Homework 1 (V2 - Corrected Sept. 18)

Deadline: Thursday, Sept. 30, at 11:59pm.

Submission: You need to submit one file through Quercus with our answers to Questions 1, 2, 3, and 4, as well as code and outputs requested for Question 1. It should be a PDF file titled `hw1_writeup.pdf`. You can produce the file however you like (e.g. LaTeX, Microsoft Word, scanner), as long as it is readable.

Neatness Point: One point will be given for neatness. You will receive this point as long as we don't have a hard time reading your solutions or understanding the structure of your code.

Late Submission: 10% of the total possible marks will be deducted for each day late, up to a maximum of 3 days. After that, no submissions will be accepted.

Computing: To install Python and required libraries, see the instructions on the course web page.

Homeworks are to be done alone or in pairs. See the Course Information handout[1] for detailed policies.

1. **[5pts] Classification with Nearest Neighbours.** In this question, you will use the `scikit-learn`'s KNN classifer to classify real vs. fake news headlines. The aim of this question is for you to read the `scikit-learn` API and get comfortable with training/validation splits.

   We will use a dataset of 1298 "fake news" headlines (which mostly include headlines of articles classified as biased, etc.) and 1968 "real" news headlines, where the "fake news" headlines are from https://www.kaggle.com/mrisdal/fake-news/data and "real news" headlines are from https://www.kaggle.com/therohk/million-headlines. The data were cleaned by removing words from titles not part of the headlines, removing special characters and restricting real news headlines after October 2016 using the word "trump". The cleaned data and starter code are available as `clean_real.txt` and `clean_fake.txt` in `hw1_starter.zip` on the course webpage. It is expected that you use these cleaned data sources for this assignment.

   We are providing starter code for this assignment. To run the code you simply need to to run `hw1.py` using your favourite Python interpreter. To make the code correct, you will need to fill in the body of two functions. If you do this correctly, the code should run and output something that looks like the following:

   ```
   Selected K: 10
   Test Acc: 0.5
   ```

   Before you implement anything, or if you implement it incorrectly, the code may raise an Exception. This is expected behaviour.

   You will build a KNN classifier to classify real vs. fake news headlines. Instead of coding the KNN yourself, you will do what we normally do in practice — use an existing implementation. You should use the `KNeighborsClassifier` included in `sklearn`. Note that figuring out

[1] https://www.cs.toronto.edu/~cmaddis/courses/sta314_f21/sta314_f21_syllabus.pdf

how to use this implementation, its corresponding attributes and methods is a part of the assignment.

(a) **[2pts]** Complete the function `process_data`. It should do the following.

- First, split the entire dataset randomly into 70% training, 15% validation, and 15% test examples using `train_test_split` function (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html). You can use the stratify option to keep the label proportions the same in the split.
- Then, preprocess the data using a `CountVectorizer` (https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html#sklearn.feature_extraction.text.CountVectorizer). You will need to understand what this preprocessing does, so you should play around with it. Also, the `CountVectorizer` should be fit only on the training set.

In your writeup, report the function `process_data` that you wrote.

(b) **[2pts]** Complete the function `select_knn_model` that selects a k-NN classifer using a training set and a validation set to classify between real vs. fake news. This function should do the following.

- Iterate over k values between 1 to 20.
- For each k value, fit a `KNeighborsClassifier` to the training set, leaving other arguments at their default values.
- Measure the validation accuracy.
- Return the best choice of k and the corresponding model that has been fit to the training set with that value of k.

In your writeup, report the function `select_knn_model` that you wrote, as well as the output of the `hw1.py` script.

(c) **[1pts]** Repeat part (b), passing argument `metric='cosine'` to the `KNeighborsClassifier`. You should observe an improvement in accuracy. How does `metric='cosine'` compute the distance between data points, and why might this perform better than the Euclidean metric (default) here? *Hint: consider the dataset* `['cat', 'bulldozer', 'cat cat cat']`.

2. **[2pts] Nearest Neighbors and the Curse of Dimensionality** In this question, you will verify the claim from lecture that "most" points in a high-dimensional space are far away from each other, and also approximately the same distance. Consider two independent univariate random variables X and Y sampled uniformly from the unit interval $[0, 1]$. Define the squared distance $Z = (X - Y)^2$.

(a) **[2pts]** Suppose we sample two points independently from a unit cube in d dimensions. Observe that each coordinate is sampled independently from $[0, 1]$, i.e. we can view this as sampling random variables $X_1, \ldots, X_d, Y_1, \ldots, Y_d$ independently from $[0, 1]$. The squared Euclidean distance can be written as $R = Z_1 + \cdots + Z_d$, where $Z_i = (X_i - Y_i)^2$. Using the properties of expectation and variance, determine $\mathbb{E}[R]$ and $\mathrm{Var}[R]$. You may give your answer in terms of the dimension d, and $\mathbb{E}[Z]$ and $\mathrm{Var}[Z]$ (the random variable defined above).

(b) **[0pts]** This question is for your own benefit, not to be handed in. Based on your answer to part (a), compare the mean and standard deviation of R to the maximum possible

squared Euclidean distance (i.e. the distance between opposite corners of the cube). Why does this support the claim that in high dimensions, "most points are far away, and approximately the same distance"?

3. **[2pts] Benefit of Averaging.** Consider m predictors $h_1, \ldots, h_m$, each of which accepts an input x and produces an output y, i.e., $y_i = h_i(x)$. Consider the squared error loss function $L(y,t) = \frac{1}{2}(y-t)^2$. Show that the loss of the average estimator

$$\bar{h}(x) = \frac{1}{m}\sum_{i=1}^{m} h_i(x),$$

is smaller than the average loss of the estimators. That is, for any x and t, we have

$$L(\bar{h}(x), t) \leq \frac{1}{m}\sum_{i=1}^{m} L(h_i(x), t).$$

Hint: Use the fact that for any discrete random variable D with finitely many states, we have

$$\mathbb{E}[D^2] - \mathbb{E}[D]^2 = \mathrm{Var}[D] \geq 0. \tag{0.1}$$

4. **[5pts] Optimal predictors for the 0-1 loss.** In this question, we will study the 0-1 loss in more detail, and derive the optimal predictor for classification. We will consider a simple binary data problem. The input $X \in \{0,1\}$ and label $T \in \{0,1\}$ are binary random variables, and the set of predictors that we consider are the functions $y : \{0,1\} \to \{0,1\}$. Recall the 0-1 loss when predicting t with $y(x)$,

$$L_{0-1}(y(x), t) = \begin{cases} 0 & \text{if } y(x) = t \\ 1 & \text{if } y(x) \neq t \end{cases}, \tag{0.2}$$

and recall the definition of the expected error,

$$\mathcal{R}[y] = \mathbb{E}[L_{0-1}(y(X), T)] = \sum_{t\in\{0,1\}}\sum_{x\in\{0,1\}} L_{0-1}(y(x), t)P(X = x, T = t), \tag{0.3}$$

where $p(x,t)$ is some data generating distribution. We use the following short hand.

$$p(x,t) := P(X = x, T = t) \tag{0.4}$$

$$p(t|x) := P(T = t|X = x) \tag{0.5}$$

(a) **[1pt]** Assuming that $p(0|x) = p(1|x) = 1/2$ for all $x \in \{0,1\}$, prove that *all* predictors $y : \{0,1\} \to \{0,1\}$ achieve

$$\mathcal{R}[y] = 1/2 \tag{0.6}$$

(b) **[3pt]** Assuming that $p(0|x) \neq p(1|x)$ for all $x \in \{0,1\}$, prove that

$$y^\star(x) = \arg\max_{t\in\{0,1\}} p(t|x) \tag{0.7}$$

is the unique optimal predictor. In other words, show that

$$\mathcal{R}[y^\star] \leq \mathcal{R}[y] \tag{0.8}$$

with equality only if $y^\star(x) = y(x)$ for all $x \in \{0,1\}$.

(c) **[1pt]** Give an example of a joint distribution $p(x,t)$ on $\{0,1\}^2$ such that there are *exactly* two *distinct* optimal predictors y_1 and y_2.