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?

   (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

   (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.

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. $\mathbf{Pr}$(current word = `rabbit`|previous word = `the`)

   ii. $\mathbf{Pr}$(previous word = `a`|current word = `rabbit`)

   iii. $\mathbf{Pr}$(previous word = `the`|current word = `rabbit`)

   iv. $\mathbf{Pr}$(previous word = `the` or `a`|current word = `rabbit`)

   v. Is the Naive Bayes assumption valid here, if previos word is used as a feature to predict future word?

   (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’`

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 $0\log_2(0) = 0$.)
- 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.
- 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?

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?

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

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

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

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

   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).

   (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.

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

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

then

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

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.

   (b) Show that this MLE is biased. Hint: $f(x) = 1/x$ is a strictly convex function whenever $x > 0$.