CSE 512: Homework 4 Due Oct. 19

1. **Jensen's inequality. (1 pts)** Jensen's inequality is an application of the convexity law to expectations. Recall that the expectation of a discrete random variable $f(X)$ with pmf $p_X(x)$ is written as

$$\mathbb{E}[f(X)] = \sum_{i=1}^{m} p_i f(x_i)$$

if the number of values X can take is finite, e.g. in $\{x_1, ..., x_m\}$ and $p_i = \mathbf{Pr}(X = x_i)$.

Show that if f is a convex function, then

$$\mathbb{E}[f(X)] > f(\mathbb{E}[X]).$$

Ans. The question is essentially asking

$$\mathbb{E}[f(X)] = \sum_{i=1}^{m} p_i f(x_i) \overset{?}{\geq} f\left(\sum_{i=1}^{m} x_i p_i\right) = f(\mathbb{E}[X])$$

We can now use recursion to get what we want, by splitting the terms one by one:

$$f(\mathbb{E}[X]) = f\left(x_1 p_1 + \sum_{i=2}^{m} x_i p_i\right) \overset{\text{convexity}}{\leq} p_1 f(x_1 p_1) + (1-p_1) f\left(\sum_{i=2}^{m} \frac{x_i p_i}{1-p_1}\right).$$

Next,

$$f\left(\frac{x_2 p_2}{1-p_1} + \sum_{i=3}^{m} \frac{x_i p_i}{1-p_1}\right) \leq \frac{p_2}{1-p_1} f(x_2) + \left(\frac{1-p_1-p_2}{1-p_1}\right) f\left(\sum_{i=3}^{m} \frac{x_i p_i}{1-p_1-p_2}\right)$$

which simplifies to

$$f\left(x_1 p_1 + x_2 p_2 + \sum_{i=3}^{m} x_i p_i\right) \leq p_1 f(x_1 p_1) + p_2 f(x_2) + (1-p_1-p_2) f\left(\sum_{i=3}^{m} \frac{x_i p_i}{1-p_1-p_2}\right)$$

I now see a pattern, and at the mth step, will get

$$f\left(\sum_{i=1}^{m} x_i p_i\right) \leq p_1 f(x_1 p_1) + p_2 f(x_2) + \cdots + (1-p_1-\cdots-p_{m-1}) f(\frac{x_m p_m}{1-p_1-\cdots-p_{m-1}})$$

Since $1 - p_1 - \cdots - p_{m-1} = p_m$, the right hand side reduces to $\mathbb{E}[f(X)]$, and the statement is proven.

2. **Entropy, conditional entropy, mutual information, information gain**

I have a very messy sock drawer, containing 10 red socks, 5 blue socks, 4 yellow socks, and 1 black sock.

(a) **(0.5 pts)** Recall the formula for entropy:

$$H(X) = \sum_{X=x} \mathbf{Pr}(X = x) \log_2(\mathbf{Pr}(X = x)).$$

Define X a random variable which represents the color of a sock, randomly (uniformly) picked. What is the entropy of this sock?

Ans.

$$\begin{aligned} H(X) &= -\mathbf{Pr}(\text{red}) \log_2 \mathbf{Pr}(\text{red}) - \mathbf{Pr}(\text{blue}) \log_2 \mathbf{Pr}(\text{blue}) - \mathbf{Pr}(\text{yellow}) \log_2 \mathbf{Pr}(\text{yellow}) - \mathbf{Pr}(\text{black}) \log_2 \mathbf{Pr}(\text{black}) \\ &= -\tfrac{1}{2} \log_2 \tfrac{1}{2} - \tfrac{1}{4} \log_2 \tfrac{1}{4} - \tfrac{1}{5} \log_2 \tfrac{1}{5} - \tfrac{1}{20} \log_2 \tfrac{1}{20} \\ &\approx -1.68 \end{aligned}$$

(b) **(0.5 pts)** My mom comes and tells me I must organize my socks better. So, I put all my red socks in the top drawer and the rest in my bottom drawer. Recall the formula for conditional entropy:

$$H(X|Y) = \sum_{X=x,Y=y} \mathbf{Pr}(X = x, Y = y) \log_2(\mathbf{Pr}(X = x|Y = y)).$$

What is the conditional entropy, where X is the color of a sock randomly picked, and Y is the drawer of which I pick it from? Assume that I pick the top drawer with twice the probability as picking the bottom drawer, but given a drawer, my choice of sock is uniformly distributed.

Ans.

$$\begin{aligned}
H(X|Y) &= -\mathbf{Pr}(\text{red}, \text{top}) \log_2 \mathbf{Pr}(\text{red}|\text{top}) - \mathbf{Pr}(\text{blue}, \text{bottom}) \log_2 \mathbf{Pr}(\text{blue}|\text{bottom}) \\
&\quad -\mathbf{Pr}(\text{yellow}, \text{bottom}) \log_2 \mathbf{Pr}(\text{yellow}|\text{bottom}) - \mathbf{Pr}(\text{black}, \text{bottom}) \log_2 \mathbf{Pr}(\text{black}|\text{bottom}) \\
&= -\mathbf{Pr}(\text{red}|\text{top})\mathbf{Pr}(\text{top}) \log_2 \mathbf{Pr}(\text{red}|\text{top}) \\
&\quad -\mathbf{Pr}(\text{blue}|\text{bottom})\mathbf{Pr}(\text{bottom}) \log_2 \mathbf{Pr}(\text{blue}|\text{bottom}) \\
&\quad -\mathbf{Pr}(\text{yellow}|\text{bottom})\mathbf{Pr}(\text{bottom}) \log_2 \mathbf{Pr}(\text{yellow}|\text{bottom}) \\
&\quad -\mathbf{Pr}(\text{black}|\text{bottom})\mathbf{Pr}(\text{bottom}) \log_2 \mathbf{Pr}(\text{black}|\text{bottom}) \\
&= -1 \cdot \tfrac{2}{3} \log_2(1) - \tfrac{1}{2} \cdot \tfrac{1}{3} \log_2 \tfrac{1}{2} - \tfrac{2}{5} \cdot \tfrac{1}{3} \log_2 \tfrac{2}{5} - \tfrac{1}{10} \cdot \tfrac{1}{3} \log_2 \tfrac{1}{10} \\
&\approx -0.4537
\end{aligned}$$

(c) **(0.5 pts)** The *information gain* (also called *mutual information*) can be defined in terms of the entropy and conditional entropy

$$I(X; Y) = H(X) - H(X|Y).$$

Give the mutual information between X the color of the sock and Y the drawer which it comes from.

Ans.

$$I(X; Y) = H(X) - H(X|Y) \approx -(1.68 - 0.454) = 1.23$$

3. **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.

   i. **(0.3 pts)** Compute x_{MLE} in terms of y_i and m.
   Ans. Since each

   $$y_i \sim x + \mathcal{N}(0, 1)$$

   then the likelihood of this estimation is

   $$\mathbf{Pr}(y_1, ..., y_m|x) \propto \prod_{i=1}^{m} \exp(y_i - x)$$

   and is maximized with $\hat{x} = \frac{1}{m} \sum_{i=1}^{m} y_i$, e.g. the sample mean.

   ii. **(0.4 pts)** Compute the bias and variance of x_{MLE}.
   Ans. We know that $\hat{x}$ is an unbiased estimator of x; namely,

   $$\mathbb{E}[\hat{x}] = \mathbb{E}\left[\frac{1}{m} \sum_{i=1}^{m} y_i\right] = \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}[y_i] = x$$

   and so $\mathbf{Bias}(\hat{x}) = \mathbb{E}[\hat{x}] - x = 0$.

The variance of this estimator is

$$\begin{aligned}
\mathbf{Var}(\hat{x}) &= \mathbb{E}[(\hat{x} - \mathbb{E}[\hat{x}])^2] \\
&= \mathbb{E}\left[\left(\frac{1}{m}\sum_{i=1}^{m} y_i - \mathbb{E}[\hat{x}]\right)^2\right] \\
&\overset{\mathbb{E}[\hat{x}]=x}{=} \mathbb{E}\left[\left(\frac{1}{m}\sum_{i=1}^{m} (y_i - x)\right)^2\right] \\
&= \frac{1}{m^2}\mathbb{E}\left[\left(\sum_{i=1}^{m} \underbrace{(y_i - x)}_{z_i}\right)^2\right] \\
&\overset{\text{linearity of variance}}{=} \frac{1}{m^2}\sum_{i=1}^{m} \mathbf{var}(z_i) = \frac{1}{m}.
\end{aligned}$$

iii. **(0.3 pts)** Describe the behavior of the bias and variance of x_{MLE} as $m \to +\infty$.
Ans. The bias is 0 for all m, and the variance $1/m \overset{m\to\infty}{\to} 0$.

(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 $\bar{x} \in \mathbb{R}^n$. "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

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

for some $\rho > 0$. Denote x_{MAP} as the solution to this linear regression problem. You should treat $\bar{x}$ as an external constant, which is not random, but may not be equal to x.

i. **(0.3 pts)** Compute x_{MAP} in terms of y_i, m, $\bar{x}$, and ρ.
Ans. By setting the gradient of the objective to 0, I see that this estimator can be written explicitly as

$$x_{\text{MAP}} = \frac{1}{1+\rho}\left(\frac{1}{m}\sum_{i=1}^{m} y_i + \rho\bar{x}\right)$$

ii. **(0.4 pts)** Compute the bias and variance of x_{MAP}.
Ans. For positive ρ, the bias can be computed as

$$\mathbb{E}[x_{\text{MAP}}] = \mathbb{E}\left[\frac{1}{1+\rho}\left(\frac{1}{m}\sum_{i=1}^{m} y_i + \rho\bar{x}\right)\right] = \frac{1}{1+\rho}\left(\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}[y_i] + \rho\bar{x}\right) = \frac{1}{1+\rho}(x + \rho\bar{x})$$

and thus $\mathbf{Bias}(x_{\text{MAP}}) = \frac{1}{1+\rho}(x + \rho\bar{x}) - x$, and x_{MAP} is biased if $x \neq \bar{x}$.
The variance can be computed as

$$\begin{aligned}
\mathbf{var}(x_{\text{MAP}}) &= \frac{1}{(1+\rho)^2}\mathbf{var}\left(\frac{1}{m^2}\sum_{i=1}^{m} y_i + \rho\bar{x}\right) \\
&\overset{\text{linearity of variance}}{=} \frac{1}{(1+\rho)^2}\underbrace{\mathbf{var}\left(\frac{1}{m^2}\sum_{i=1}^{m} y_i\right)}_{1/m} + \rho\underbrace{\mathbf{var}(\bar{x})}_{=0} \\
&= \frac{1}{(1+\rho)^2 m}.
\end{aligned}$$

iii. **(0.3 pts)** Describe the behavior of the bias and variance of x_{MAP} as $m \to +\infty$.
Ans. The problem is biased for all m, and does not really increase or decrease. The variance of the MAP estimator is smaller than that of the MLE estimator. However, it also goes to 0 as $O(1/m)$, the same rate as the MLE estimator.

(c) **(0.5 pts)** A natural question to ask is, how to choose ρ? In general, we want ρ to be big when our MLE estimate is not very powerful, either because m is very small or the variance of y_i is very big. On the other hand, if our prior $\bar{x}$ is very close to x, large ρ can also help guide our guess. But while we can't in general know $\|x - \bar{x}\|$, we can try to make some statistical arguments over how ρ should depend on m.
Recall from lecture that

$$\text{MSE} = \mathbb{E}[(x - \hat{x})^2] = B^2 + V.$$

Show that by introducing a dummy variable $\beta = \frac{1}{1+\rho}$, that the MSE can be written as a convex function of β. Use this to find the ρ that minimizes the MSE, as a function of m and an error term $\Delta = \bar{x} - x$.
Ans. From the previous parts, we have that

$$\mathbb{E}[(x - \hat{x})^2] \quad = \quad B^2 + V = \left(\frac{x + \rho\bar{x}}{1+\rho} - x\right)^2 + \frac{1}{m(1+\rho)^2} = ((1-\beta)\Delta)^2 + \frac{\beta^2}{m} =: g(\beta).$$

Taking the second derivative of g,

$$g'(\beta) = -2((1-\beta)\Delta^2) + 2\frac{\beta}{m}, \qquad g''(\beta) = 2\Delta^2 + \frac{2}{m} > 0, \quad \forall\beta$$

and therefore g is convex in β. It is minimized by setting $g'(\beta) = 0$, giving

$$(1-\beta)\Delta^2 = \frac{\beta}{m} \iff \beta = \frac{\Delta^2}{1/m + \Delta^2} = \frac{1}{1/(m\Delta^2) + 1} \iff \rho = \frac{1}{m\Delta^2}.$$

While we do not know Δ^2 in practice, we do know m, and we know that whatever parameter we pick for ρ should decay like $O(1/m)$.

4. **Decision trees (coding). (3 pts)** In real life, you would use one of many highly optimized packages to program decision trees, but for the sake of understanding, here we will build a tiny decision tree on a simplified version of a multiclass classification problem, using our greedy method.

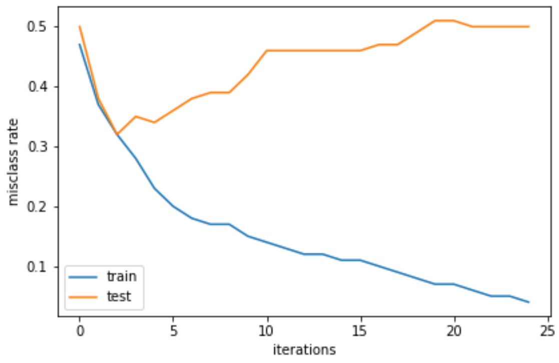
- Download `covtype.zip`, which is a remote sensing classification problem (more details here `https://archive.ics.uci.edu/ml/datasets/covertype`). Inside there are two raw files: `covtype.info` and `covtype.data`. Skim `covtype.info` to understand the basic task.
- The file `covtype_reduced.mat` has all the data already loaded and separated into a train and test set, and is subsampled severely to reduce computational complexity. (Like I said, this is a simple implementation. Go ahead and use scikit-learn for a "full" version.)
- The task is now to construct a decision tree classifier, that uses information gain to pick the best next split. Open the iPython notebook `covtype.ipynb`, and follow the instructions there.
- Fill in the box to return functions that computes entropy and conditional entropy for a sequence of labels. Return the values given by the test problem provided in the notebook. Remember to include special cases for when one is required to take the log of 0. (We assume that $0\log_2(0) = 0$.)
Ans. `entropy = 3.314182323161083, conditional entropy = 3.3029598816135173`
- Fill in the box to return a function that, at each step, given X, y, and a set of data sample indices to consider, returns the best feature to split and the best split value, as well as the indices split into two sets. (Follow the template in the iPython notebook). Again, return the solution to the test problem given. Don't forget to handle the special case if the split results in an empty set–something special should happen, so the algorithm knows to reject this split.
Ans. This could vary. In my implementation, I discretized for each feature from the smallest to largest value in the training set. For 5 discretization points, my IG = 0.207. For 25, it rose to IG = 0.278. For 100, it rose to IG = 0.294. Anything in these ballparks will work.

- The rest of the implementation is up to you. You can continue to use my iPython notebook, or you can implement your own decision tree using 1) your own implementation of trees and nodes, or 2) something you find online. What you *cannot* do is to use an online implementation of a decision tree. (You can steal the tree, but not the decision part.)
- **My first step.** You can run the "first step" box to start to debug your tree. If you are using my implementation, if you have correctly filled in the holes and added the steps, you should get a result that looks similar to this. (The actual implementation of the split function can cause some variation.)

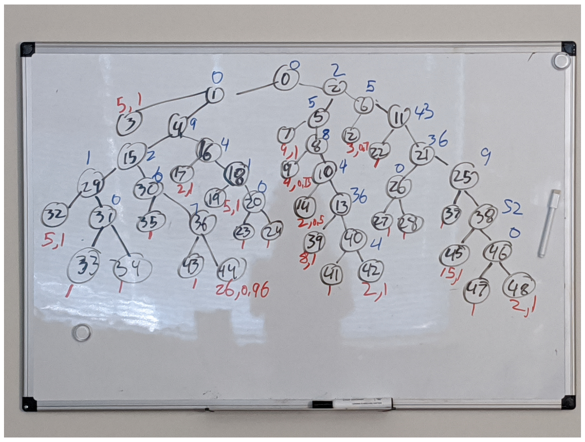
```
printing tree...
    root 0 2.0 leaf nodes  100 number of samples
current train err: 0.47
current test err: 0.5
printing tree...
    root 0 2.0 split 0, val 2930.62
    0 1 2.0 leaf nodes  41 number of samples
    0 2 1.0 leaf nodes  59 number of samples
one step train err: 0.41
one step test err: 0.42
```

- Report your train and test misclassification rate for 25 steps of training. Describe your resulting tree. Report how many nodes there are in your resulting tree, how many are leaves. List, for the leaves, how many samples ended up on each leaf, and report also their "purity", e.g. # values most frequent / total size of set.
- Did your tree overfit? What are some hints that this may have happened?

  Ans. Again, answers may vary, but this is what I got for 25 steps:



After drawing out my tree, here's what it looks like



In particular, the red numbers indicate the number of samples on each leaf node, and, if that number is > 1, its purity. The fact that most of the leaves only contain 1 element is a dead giveaway that the tree has way overfitted, and will probably not generalize well. (The blue numbers show which feature was split. This number did not tell me much.)

5. **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
        $$h^{(t)}(x) = \operatorname*{argmin}_{h\in\mathcal{H}} \sum_{\substack{i=1\\ h(x_i)\neq y_i}}^{m} w_i^{(t-1)}$$
    Update
        $$\alpha^{(t)} = \frac{1}{2}\log\left(\frac{1-\epsilon^{(t)}}{\epsilon^{(t)}}\right), \qquad \epsilon^{(t)} := \sum_{h^{(t)}(x_i)\neq y_i} w_i^{(t-1)} \qquad (\star)$$
    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.** We will now 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

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

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 $\{-1, 1\}$. We will do this in several steps.

i. **(0.3 pts)** Define

$$v_i^{(t)} = \exp\left(-y_i\left(\sum_{t'=1}^{t}\alpha^{(t')}h^{(t')}(x_i)\right)\right).$$

Show that the sequence of $w_i^{(t)}$ and $H^{(t)}(x)$ as generated in the algorithm can be written succinctly as

$$w_i^{(t+1)} = \frac{v_i^{(t)}}{\sum_j v_j^{(t)}}.$$

Ans. There is probably more than one way to make this argument. One way to do this is to set a constant $c^{(t)} = \sum_j \hat{w}_j^{(t+1)}$, and note that it does not depend on j itself. So, then we can write

$$w_i^{(t+1)} = \frac{1}{\bar{c}^{(t)}} \prod_{\tau=1}^{t} \exp(-y_i \alpha^{(\tau)} h^{(\tau)}(x_i))$$

for some $\bar{c}^{(t)} > 0$ that does not depend on i, and in fact, $w_i^{(t+1)}$ is exactly the scaling of the right hand side such that $\sum_i w_i^{(t+1)} = 1$. Therefore, whether we normalize at each step or we normalize at time t only, the result is the same. In other words, $w_i^{(t+1)} = \frac{v_i^{(t)}}{\sum_j v_j^{(t)}}$.

ii. **(0.4 pts)** Noting that $v_i \geq 0$ for all i, define

$$g(\alpha) := \sum_{i=1}^{m} v_i \exp(-y_i \alpha^{(t)} h^{(t)}(x_i)).$$

Show that $g(\alpha)$ is convex, and is minimized when $\alpha = \alpha^{(t)}$ as defined in ($\star$). Hint: observe that $h^{(t)}(x_i) \neq y_i \iff y_i h^{(t)}(x_i) = -1$.

Ans.

First, taking

$$g''(\alpha) = \sum_{i=1}^{m} v_i (\underbrace{-y_i h^{(t)}(x_i)}_{\in\{-1,1\}})^2 \exp(-y_i \alpha^{(t)} h^{(t)}(x_i)) = \sum_{i=1}^{m} v_i \exp(-y_i \alpha^{(t)} h(x_i))$$

Since $v_i \geq 0$ for all i and exponentials are always nonnegative, this implies $g''(\alpha) \geq 0$ for all α, and thus g is convex w.r.t. α. Taking its derivative and setting to 0 will therefore get the minimizer:

$$\begin{aligned} 0 = g'(\alpha) &= \sum_{i=1}^{m} y_i h^{(t)}(x_i) v_i \exp(-y_i \alpha^{(t)} h^{(t)}(x)) \\ &= \sum_{i:y_i h^{(t)}(x_i)=1} v_i \exp(-y_i \alpha^{(t)} h^{(t)}(x_i)) - \sum_{i:y_i h^{(t)}(x_i)=-1} v_i \exp(-y_i \alpha^{(t)} h^{(t)}(x_i)) \end{aligned}$$

we get

$$\sum_{i:y_i h^{(t)}(x_i)=1} v_i \exp(-\alpha^{(t)}) = \sum_{i:y_i h^{(t)}(x_i)=-1} v_i \exp(\alpha^{(t)})$$

of which can be rearranged to

$$\frac{\sum_{i:y_i h^{(t)}(x_i)=1} v_i}{\sum_{j:y_j h^{(t)}(x_j)=-1} v_j} = \exp(2\alpha^{(t)}).$$

Plugging in $\epsilon^{(t)} = \sum_{j:y_j h^{(t)}(x_j)=-1} v_j$ and $1 - \epsilon^{(t)} = \sum_{i:y_i h^{(t)}(x_i)=1} v_i$ and simplifying gives the desired result.

iii. **(0.3 pts)** Use these two parts to argue that the $\alpha^{(t)}$ picked at each iteration is the one that greedily minimizes the loss at each iteration, e.g. satisfies (Δ). That is, show that $g(\alpha)$ is exactly the function where

$$g(\alpha^{(t)}) = \sum_{i=1}^{m} \mathcal{L}(H^{(t)}(x_i); y_i).$$

Ans. Expanding out the right hand side of the first equation in (Δ),

$$\mathcal{L}(H^{(t)}(x_i); y_i) = \exp(-y_i H^{(t)}(x_i)) = \exp(-y_i \sum_{t'=1}^{t} \alpha^{(t')} h^{(t')}(x_i)) = \exp(-y_i \alpha^{(t)} h^{(t)}(x_i)) \cdot \underbrace{\exp(-y_i \sum_{t'=1}^{t-1} \alpha^{(t')} h^{(t')}(x_i))}_{=v_i}$$

We can see that summing over the last term for $i = 1, ..., m$ gives $g(\alpha^{(t)})$.

(b) **Coding. (1 pts)** 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. You may now use sklearn's tree implementation.

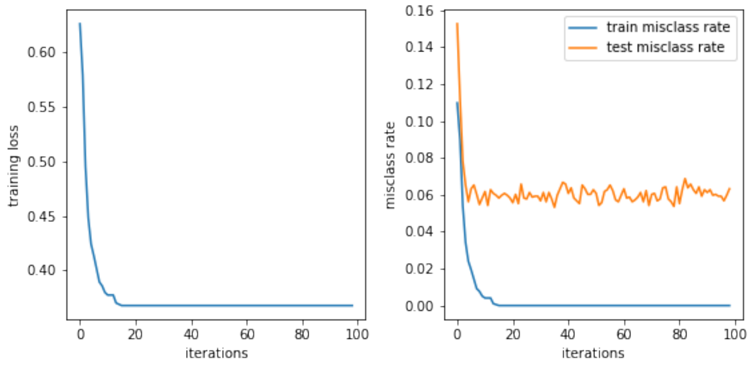
- Write a function that computes the train and misclassification rates , as well as the train exponential loss value, using just this decision stump (tree of depth = 1). Over the given dataset, these values should be:

$$10.98\%\text{ train misclassification}, 15.27\%\text{ test misclasification}, 0.6259675480492947\text{ exponential loss}$$

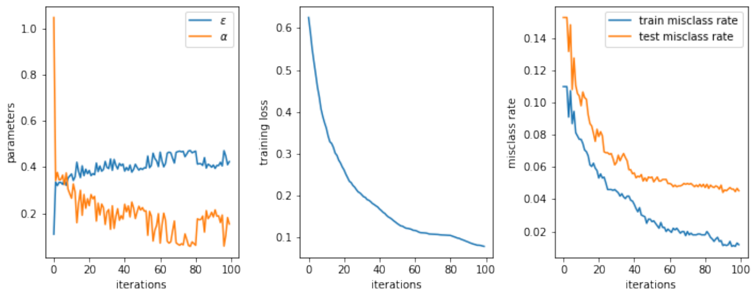
- **Deep trees.** Using the tree library from sklearn, train trees with depth 1 to 100, and plot the train and test misclassification rate, as well as the train exponential loss, as a function of depth. Report also the smallest train and test misclassification rate, and smallest train exponential loss, over the sweep.
- **Boosted decision stumps.** Now build a decision stump, e.g. a 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).
```

  Ans.



  best train loss: 0.368, best train misclassification rate: 0%, best test misclassification rate: 5.32 %
  Implement the Adaboost method, as shown in algorithm 1. Plot the training exponential loss, and train and test misclassification rate. Report also the smallest train and test misclassification rate, and smallest train exponential loss, over the sweep. Comment on how the boosted decision stumps performed compared to the deep decision tree.
  Ans.



  The second and third plot show the train loss and both misclasification rates evolutions.
  best train loss: 0.078, best train misclassification rate: 1.07%, best test misclassification rate: 4.42 %
  In general, the boosted decision stumps seemed to be much better than the deep decision tree, despite similar computational complexity!

- 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?
  Ans. The first plot above shows the evolution of α and ϵ. We notice that α is larger when ϵ is smaller (weak classifier did well!), and $\alpha \to 0$ as $\epsilon \to 0.5$ (about half the samples are classified correctly). In

this way, boosting seems to put appropriate weight on the weak classifier based on its contribution to improvement.

Challenge!

1. **Medians.** We will show that the solution to the scalar optimization problem

$$\underset{x}{\text{minimize}} \quad \sum_{i=1}^{m} |z_i - x|$$

is in fact achieved when x^* is the median of z_i. We can do this using *subdifferentials*, which are a generalization of gradients. In particular, for a function that is differentiable almost everywhere, you can find the subdifferential by taking the convex hull of all the derivatives approaching that point; e.g.

$$\partial f(x) = \text{conv}\left(\left\{\lim_{\epsilon\to 0} \nabla f(x+\epsilon z) : \text{ for all z } \right\}\right).$$

For reference, the *convex hull* of $\mathcal{S}$ is defined as the set that contains all convex combinations of elements in $\mathcal{S}$:

$$\text{conv}(\mathcal{S}) := \{\theta x + (1-\theta)y : x \in \mathcal{S}, y \in \mathcal{S}\}.$$

(a) Show that the subdifferential of the absolute function $f(x) = |x|$ is

$$\partial f(x) = \begin{cases} \{1\}, & \text{if } x > 0 \\ \{-1\}, & \text{if } x < 0 \\ [-1, 1], & \text{if } x = 0. \end{cases}$$

Ans. If $x \neq 0$, then $f(x)$ is smooth at x, and the limiting gradient is the same from all directions: it's just the gradient $f'(x) = \textbf{sign}(x)$. If $x = 0$, then the gradient coming from $x < 0$ is always -1, and the gradient coming from $x > 0$ is always $+1$. So, $\partial f(0) = [-1, 1]$, which is the convex hull of $\{-1, 1\}$.

(b) For a minimization of a nonsmooth convex function $g(x)$,

$$x^* = \underset{x}{\text{argmin}}\ g(x) \quad \iff \quad 0 \in \partial g(x^*).$$

Write down the subdifferential of $f(x) = \sum_{i=1}^{m} |z_i - x|$. Assume that each z_i are distinct (no 2 values are the same).

Ans. Define $f_i(x) = |x - z_i|$. Then $f(x) = \sum_i f_i(x)$, and we can use linearity of gradients to argue that $\partial f(x) = \sum_i \partial f_i(x)$. Using the previous answer, we know that

$$\partial f_i(x) = \begin{cases} \{\textbf{sign}(x - z_i)\}, & x \neq z_i \\ [-1, 1] & , x = z_i. \end{cases}$$

Since we assume that z_i are all distinct, we know that $x = z_i$ can only happen for at maximum one z_i. Therefore, we can summarize our answer as

$$\partial f(x) = \begin{cases} \{\sum_i \textbf{sign}(x - z_i)\}, & x \neq z_i \text{ for all } i \\ [-1+d, 1+d],\ d = \sum_{i \neq j} \textbf{sign}(x - z_j), & x = z_i \text{ for some } i. \end{cases}$$

(c) Use these pieces to argue that x^* is the median of z_i, under the assumption that each z_i are distinct and m is an odd number.

Ans. We know that $x = x^*$ when $0 \in \partial f(x^*)$. From the previous answer, assuming that each z_i are distinct, we have one of two choices: either $x = z_i$ for one i, or $x \neq z_i$ for no i. Since m is an odd number, then the first choice is not possible, since it is not possible for an odd number of 1's, -1's to sum up to 0.

Therefore it must be that $x = z_i$ for one i. Then, all the $z_j > z_i$ result in $\textbf{sign}(x - z_j) = 1$ and all the $z_k < z_i$ result in $\textbf{sign}(x - z_k) = -1$. In order for these numbers to sum to some number in the interval $[-1, 1]$, there must be exactly equal number of $z_j > z_i$ as $z_k < z_i$.

Therefore x must be the median of the collection $\{z_1, ..., z_m\}$.

2. 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

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

   Ans. To maximize entropy, I will need to buy 5 red socks and 5 black socks. In fact, this is just a special instance of the next problem; namely, to maximize the entropy using $p = \mathbf{Pr}(X = \text{red})$, we would solve

$$\begin{array}{ll} \underset{p}{\text{maximize}} & f(p) := -p\log_2(p) - (1-p)\log_2(1-p) \\ \text{subject to} & 0 \le p \le 1. \end{array}$$

   Using a graphing calculator, or by setting the objective gradient to 0, we see that the minimum of $f(p)$ is reached at $p = 1/2$. This trick works because f is concave over the domain, and the proposed stationary point is in the interior of the constraint.

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

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

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

   Ans. By now, we will just use our favorite trick, which is to show that $f(p)$ is concave, take the derivative, and show that $f'(p) = 0$ at a value of p which is in the interior of the feasible set.

   Step 1: Show that f is concave in p over the feasible domain.

$$f'(p) = -\log_2(p) - \frac{1}{\ln(2)} + \log_2(c-p) + \frac{1}{\ln(2)} = -\log_2(p) + \log_2(c-p)$$

$$f''(p) = -\frac{1}{p\ln(2)} - \frac{1}{(c-p)\ln(2)} \le 0, \quad \forall 0 \le p \le c.$$

   Therefore f is concave over the domain of p.

   Step 2: Find the point where $f'(p) = 0$. This occurs at

$$\log_2(p) = \log_2(c-p) \iff p = c - p \iff p = c/2.$$

   Step 3: Since $c/2$ is squarely in the interior of the feasible set, we know that $f(p)$ reaches its highest value 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.

   Ans. The idea here is to conclude that in all cases, the uniform distribution maximizes entropy. In other words, in general, if there are K possible socks to choose from, then

$$H(X) = -\sum_{i=1}^{K} p_i \log_2(p_i)$$

   and

$$\begin{array}{ll} \underset{p_i}{\text{maximize}} & -\sum_{i=1}^{K} p_i \log_2(p_i) \\ \text{subject to} & \sum_i p_i = 1, \quad p_i \ge 0\ \forall i \end{array}$$

is achieved when $p_i = 1/K$ for all i.

To prove this rigorously, we form a recursive proof. If $K = 1$, then $p_1 = 1$; this is a trivial case. Taking the above problem's solution for $c = 1$ also proves it for $K = 2$.

Now suppose that for some $K > 2$, we have a distribution where $p_i \neq p_j$ for $i \neq j$. Take $c = p_i + p_j$. Then we know that

$$-p_i \log_2(p_i) - p_j \log_2(p_j) < -\tfrac{c}{2}\log_2(\tfrac{c}{2}) - \tfrac{c}{2}\log_2(\tfrac{c}{2})$$

again, as a conclusion from the previous question.

In other words, for any non-uniform distribution, we can find a distribution whose entropy is strictly greater than the one proposed, simply by finding any two values with different probabilities, and showing that a distribution with those two values set equal has a higher entropy.

Therefore we can conclude that a uniform distribution maximizes entropy.

Therefore when I go to Mall of America, I should buy 100 socks of every color, to maximize my drawer's entropy.

Alternative student-inspired proof. Since everything is in terms of probability, we can write

$$p_{100} = 1 - \sum_{i=1}^{99} p_i$$

and the entropy of my selection is

$$H(p) = -\sum_{i=1}^{99} p_i \log(p_i) - p_{100}\log(p_{100}) = -\sum_{i=1}^{99} p_i \log(p_i) - \left(1 - \sum_{i=1}^{99} p_i\right) \log\left(1 - \sum_{i=1}^{99} p_i\right).$$

Let's take some partial derivatives

$$\begin{aligned}
\frac{\partial H}{\partial p_i} &= -\log(p_i) + \log\left(1 - \sum_{i=1}^{99} p_i\right)\\
\frac{\partial^2 H}{\partial p_i^2} &= -\frac{1}{p_i} - \frac{1}{1 - \sum_{i=1}^{99} p_i}\\
\frac{\partial^2 H}{\partial p_i \partial p_j} &= -\frac{1}{1 - \sum_{i=1}^{99} p_i}
\end{aligned}$$

which means that

$$\nabla^2 H = -\mathbf{diag}(p)^{-1} - s\mathbf{1}\mathbf{1}^T, \quad s = \frac{1}{1 - \sum_{i=1}^{99} p_i}$$

which using our usual $u^T \nabla^2 H u \leq 0$ for all u trick shows that H is concave over the domain of possible p. Setting the gradient to 0, we see from our partial derivatives that this implies the stationary point for all i,

$$\log(p_i) = \log\left(1 - \sum_{j=1}^{99} p_j\right) \iff p_i = 1 - \sum_{j=1}^{99} p_j \iff p_i = 1/100, \ \forall\, i.$$

Since this solution is in the interior of the feasibility constraint $0 \leq p \leq 1$, this is the global maximum.

Alternative student-inspired proof. We use the strict convexity of the function $\log(x)$ to claim that

$$H(p) = -\sum_{i=1}^{m} p_i \log_2(p_i) = \sum_{i=1}^{m} p_i \log_2\left(\frac{1}{p_i}\right) \leq \log_2\left(\sum_{i=1}^{m} \frac{p_i}{p_i}\right) = \log_2(m)$$

and equality is only achieved if p_i are all equal, e.g. uniform distribution.