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 $m \to +\infty$.

   Ans. (1 pts) 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.
We know that $\hat{x}$ is an unbiased estimator of x; namely,
$$\mathbb{E}[\hat{x}] = \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} y_i\right] = \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}[y_i] = x.$$
The variance of this estimator is
$$\begin{aligned}\mathbf{Var}(\hat{x}) &= \mathbb{E}\left[\left(\frac{1}{m}\sum_i y_i - x\right)^2\right] \\ &= \mathbb{E}\left[\left(\frac{1}{m}\sum_i \underbrace{(y_i - x)}_{z_i}\right)^2\right]\end{aligned}$$
which is the variance of $\frac{1}{m}\sum_i z_i$. By linearity, this is the variance of $\frac{1}{m}\mathbf{var}(z_i) = \frac{1}{m}$.

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

   Ans. (1 pts) By setting the gradient of the objective to 0, I see that this estimator can be written explicitly as
$$x_{\text{MAP}} = \frac{1}{2+\rho}\left(\frac{2}{m}\sum_{i=1}^{m} y_i + \rho\bar{x}\right)$$
For positive ρ, the bias can be computed as
$$\mathbb{E}[x_{\text{MAP}}] = \frac{1}{2+\rho}\left(\frac{2}{m}\sum_{i=1}^{m}\mathbb{E}[y_i] + \rho\bar{x}\right) = \frac{1}{2+\rho}(2x + \rho\bar{x})$$

which is biased if $x \neq \bar{x}$.

The variance can be computed as

$$\mathbf{var}(x_{\text{MAP}}) = \frac{1}{(2+\rho)^2}\left(\frac{4}{m^2}\sum_{i=1}^m \mathbf{var}(\mathbb{E}[y_i])\right) = \frac{4}{m(2+\rho)^2}$$

again, using linearity of variance ($\mathbf{var}(aX) = a^2\mathbf{var}(X)$, $\mathbf{var}(X+Y) = \mathbf{var}(X) + \mathbf{var}(Y)$ if X and Y are independent.) Clearly, 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) 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.

Ans. (1 pts)

$$\begin{aligned}
\mathbb{E}[(x-\hat{x})^2] &= \mathbb{E}[(x - \mathbb{E}[\hat{x}] + \mathbb{E}[\hat{x}] - \hat{x})^2] \\
&= \mathbb{E}[\underbrace{(x-\mathbb{E}[\hat{x}])^2}_{\text{not random}}] + \mathbb{E}[(\mathbb{E}[\hat{x}]-\hat{x})^2] + \underbrace{2\mathbb{E}[(x-\mathbb{E}[\hat{x}])(\mathbb{E}[\hat{x}]-\hat{x})]}_{=0} \\
&= \underbrace{(x-\mathbb{E}[\hat{x}])^2}_{B^2} + \underbrace{\mathbb{E}[(\mathbb{E}[\hat{x}]-\hat{x})^2]}_{V}
\end{aligned}$$

(d) In terms of constants $\Delta = x - \bar{x}$ and m, what value of ρ minimizes the 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.

Ans. (1 pts) From the previous parts, we have that

$$\mathbb{E}[(x-\hat{x})^2] \quad = \quad B^2 + V = \left(\frac{(2x+\rho\bar{x})}{(2+\rho)} - x\right)^2 + \frac{4}{m(2+\rho)^2}.$$

This is not a convex function of ρ, but using the hint, we can express $\beta = \frac{2}{2+\rho} \iff \rho = (2-2\beta)/\beta$. Then

$$\begin{aligned}
\mathbb{E}[(x-\hat{x})^2] &= (\beta x + (1-\beta)\bar{x} - x)^2 + \frac{\beta^2}{m} \\
&= (1-\beta)^2\Delta^2 + \frac{\beta^2}{m} =: g(\beta)
\end{aligned}$$

which *is* convex in $0 \leq \beta \leq 1$. This can be shown, for example, by observing that $g''(\beta) = \Delta^2 + 1/m > 0$. Setting the derivative to 0 gives

$$0 = 2(\beta-1)\Delta^2 + \frac{2}{m}\beta \iff \beta = \frac{\Delta^2 m}{\Delta^2 m + 1} \iff \rho = \frac{2}{\Delta^2 m}.$$

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 $O(1/m)$.

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.

  Ans. (0.5 pts) There were 3 observations I personally had.

  - The dataset is extremely sparse. This is actually a good thing for decision trees, which often handle sparse features better than other classifiers.
  - The dataset has way more features than samples. Such a dataset will be prone to overfitting; hence, the need for cross validation!
  - The dataset labels are very unbalanced–there are way more instances of $y = -1$ than $y = +1$. We need to keep track of this, and make sure our evaluation metrics and tree design take this into account.

- 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?

  Ans. (0.25 pts) Answers may vary, but I got 0.041 as my train misclassification rate, and 0.0743 as my test misclassification rate. On face value these numbers are terrific! But recall that if I predict -1 for everything, I get a 10% misclassification rate. So... maybe we hold the champagne.

- An alternative performance metric when dealing with unbalanced dataset is the F1 score, which can be written as

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

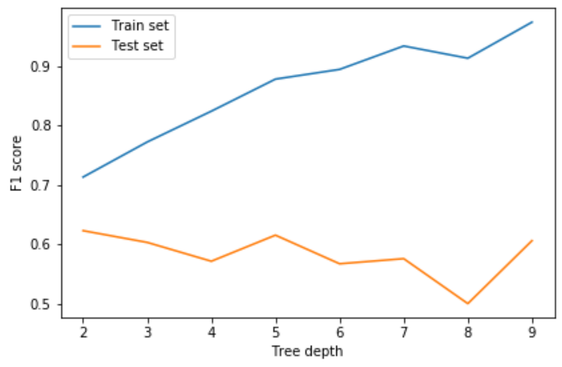
  For our data set, the event that $y_i = 1$ is a rare event, and we want to measure our ability to detect this rare event. So, an event is detected if $y_i = 1$ and $\hat{y}_i = 1$, retrieved if $\hat{y}_i = 1$, and relevant if $y_i = 1$.

  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?

  Ans. (0.25 pts) Answers may vary, but I got 0.7724 as my train F1 score, and 0.5806 as my test F1 score. Overall, these scores are too bad, as a totally random guess gets me an F1 score of about 0. So, I may believe that my classifier isn't terrible! (Champagne uncorked!)

- *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.

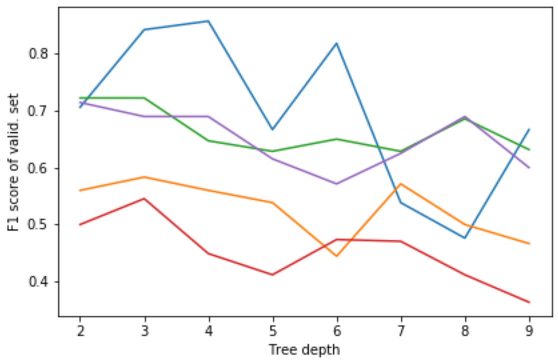
  Ans. (0.5 pts) Below is my plot



  As you can see, the train F1 score increases with depth, so it really wants the tree to be as deep as possible! But the test F1 score doesn't seem to grow at all, suggesting that maybe a shallow tree is sufficient for generalization.

- 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?

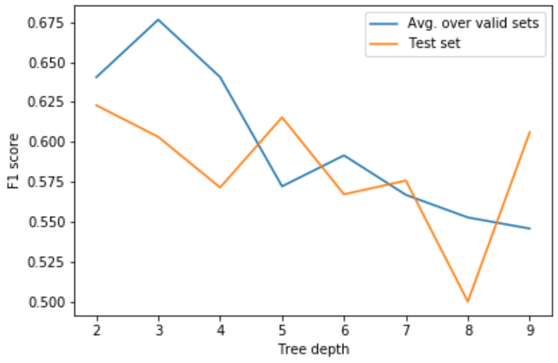
  Ans. (1 pts) Below is my plot



The variance in the validation F1 scores is pretty high, and it's obvious that just using one of the scores isn't reliable. However, on average, there is a downward trend of favoring shallow trees.

- 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?

  Ans. (0.5 pts) Below is my plot



Well, the validation score shows a clear preference for the shallowest tree ever. The test score isn't that steady, but also seems to concur with this conclusion. Well, I feel much more assured that shallow trees are the way to go!

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.)

Ans. (1 pts) If we write our loss function as a function of the newly updated α, it will look like

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

So, our goal is to show that the adaboost algorithm update for $\alpha^{(t)}$ is indeed the α that minimizes $f(\alpha)$. Before going further, we first do some simplifications. We notice that

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

so we can simplify f as

$$f(\alpha) = \sum_{i=1}^{m} w_i^{(t-1)}\exp(-\alpha z_i), \qquad z_i = y_i h^{(t)}(x_i) \in \{-1, 1\}.$$

Now to show that f is convex in α, we first compute the two derivatives

$$f'(\alpha) = \sum_{i=1}^{m} -z_i w_i^{(t-1)}\exp(-\alpha z_i), \qquad f''(\alpha) = \sum_{i=1}^{m}(z_i)^2 w_i^{(t-1)}\exp(-\alpha z_i).$$

In particular, in the $f''(\alpha)$ form, each element in the sum is nonnegative, so $f''(\alpha) \geq 0$ for all α. So, since α is unconstrained, we know we are just looking for α where $f'(\alpha) = 0$. We use the last hint to break apart the sum:

$$f'(\alpha) = \sum_{i:z_i=1}\left(-w_i^{(t-1)}e^{-\alpha}\right) + \sum_{z_i=-1} w_i^{(t-1)}e^{\alpha} = 0$$

which, scaling both terms by e^{α} and solving for $e^{2\alpha}$ yields

$$e^{2\alpha} = \frac{\sum_{i:z_i=1} w_i^{(t-1)}}{\sum_{z_i=-1} w_i^{(t-1)}} = \frac{1-\epsilon^{(t)}}{\epsilon^{(t)}},$$

since $\sum_{i:z_i=-11} w_i^{(t-1)}$ is just the weighted misclassification rate. Solving for α now yields the claimed result.

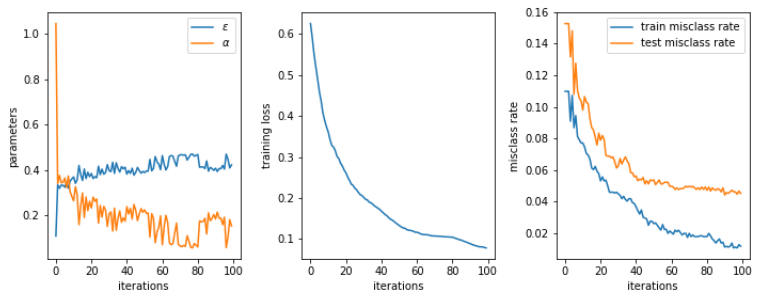
(b) **Coding.** Open the mnist_adaboost_release directory and in the iPython notebook, download the data. We are again going to do 4/9 handwriting disambiguation. Only minimal preprocessing was used in this dataset, since for decision trees, normalization is less of an issue.

- 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. **Ans.** Answers may vary. I got a train misclassification rate of 0.110, a test misclassification rate of 0.153, and a training exponential loss of 0.626.

- 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?
  Ans. (2 pts) The code will be attached, and here is my final performance plot.



In fact, I was pretty surprised to see the performance outperforms linear and logistic regression easily. We also notice that $\alpha \to +\infty$ as $\epsilon \to 0$ (weak classifier did well!), $\alpha \to -\infty$ as $\epsilon \to 0$ (weak classifier did terribly!), 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. 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 $p = 1/2$.

   (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

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

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.

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.

Ans. It suffices to show that $\mathbf{Pr}(Y_2 = 1) = p$, since everything else will happen recursively. Using Law of Total Probability,

$$\begin{aligned} \mathbf{Pr}(Y_2 = 1) &= \mathbf{Pr}(Y_2 = 1|Y_1 = 1)\mathbf{Pr}(Y_1 = 1) + \mathbf{Pr}(Y_2 = 1|Y_1 = -1)\mathbf{Pr}(Y_1 = -1) \\ &= \mathbf{Pr}(Y_2 = 1|Y_1 = 1)\mathbf{Pr}(Y_1 = 1) + (1 - \mathbf{Pr}(Y_2 = -1|Y_1 = -1))(1 - \mathbf{Pr}(Y_1 = 1)) \\ &= (c + p - cp)p + (1 - (cp - p + 1))(1 - p) \\ &= p. \end{aligned}$$

(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.

Ans. Taking each piece one at a time,

$$\begin{aligned}
\mathbb{E}[Y_iY_{i-1}] &= \mathbf{Pr}(Y_i = 1, Y_{i-1} = 1) + \mathbf{Pr}(Y_i = -1, Y_{i-1} = -1) - \mathbf{Pr}(Y_i = 1, Y_{i-1} = -1) - \mathbf{Pr}(Y_i = -1, Y_{i-1} = 1)\\
&= \mathbf{Pr}(Y_i = 1|Y_{i-1} = 1)\mathbf{Pr}(Y_{i-1} = 1) + \mathbf{Pr}(Y_i = -1|Y_{i-1} = -1)\mathbf{Pr}(Y_{i-1} = -1)\\
&\quad -\mathbf{Pr}(Y_i = 1|Y_{i-1} = -1)\mathbf{Pr}(Y_{i-1} = -1) - \mathbf{Pr}(Y_i = -1|Y_{i-1} = 1)\mathbf{Pr}(Y_{i-1} = 1)\\
&= (c + p - cp)p + (cp - p + 1)(1 - p) - (1 - (cp - p + 1))(1 - p) - (1 - (c + p - cp))p\\
&= 1 + 4p(c(1 - p) + p - 1)
\end{aligned}$$

Then

$$\begin{aligned}
\mathbb{E}[Y_i] &= p - (1 - p) = 2p - 1\\
\mathbf{var}(Y_i) &= \mathbb{E}[\underbrace{Y_i^2}_{=1}] - (\mathbb{E}[Y_i])^2 = 1 - (2p - 1)^2
\end{aligned}$$

Putting it all together,

$$\mathbf{corr}(Y_i, Y_{i-1}) = \frac{1 + 4p(c(1 - p) + p - 1) - (2p - 1)^2}{1 - (2p - 1)^2} = c.$$

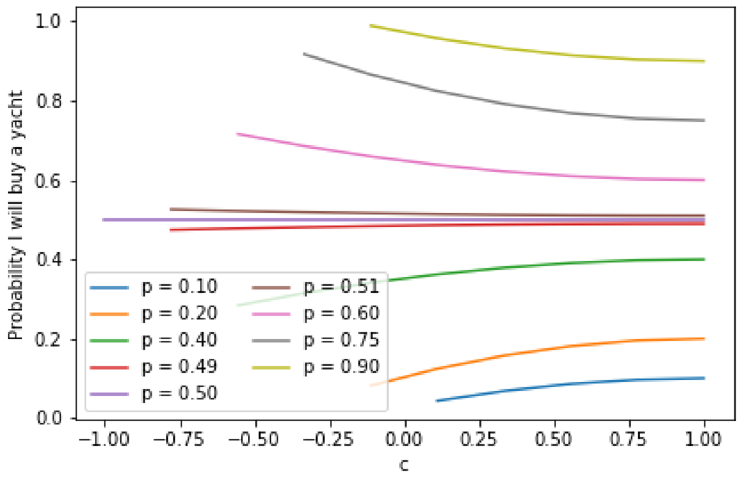
(c) For $m = 3$, what is the probability that I will buy a yacht, in terms of $P_{11} = \mathbf{Pr}(Y_i = 1|Y_{i-1} = 1)$,$P_{00} = \mathbf{Pr}(Y_i = -1|Y_{i-1} = -1)$ and p?

Ans. For $m = 3$, I will buy a yacht if any two or all 3 advisers advise for it. That is,

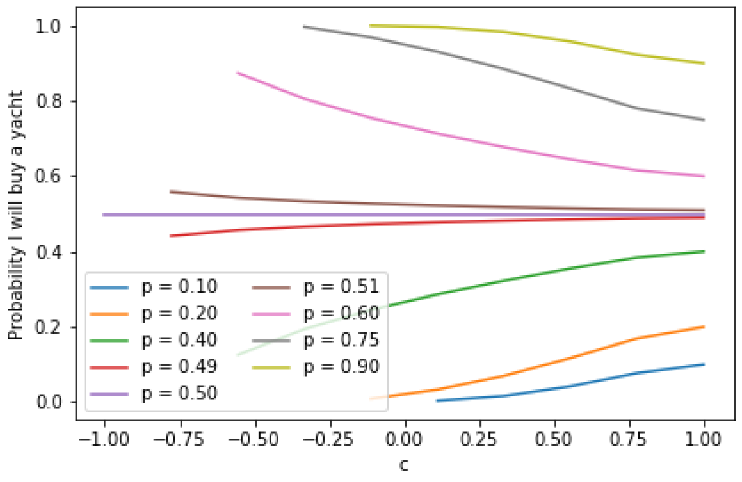
$$\begin{aligned}
\mathbf{Pr}(\text{yacht is bought}) &= \mathbf{Pr}(Y_1 = 1, Y_2 = 1) + \mathbf{Pr}(Y_1 = -1, Y_2 = 1, Y_3 = 1) + \mathbf{Pr}(Y_1 = 1, Y_2 = -1, Y_3 = 1)\\
&= \mathbf{Pr}(Y_2 = 1|Y_1 = 1)\mathbf{Pr}(Y_1 = 1) + \mathbf{Pr}(Y_3 = 1|Y_2 = 1)\mathbf{Pr}(Y_2 = 1|Y_1 = -1)\mathbf{Pr}(Y_1 = -1)\\
&\quad +\mathbf{Pr}(Y_3 = 1|Y_2 = 1)\mathbf{Pr}(Y_2 = -1|Y_1 = 1)\mathbf{Pr}(Y_1 = 1)\\
&= P_{11}p + P_{11}(1 - P_{00})(1 - p) + P_{11}(1 - P_{11})p
\end{aligned}$$

(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.

Ans. Coding this up requires 10 nested for loops, but still I was able to produce a fairly smooth plot waiting only maybe 1 minute total. What's interesting here is to see how, when c is low, the experts have the ability to "gravitate" toward 1 if $p > 0.5$ and 0 otherwise; but as c approaches 1, the "gravitation speed" seems to reduce, converging to the original probability p. With 10 vs 3 experts, it seems like the "separation" lasts a bit longer (for larger values of c), but still exhibits a similar trend. This is consistent with what we would expect: more experts is better (concentrates on a more definitive choice), but higher correlation leads to less useful information.



3 experts

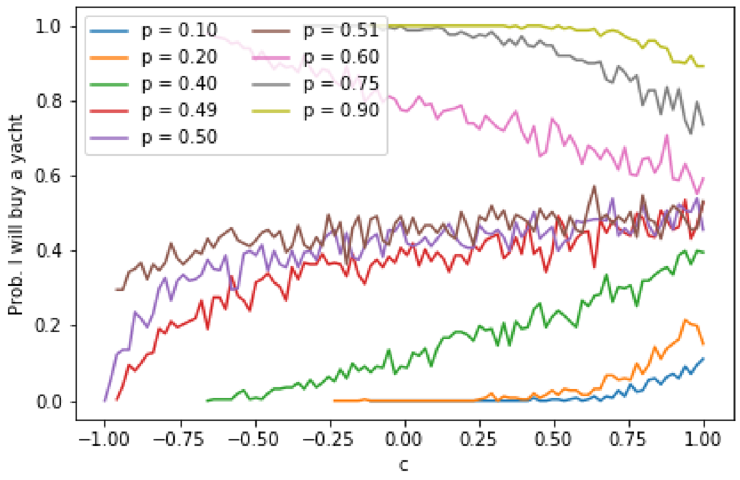


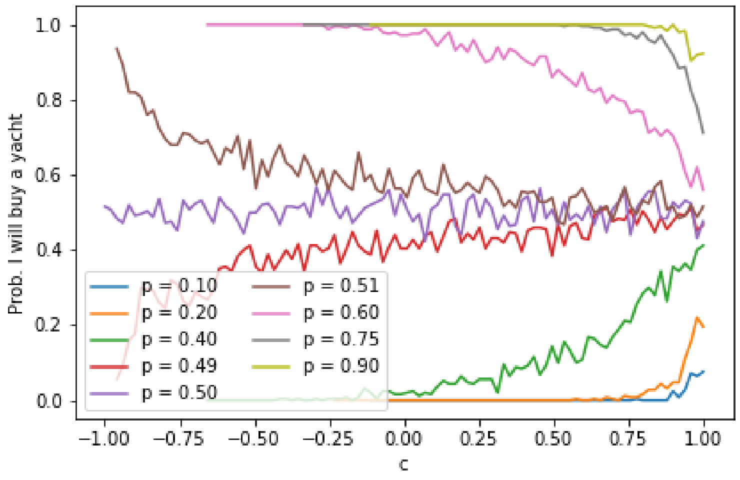
10 experts

(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.

Ans. Here are plots I generated, averaging over 250 trials. It's pretty messy, but it's clear that the trends are similar!

At this point we can also notice an interesting separation between the initial seed probabilities p: for p closer to 1 or 0, more experts allow for a more concentrated decision, even as c is close to 1. But for p closer to 0.5, more experts seem to cause more uncertainty, concentrating to 0.5 as m increases!


p = 0.10
p = 0.20
p = 0.40
p = 0.49
p = 0.50
p = 0.51
p = 0.60
p = 0.75
p = 0.90
Prob. I will buy a yacht
c


p = 0.10
p = 0.20
p = 0.40
p = 0.49
p = 0.50
p = 0.51
p = 0.60
p = 0.75
p = 0.90
Prob. I will buy a yacht
c