CSE 512: Homework 5 — Due Nov. 9

1. **Bias-variance tradeoff.** Suppose I want to identify the location of a star, which lives at coordinates defined by $x \in \mathbb{R}$. Every day I go to the telescope, and I receive a new measurement $y_i = x + z_i$, where $z_i \sim \mathcal{N}(0,1)$ is the noise in each measurement.

   After m days, I receive m measurements, $y_1, ..., y_m \in \mathbb{R}$.

   (a) Denote x_{MLE} as the maximum likelihood estimate of x. Compute x_{MLE} in terms of y_i and m. Compute also the bias and variance of x_{MLE}. Describe the behavior of the bias and variance as $m \to +\infty$.

   (b) A colleague walks in the room and scoffs at my experiment. "I already know where this star is!" the colleague exclaims, and gives me a new set of measurements $\bar{x} \in \mathbb{R}$. "You can just cancel your experiment now!" Trouble is, I know the colleague is full of hot air, so while this is valuable information, I'm not willing to take it without any verification. Instead, I estimate x by solving a linear regression problem

   $$\underset{x}{\text{minimize}} \quad \frac{1}{m}\sum_{i=1}^{m}(y_i - x)^2 + \frac{\rho}{2}(x - \bar{x})^2$$

   for some $\rho > 0$. Denote x_{MAP} as the solution to this linear regression problem. Write out x_{MAP} in terms of y_i, $\bar{x}$, ρ, and m. Compute also the bias and variance of x_{MAP}, its behavior as $m \to +\infty$, and how it compares to the bias and variance of the MLE estimator.

   (c) Show that for any estimator $\hat{x}$, the expected squared error $\mathbb{E}[(x - \hat{x})^2] = B^2 + V$ where $B = x - \mathbb{E}[\hat{x}]$ the estimator bias and $V = \mathbb{E}[(\mathbb{E}[\hat{x}] - \hat{x})^2]$ the estimator variance.

   (d) In terms of constants $\Delta = x - \bar{x}$ and m, what value of ρ minimizes the expected squared error of the estimator x_{MAP}?
   Hint: if a function is not convex in α, it may be convex in β where $\beta = f(\alpha)$ is a 1-1 function of α.

2. **Cross validation**

   - Download the dorothea dataset, and the corresponding ipython notebook (cross_validation_release.ipynb). Take a look at the dorothea pdf to understand the dataset and the task.
   - The Dorothea dataset is interesting and different from our previous datasets for 2 reasons. First, after loading the python notebook, run the cell that loads the data and take a look at some of the data qualities: namely, the number of features, labels, and sparsity of X. Additionally, write a script that returns some information on the balance of the training set labels. Write down some observations as to why this task might be different than previous binary classification tasks we've done in this class.
   - In the python notebook, I have given you a short implementation of a decision tree classifier using `scikit-learn`, with specifications

     `criterion='entropy',splitter='best',max_depth=depth, class_weight='balanced'`.

     You are free to use this implementation, or whatever other implementation you'd like, as long as you preserve these parameters. (Keep the depth a variable, as we will change this throughout the exercise.)

     Run this classifier over the training set, and evaluate the performance using *misclassification rate* over both the test and train set. What do you observe? Did you do a good job?
   - An alternative performance metric when dealing with unbalanced dataset is the F1 score, which can be written as

     $$F1 = \frac{2PR}{P+R}, \qquad P = \frac{\#\text{ detected}}{\#\text{ retrieved}}, \qquad R = \frac{\#\text{ detected}}{\#\text{ relevant}}.$$

     For our data set, the event that $y_i = 1$ is a rare event, and we want to measure our ability to detect this rare event. So, an event is detected if $y_i = 1$ and $\hat{y}_i = 1$, retrieved if $\hat{y}_i = 1$, and relevant if $y_i = 1$.

     Report the F1 score of the test and train set using the previously trained classifier. Does the F1 score look like a more reasonable performance metric for this task?

- *Without doing any cross validation*, sweep the depth of the tree through the values 2,3,...,10 and plot the train and test F1 scores for this sweep. Comment on what you see.
- Now, implement a K-fold cross validation, for $K = 5$. Take the train data, and partition it to K segments. Over $i = 1, ..., K$ trials, take away the ith partition as a validation set, and record the F1 score over the sweep of tree depths over the remaining train set. Plot the F1 score over the *validation set* for all K trials. Comment on what you see. Is there a stable trend?
- Finally, plot the average validation F1 score against the test F1 score for the tree depth sweep. Does the best average F1 score of the validation set correspond, more or less, to a good F1 score over the test set? Was this a successful venture?

3. **Adaboost** A popular and computationally cheap boosting method is adaboost, described in Algorithm 1. In particular, it is a greedy coordinate-wise method that minimizes the empirical exponential loss, e.g. given a predictor $h(x) = y$, we find h which minimizes

$$f(h) = \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i h(x_i)).$$

In this problem, we will implement Adaboost and analyze its greedy structure.

Algorithm 1: Discrete Adaboost (source: https://en.wikipedia.org/wiki/AdaBoost)

Data: Samples: $x_1, ..., x_m$, training labels $y_i \in \{-1, 1\}$, weak learners $\mathcal{H}$.
Result: Classifier $H(x) = \sum_{t=1}^{T} \alpha_t h^{(t)}(x)$
Initial weights $w_i^{(0)} = \frac{1}{m}$ for $i = 1, ..., m$;
for $t = 1, ..., T$ **do**

Choose $h^{(t)}(x)$ which minimizes the weighted sum error for misclassified points

$$h^{(t)}(x) = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_{\substack{i=1 \\ h(x_i) \neq y_i}}^{m} w_i^{(t-1)}$$

Update

$$\alpha^{(t)} = \frac{1}{2} \log \left(\frac{1 - \epsilon^{(t)}}{\epsilon^{(t)}} \right), \qquad \epsilon^{(t)} := \sum_{h^{(t)}(x_i) \neq y_i} w_i^{(t-1)}$$

Update weights

$$\hat{w}_i^{(t)} = w_i^{(t-1)} \exp(-y_i \alpha^{(t)} h^{(t)}(x_i)), \qquad w_{i,t+1} = \frac{\hat{w}_i^{(t+1)}}{\sum_j \hat{w}_j^{(t+1)}}$$

end

(a) **Greedy behavior.** Show that the update for α indeed minimizes the empirical risk over the already-trained classifiers, using the exponential loss function. That is, at some time t, show that

$$\alpha^{(t)} = \underset{\alpha^{(t)}}{\operatorname{argmin}} \sum_{i=1}^{m} \mathcal{L}\left(H^{(t)}(x_i); y_i\right), \quad \mathcal{L}(y; \hat{y}) = \exp(-y\hat{y})$$

where $H^{(t)}(x) = \sum_{t'=1}^{t} \alpha^{(t')} h^{(t')}(x)$ the current aggregated predictor. Note that y, $\hat{y}$, and $h^{(t)}(x)$ all take binary values in $\{-1, 1\}$.

Hint. You do not need to follow this way, but here are some hints to get you started.

- Start by writing the loss function as a function of $\alpha = \alpha^{(t)}$.
- Show that this function is convex in α by taking its second derivative, and arguing that it is nonnegative everywhere.

- When setting the gradient to 0, it will be frustrating because it will look something like

$$f'(\alpha) = \sum_{i=1}^{m} c_i e^{\alpha z_i}.$$

Note, however, that in that scenario, $z_i \in \{-1, 1\}$, and you can separate the sum to deal with both scenarios separately. That is,

$$f'(\alpha) = \sum_{z_i > 0} c_i e^{\alpha} + \sum_{z_i < 0} c_i e^{-\alpha} = 0.$$

Think about how to solve for α here.

While I hope the hint helps you, when presenting the solution, do so as if there were no hints here. (Full, standalone justifications.)

(b) **Coding.** Open the mnist_adaboost_release directory and in the iPython notebook, download the data. We are again going to do 4/9 handwriting disambiguation. Only minimal preprocessing was used in this dataset, since for decision trees, normalization is less of an issue.

- Borrowing code from the previous exercise, implement a decision stump (tree with depth = 1). Initialize weights as $w_i = 1/m$ for all $i = 1, ..., m$, and fit the decision tree over the *weighted* misclassification error, using the code snippet

```
clf = clf.fit(Xtrain, ytrain, sample_weight = w)
```

Report the train and test misclassification rates using just this decision stump. Report also the train exponential loss value.

- Now implement the Adaboost method, as shown in algorithm 1. Plot the training exponential loss, and train and test misclassification rate. How would you say the performance compares to previous versions of this task (e.g. using logistic regression, 1-NN, etc)?
Plot also $\epsilon^{(t)}$ and $\alpha^{(t)}$ as a function of t. For what values of $\epsilon^{(t)}$ is $\alpha^{(t)}$ really large and positive? really large and negative? close to 0? Interpret this mechanism; what is it saying about how boosting uses classifiers, in terms of their weighted performance?

Challenge!

1. Let's revisit the messy sock problem. We are using this problem to arrive at a well-known result in information theory. Therefore, although you are free to google around, submit here full justifications for each answer for full credit. (You cannot use the result to justify the result.)

   (a) Suppose that my mom gives me some money to buy 10 socks, and I decide only to buy red and black socks. (Stonybrook colors!) Define the random variable X as the color of the sock I randomly pull out of my shopping bag. How many socks of each color should I buy to maximize the entropy of X?

   As a reminder, the entropy of a random variable which can take 2 values is

   $$H(X) = -\mathbf{Pr}(X = \text{red}) \log_2(\mathbf{Pr}(X = \text{red})) - \mathbf{Pr}(X = \text{black}) \log_2 \mathbf{Pr}(X = \text{black}).$$

   (b) Show more generally that, for any $0 \leq c \leq 1$, the constrained optimization problem

   $$\begin{array}{ll} \underset{p}{\text{maximize}} & f(p) := -p \log_2(p) - (c-p) \log_2(c-p) \\ \text{subject to} & 0 \leq p \leq c \end{array}$$

   reaches its optimum value uniquely at $p = c/2$.

   (c) Now for my birthday my mom takes me to Mall of America, and I have access to socks of 100 different colors. I have enough money to buy 10,000 socks. Again, define X as the random variable taking the value of the color of sock I randomly pull out of my final purchase of 10,000 socks. What color socks should I buy in order to maximize the entropy of X? Use the above pieces to rigorously prove your answer.

2. **Correlated mixture of sequential experts.**

   I want to buy a yacht, but I'm not sure if it's a good idea given the economy. So, I decide to question m consultants. Each consultant has more-or-less the same qualifications, and they come in one at a time.

   The first consultant comes in my office. I ask, "Should I buy a yacht?" She says yes with probability p.

   On her way out the building, she meets the second consultant. They chat briefly, and she leaves, he comes, and the process repeats. Each time, I ask the consultant if I should buy a yacht, and receive "yes" with probability p; each time, the consultant chats briefly with the next consultant. *However*, the answers now are *not* i.i.d., but rather each expert's answer is correlated with the answer of experts he/she chatted with in the lobby.

   At the end of the day, I have met with m consultants. I will make a decision whether to buy a yacht based on majority rule. We will now calculate the probability that I will buy a yacht.

   (a) We model the answer of each consultant as $Y_i = 1$ if the ith consultant recommended "yes", and $Y_i = -1$ otherwise. Show that if distribution

   $$\mathbf{Pr}(Y_1 = 1) = p, \qquad \mathbf{Pr}(Y_i = 1 | Y_{i-1} = 1) = c + p - cp, \qquad \mathbf{Pr}(Y_i = -1 | Y_{i-1} = -1) = cp - p + 1$$

   then $\mathbf{Pr}(Y_i = 1) = p$ for all i.

   (b) The Pearson correlation coefficient between a random variable U and V can be expressed as

   $$\mathbf{corr}(U, V) = \frac{\mathbb{E}[UV] - \mathbb{E}[U]\mathbb{E}[V]}{\sqrt{\mathbf{var}(U)\mathbf{var}(V)}}.$$

   Show that the correlations between each pair of sequential experts $\mathbf{corr}(Y_i, Y_{i-1}) = c$,

   Hint: Go ahead and use a symbolic calculator, like WolframAlpha, to simplify messy expressions.

   (c) For $m = 3$, what is the probability that I will buy a yacht, in terms of $P_{11} = \mathbf{Pr}(Y_i = 1 | Y_{i-1} = 1)$,$P_{00} = \mathbf{Pr}(Y_i = -1 | Y_{i-1} = -1)$ and p?

   (d) **Code.** Compute exactly the probability that I will buy a yacht, for $m = 10$, in Python or MATLAB. Generate a plot that shows the probability that I will buy a yacht, sweeping $c \in [-1, 1]$. (Note that there are cases where p and c are an infeasible pair, and can be detected when probabilities are not in the range (0,1)–these cases should not be plotted.) Do this for several interesting values of p. Comment on how the decision changes as a function of c, p, and m.

(e) Simulate the sequential advisers, and give a numerical estimate of what my decision will be if $m = 25$ and $m = 100$. Generate a similar plot, using these numerical estimates of $\mathbf{Pr}$(I will buy a yacht). Use your own discretion to decide how many trials you need, and what values of c and p are useful.