirical error probability of h_t *weighted by* D_{t-1}, that is

$$\hat{\epsilon}_{D_{t-1}}(h_t) = \sum_{i=1}^{N} D_{t-1}(i) \mathbb{1}(h_t(x_i) \neq y_i)$$

($\mathbb{1}(A) \in \{0, 1\}$ is the indicator function).

Problem 1 **[10 points]**

Show that α_t results in $\hat{\epsilon}_{D_t}(h_t) = 1/2$. That is, AdaBoost chooses α_t so that the *updated* data distribution yields the worst possible weighted loss for h_t (weighted by D_t, not D_{t-1}!). More specifically, (1) express $\hat{\epsilon}_{D_t}(h_t)$ as a function of α_t and $\hat{\epsilon}_{D_{t-1}}(h_t)$, then (2) set it to $1/2$ and solve for α_t to obtain the expression (2). Justify your derivations precisely: do not just say $A = B$ if the equality is not obvious, explain what fact you're using.

End of problem 1

Problem 2 **[10 points]**

In light of the previous problem, is it possible that $h_{t+1} = h_t$ for some t? That is, could we select the same classifier again in the *immediately* following round? Explain why or why not. For this problem, you may assume that the base classifiers in $\mathcal{H}$ are neither too weak nor too strong, so that for all data distribution D that is full support (i.e., $D(i) > 0$ for all i) there is always some $h \in \mathcal{H}$ such that $\hat{\epsilon}_D(h) < 1/2$ but never $h' \in \mathcal{H}$ such that $\hat{\epsilon}_D(h') = 0$.

End of problem 2

Problem 3 **[10 points]**

Is it possible that $h_{t+n} = h_t$ for some t and $n > 1$? Explain why or why not.

End of problem 3

Problem 4 **[10 points]**

Recall that given any $f : \mathcal{X} \to \mathbb{R}$ used for binary classification as $\hat{y} = \textbf{sign}\,(f(x))$, its margin $yf(x) \in \mathbb{R}$ on a labeled example $(x, y) \in \mathcal{X} \times \{\pm 1\}$ measures the degree of correctness (if positive) or incorrectness (if negative) on that example. In the lecture, we discussed how the **exponential loss** $\exp(-\alpha yf(x))$ is an upper bound on the zero-one loss $\mathbb{1}(yf(x) < 0)$ as a function of the margin, for any value of $\alpha \geq 0$ (which is evident in a plot). At the t-th round of boosting, the weighted empirical exponential loss of h_t is

$$\hat{l}_{D_{t-1}}(h_t) = \sum_{i=1}^{N} D_{t-1}(i) \exp(-\alpha_t y_i h_t(x_i))$$

Assuming that h_t is selected, show that the choice of α_t in (2) minimizes $\hat{l}_{D_{t-1}}(h_t)$.

End of problem 4

2 Spam Detection with Decision Trees

Task In this section we will apply decision trees to the problem of determining whether or not an email is spam.

Dataset The dataset has been split into a training set of 3000 examples, along with validation and test sets. Each example has 57 continuous valued features, mostly indicating the frequencies of words such as "money", "free", and "credit". Other features include the length of the longest run of capital letters, and some character frequencies (e.g., "$" and "!"). The labels are binary, and the annotation is slightly noisy (there are two pairs of training examples with exactly the same features, but different labels).

Problem 5 **[10 points]**
Fill in missing pieces of the decision tree code, specifically in `gini_impurity`, `fit_stump` (pass the unit test), and the `fit` method in the `DecisionTree` class.

End of problem 5

Problem 6 **[5 points]**
Train decision trees on the spam dataset with various hyperparameter settings. We have provided a grid search over some reasonable parameter settings, but you are welcome to explore more. Once you are satisfied with your validation accuracy, report the hyperparameter setting and the training/validation accuracy of your best model in the markdown cell.

End of problem 6

Problem 7 **[20 points]**
Fill in missing pieces of the AdaBoost code, specifically in `adaboost`. You can expect very good performance if your implementation is correct.

End of problem 7

Problem 8 **[5 points]**
Train an ensemble on the spam dataset with various hyperparameter settings. Feel free to improve your performance by modifying the hyperparameters. Once you are satisfied with your validation accuracy, report the hyperparameter setting and the training/validation accuracy of your best model in the markdown cell. Also, report the test accuracy from the public Kaggle leaderboard (see below).

End of problem 8

Kaggle submission We have set up a Kaggle competition to which you will be submitting your final predictions on the test set (link). First, you will have to create a Kaggle account (with your email: `scarletmail.rutgers.edu`). Once you have access to the competition page, read through all three information pages carefully (Description, Evaluation, and Rules) and accept the rules. You will now be ready to make submissions. The notebook creates a file `cs461hw4_mynetid.csv` in your Google Drive where `mynetid` is your NetID. Once you have accepted the Kaggle rules, there will be an option to "Make a submission", where you can upload this CSV file. To make sure you do not overfit this held-out set, we have limited your submissions to two per day, so start early and you will get more chances if you make mistakes. Your up to date score will appear on the public leaderboard once you submit.