CSE 512: Homework 4 — Due Oct. 19

1. **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) 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 picked. What is the entropy of this sock?

   Ans. (1 pts)

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

   (b) 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

   Ans. (1 pts)

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

   (c) 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. (1 pts)

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

2. **Naive Bayes for next word prediction.** We will consider the task of word prediction; that is, given a sentence, we predict the next word.

   (a) Consider the following text from "Alice in Wonderland".

   ```
   There was nothing so very remarkable in that; nor did Alice think it so very much out of the
   ```

```
way to hear the Rabbit say to itself, ‘‘Oh dear!  Oh dear!  I shall be late!" (when she thought
it over afterwards, it occurred to her that she ought to have wondered at this, but at the time
it all seemed quite natural); but when the Rabbit actually took a watch out of its waistcoat-pocket,
and looked at it, and then hurried on, Alice started to her feet, for it flashed across her
mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take
out of it, and burning with curiosity, she ran across the field after it, and fortunately was
just in time to see it pop down a large rabbit-hole under the hedge.
```

Using this text as a reference, what are the probabilities

i. $\mathbf{Pr}(\text{current word} = \texttt{rabbit} | \text{previous word} = \texttt{the})$
Ans. (0.25 pts)
$$\frac{\#\ \texttt{rabbit follows the}}{\#\texttt{the}} = \frac{2}{6} = 1/3$$

ii. $\mathbf{Pr}(\text{previous word} = \texttt{a} | \text{current word} = \texttt{rabbit}\ )$
Ans. (0.25 pts)
$$\frac{\#\ \texttt{rabbit follows a}}{\#\texttt{rabbit}} = \frac{1}{3}$$

iii. $\mathbf{Pr}(\text{previous word} = \texttt{the} | \text{current word} = \texttt{rabbit})$
Ans. (0.25 pts)
$$\frac{\#\ \texttt{rabbit follows the}}{\#\texttt{rabbit}} = \frac{2}{3}$$

iv. $\mathbf{Pr}(\text{previous word} = \texttt{the or a} | \text{current word} = \texttt{rabbit})$ **Ans. (0.25 pts)**
$$\frac{\#\ \texttt{rabbit follows the, a}}{\#\texttt{rabbit}} = \frac{3}{3} = 1$$

v. Is the Naive Bayes assumption valid here, if previous word is used as a feature to predict future word?
Ans. (0.5 pts)
This question is not really well-posed, and thus we will basically give you full credit if you just say what the Naive Bayes assumption is. (Anything along the lines of $\mathbf{Pr}(x[i], x[j]|y) = \mathbf{Pr}(x[i]|y)\mathbf{Pr}(x[j]|y)$ will work.)
This question would make more sense if we thought about not just the previous word, but the previous 2 words. In this case, language structure is not that random, and it would be evident that

$\mathbf{Pr}(\text{previous 2 words}|\text{current word}) \neq \mathbf{Pr}(\text{previous word}|\text{current word})\mathbf{Pr}(\text{previous previous word}|\text{current word}).$

Alas, that's not what the question asked, so we can skip it.

(b) **Coding.** Download `alice.zip`, which contains

- The full "Alice in Wonderland" raw text from Project Gutenberg
- A pickle file containing the preprocessed count and previous word count
- A python notebook that shows how the data was preprocessed

i. Using these files, construct a function that, given a word (y), returns the probability $\mathbf{Pr}(y)$, sampled over the corpus.

ii. Construct now a function that, given a word (x) and its next word (y), returns the probability $\mathbf{Pr}(x|y)$.

iii. Construct a predictor that, given a word x, returns the Naive Bayes estimate of the next word. Report here the word most likely to follow

A. `‘a’`
B. `‘the’`
C. `‘splendidly’`
D. `‘exclaimed’`

Ans. (1.5 pts)

A. `'a little'`

B. `'the queen'`

C. `'splendidly dressed'`

D. `'exclaimed alice'`

3. **Decision trees (coding).** 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.

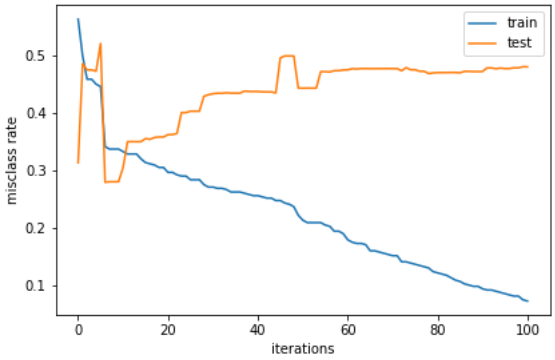
   - Download `covtype.zip`, which is a remote sensing classification problem (more details here `https://archive.ics.uci.edu/ml/datasets/covertype`). Inside there are two raw files: `covtype.info` and `covtype.data`. Skim `covtype.info` to understand the basic task.
   - The file `covtype_reduced.mat` 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.

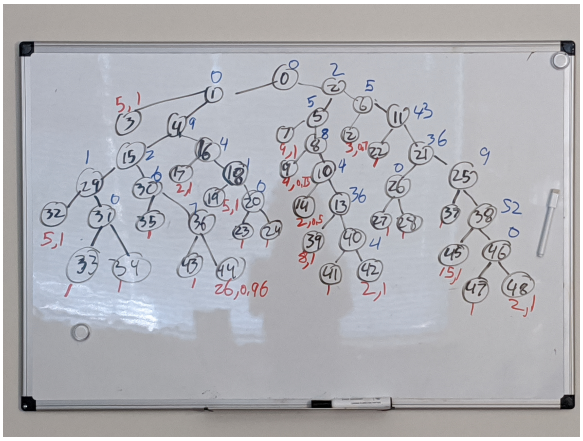
     Ans. This could vary. In my implementation, I discretized for each feature from the smallest to largest value in the training set. For 5 discretization points, my IG = 0.207. For 25, it rose to IG = 0.278. For 100, it rose to IG = 0.294. Anything in these ballparks will work.
   - The rest of the implementation 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.)

Challenge!

1. **Word generation.** Previously, you used the Naive Bayes classifier to predict next words, by finding the word that maximized $\mathbf{Pr}(\text{next}|\text{previous})$. Now, create a text generator. Starting with a seed word x_0, pick x_1 by sampling the next word with probability $\mathbf{Pr}(x_1|x_0)$; then pick x_2 by sampling via the probability $\mathbf{Pr}(x_2|x_1)$, and so forth. Use whatever seed word you want, and submit your generated paragraph. What do you think of your text generator?

   Ans. (4 pts) **Note**: Most of the solutions I saw either repeated the previous exercise of picking the next word based on highest probability, or randomly picked from top k responses. Actually, what I was looking for is a *random sampling* of the next word, based on the probability computed. If you did this, you would not have seen cycling of words and did not need to pick a hyperparameter for k.

   The solution code is attached in the python notebook. Here is one fun example, generated with seed word `the`:

   ```
   the first was the king and a little alice the last she went on a very little of the dance said the mouse to be a voice of a little and all the time the mock turtle said the look for the words i said the queen said the time she was a very ill be of the poor alice to the gryphon and the well be the king the time the hatter was and the first she and she said the jury and the table she had at the queen had to the king and the place on the queen
   ```

2. **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 $f'(x) = \mathbf{sign}(x)$. 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

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

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 $\{z_1, ..., z_m\}$.

3. **Jensen's inequality.** 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

$$\mathbb{E}[f(X)] = \sum_{x \in \mathcal{X}} f(x) p_X(x).$$

Assume that $|\mathcal{X}|$ is finite; that is, there are only a finite number of points where $X = x$ with nonzero probability. Show that if f is a *strictly* convex function; that is, if

$$f(\theta x + (1 - \theta)y) < \theta f(x) + (1 - \theta) f(y), \quad \text{for all } x \neq y,\ 0 < \theta < 1$$

then

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

Ans. Without loss of generality, we can write $\mathcal{X} = \{x_1, ..., x_m\}$, where $m = |\mathcal{X}|$. We write $p_i = p_X(x_i)$. Then 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. First, consider the base case of $m = 2$. Then

$$f(\mathbb{E}[X]) = f\left(x_1 p_1 + x_2(1 - p_1)\right) \quad < \quad p_1 f(x_1)) + (1 - p_1) f\left(x_2\right) = \mathbb{E}[f(x)].$$

Now assume that Jensen's inequality holds for $|\mathcal{X}| \leq m - 1$. We now explore $|\mathcal{X}| = m$:

$$\begin{aligned} f(\mathbb{E}[X]) = f\left(\sum_{i=1}^{m-1} x_i p_i + x_m p_m\right) &\overset{(a)}{<} (1 - p_m) f\left(\frac{1}{1 - p_m} \cdot \sum_{i=1}^{m-1} x_i p_i\right) + p_m f(p_m) \\ &\overset{(b)}{<} (1 - p_m) \cdot \left(\sum_{i=1}^{m-1} \frac{p_i}{1 - p_m} f(x_i)\right) + p_m f(p_m) \\ &= \sum_{i=1}^{m} p_i f(x_i) \\ &= \mathbb{E}[f(x)] \end{aligned}$$

where (a) is from the strong convexity of f using $\theta = p_m$ and (b) is from the inductive step (holds for $|\mathcal{X}| = m - 1$).

Alternative student-inspired proof: Since f is strictly convex, we know that for all x, y,

$$f(x) > f(y) + (x - y)f'(y).$$

Pick $y = \mathbb{E}[x]$. Then

$$f(x) > f(\mathbb{E}[x]) + (x - \mathbb{E}[x])f'(\mathbb{E}[x]).$$

Since this inequality holds pointwise, over all x, we can also take the expectation over both sides. In particular, note that

$$\mathbb{E}[(x - \mathbb{E}[x])f'(\mathbb{E}[x])] = (\mathbb{E}[x] - \mathbb{E}[x])f'(\mathbb{E}[x]) = 0.$$

Therefore,

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

4. **Exponential distribution.** In the previous homework, you worked with the exponential distribution, defined by a pdf of

$$p_\lambda(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x > 0 \\ 0 & \text{else.} \end{cases}$$

In particular, you showed that, given samples $x_1, ..., x_m$ drawn i.i.d. from this distribution, that $\hat{\theta} = \frac{1}{m}\sum_{i=1}^{m} x_i$ served as both the maximum likelihood and unbiased estimator for the true mean, $\frac{1}{\lambda}$.

(a) Derive the MLE for λ. **Ans.** If $\hat{\theta}$ is the maximum likelihood estimate for $1/\lambda$, then since $f(x) = 1/x$ is an invertible function for $x > 0$, then

$$\frac{1}{\hat{\theta}} = \frac{m}{\sum_{i=1}^{m} x_i}$$

is the maximum likelihood estimate for λ.

(b) Show that this MLE is biased. Hint: $f(x) = 1/x$ is a strictly convex function whenever $x > 0$. **Ans.** Since $f(x) = 1/x$ is a strictly convex function, and since $\mathbb{E}[\hat{\theta}] = \lambda$, then

$$\mathbb{E}[1/\hat{\theta}] > 1/\mathbb{E}[\hat{\theta}] = 1/(1/\lambda) = \lambda.$$

Therefore, $\mathbb{E}[1/\hat{\theta}] \neq \lambda$.