CPSC 340 Assignment 1 (due Friday January 20 at 11:55pm)

Commentary on Assignment 1: CPSC 340 is tough because it combines knowledge and skills across several disciplines. To succeed in the course, you will need to know or very quickly get up to speed on:

- Math to the level of the course prerequisites: linear algebra, multivariable calculus, some probability.
- Basic Python programming, including NumPy and plotting with matplotlib.
- Statistics, algorithms, and data structures to the level of the course prerequisites.
- Some basic LaTeX skills so that you can typeset equations and submit your assignments.

The purpose of this assignment is to make sure you are prepared for this course. We anticipate that each of you will have different strengths and weaknesses, so don't be worried if you struggle with *some* aspects of the assignment. But if you find this assignment to be very difficult overall, that is a sign that you may not be prepared to take CPSC 340 at this time. Future assignments will be more difficult than this one (and probably around the same length).

Questions 1-4 are on review material, that we expect you to know coming into the course. The rest is related to the first few lectures. We use blue to highlight the deliverables that you must answer/do/submit with the assignment.

!!! IMPORTANT !!! Before proceeding, please carefully read the homework instructions posted on Piazza. You may receive a 50% deduction on the assignment if you don't follow these instructions.

A note on the provided code: in the `code` directory we provide you with a file called `main.py`. This file, when run with different arguments, runs the code for different parts of the assignment. For example,

```
python main.py 6.2
```

runs the code for Question 6.2. At present, this should do nothing (throws a `NotImplementedError`), because the code for Question 6.2 still needs to be written (by you). But we do provide some of the bits and pieces to save you time, so that you can focus on the machine learning aspects. For example, you'll see that the provided code already loads the datasets for you. The file `utils.py` contains some helper functions. You don't need to read or modify the code in there. To complete your assignment, you will need to modify `grads.py`, `main.py`, `decision_stump.py` and `simple_decision.py` (which you'll need to create).

Basic Information

1. Name:

   Answer: Use the "ans" command to add answers if you like.

2. Student ID:

1 Linear Algebra Review

For these questions you may find it helpful to review these notes on linear algebra:
`http://www.cs.ubc.ca/~schmidtm/Documents/2009_Notes_LinearAlgebra.pdf`

1.1 Basic Operations

Use the definitions below,

$$\alpha = 2, \quad x = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}, \quad A = \begin{bmatrix} 3 & 2 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{bmatrix}, \quad I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

and use x_i to denote element i of vector x. Evaluate the following expressions, showing at least one intermediate step of work:

1. $\|x\|$ (Euclidean norm of x).
2. $\alpha(x + y)$ (vector addition and scalar multiplication).
3. $x^T y = \sum_{i=1}^{n} x_i y_i$ (inner product).
4. xy^T (outer product).
5. Ax (matrix-vector multiplication).
6. $x^T Ax$ (quadratic form).
7. Solve for a vector v that satisfies $(I_3 - xx^T)v = y$ (linear system).

1.2 Matrix Algebra Rules

Assume that $\{x, y, z\}$ are $n \times 1$ column vectors and $\{A, B, C\}$ are $n \times n$ real-valued matrices, 0 is the zero matrix of appropriate size, and I is the identity matrix of appropriate size. State whether each of the below is true in general (you do not need to show your work).

1. $x^T x = \|x\|^2$.
2. $x^T x = xx^T$.
3. $(x - y)^T (y - x) = \|x\|^2 - 2x^\top y + \|y\|^2$.
4. $AB = BA$.
5. $A(B + C) = AB + AC$.
6. $(AB)^T = A^T B^T$.
7. $x^T Ay = y^T A^T x$.
8. $A^n = (A^n)^T$ for any non-negative integer n if A is symmetric.
9. $A^\top A = I$ if the columns of A are orthonormal.

2 Probability Review

For these questions you may find it helpful to review these notes on probability:
`http://www.cs.ubc.ca/~schmidtm/Courses/Notes/probability.pdf`
And here are some slides giving visual representations of the ideas as well as some simple examples:
`http://www.cs.ubc.ca/~schmidtm/Courses/Notes/probabilitySlides.pdf`

2.1 Rules of probability

Answer the following questions. You do not need to show your work.

1. Consider two events A and B such that $\Pr(A, B) = 0$ (they are mutually exclusive). If $\Pr(A) = 0.4$ and $\Pr(A \cup B) = 0.95$, what is $\Pr(B)$? Note: $p(A, B)$ means "probability of A and B" while $p(A \cup B)$ means "probability of A or B". It may be helpful to draw a Venn diagram.
2. Instead of assuming that A and B are mutually exclusive ($\Pr(A, B) = 0$), what is the answer to the previous question if we assume that A and B are independent?
3. You are offered the opportunity to play the following game: first your opponent rolls a 4-sided dice and records the outcome r_1. Then you roll a $(5 - r_1)$-sided dice and record the outcome r_2. Your payout is $r_1 + r_2 - 1$ dollars. You can enter the game either before or after your opponent's turn.
   - If you enter *after* your opponent's turn you know r_1. What is a fair price for a ticket in this case, i.e., what is the expected payout as a function of r_1?
   - If you enter *before* your opponent's turn you do not know r_1. What is the expected payout now?

2.2 Bayes Rule and Conditional Probability

Answer the following questions. You do not need to show your work.

Suppose a drug test produces a positive result with probability 0.95 for drug users, $P(T = 1|D = 1) = 0.95$. It also produces a negative result with probability 0.99 for non-drug users, $P(T = 0|D = 0) = 0.99$. The probability that a random person uses the drug is 0.0001, so $P(D = 1) = 0.0001$.

1. What is the probability that a random person would test positive, $P(T = 1)$?
2. In the above, do most of these positive tests come from true positives or from false positives?
3. What is the probability that a random person who tests positive is a user, $P(D = 1|T = 1)$?
4. Suppose you have given this test to a random person and it came back positive, are they likely to be a drug user?
5. What is one factor you could change to make this a more useful test?

3 Calculus Review

For these questions you may find it helpful to review these notes on calculus:
`http://www.cs.ubc.ca/~schmidtm/Courses/Notes/calculus.pdf`

3.1 One-variable derivatives

Answer the following questions. You do not need to show your work.

1. Find the derivative of the function $f(x) = 3x^2 - 2x + 5$.
2. Find the derivative of the function $f(x) = x^2 \cdot \exp(x)$.
3. Let $p(x) = \frac{1}{1+\exp(-x)}$ for $x \in \mathbb{R}$. Compute the derivative of the function $f(x) = x - \log(p(x))$ and simplify it by using the function $p(x)$.

Note that in this course we will use $\log(x)$ to mean the "natural" logarithm of x, so that $\log(\exp(1)) = 1$. Also, observe that $p(x) = 1 - p(-x)$ for the final part.

3.2 Multi-variable derivatives

Compute the gradient $\nabla f(x)$ of each of the following functions. You do not need to show your work.

1. $f(x) = x_1^2 + \exp(x_2)$ where $x \in \mathbb{R}^2$.
2. $f(x) = \exp(x_1 + x_2 x_3)$ where $x \in \mathbb{R}^3$.
3. $f(x) = a^T x$ where $x \in \mathbb{R}^2$ and $a \in \mathbb{R}^2$.
4. $f(x) = x^\top A x$ where $A = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$ and $x \in \mathbb{R}^2$.
5. $f(x) = \frac{1}{2}\|x\|^2$ where $x \in \mathbb{R}^d$.

Hint: it is helpful to write out the linear algebra expressions in terms of summations.

3.3 Derivatives of code

Your repository contains a file named `grads.py` which defines several Python functions that take in an input variable x, which we assume to be a 1-d array (in math terms, a vector). It also includes (blank) functions that return the corresponding gradients. For each function, write code that computes the gradient of the function in Python. You should do this directly in `grads.py`; no need to make a fresh copy of the file. When finished, you can run `python main.py 3.4` to test out your code. Include this code following the instructions in the submission instructions.

Hint: it's probably easiest to first understand on paper what the code is doing, then compute the gradient, and then translate this gradient back into code.

Note: do not worry about the distinction between row vectors and column vectors here. For example, if the correct answer is a vector of length 5, we'll accept numpy arrays of shape `(5,)` (a 1-d array) or `(5,1)` (a column vector) or `(1,5)` (a row vector). In future assignments we will start to be more careful about this.

Warning: Python uses whitespace instead of curly braces to delimit blocks of code. Some people use tabs and other people use spaces. My text editor (Atom) inserts 4 spaces (rather than tabs) when I press the Tab key, so the file `grads.py` is indented in this manner (and indeed, this is standard Python style that you should probably also follow). If your text editor inserts tabs, Python will complain and you might get mysterious errors... this is one of the most annoying aspects of Python, especially when starting out. So, please be aware of this issue! And if in doubt you can just manually indent with 4 spaces, or convert everything to tabs. For more information see `https://www.youtube.com/watch?v=SsoOG6ZeyUI`.

4 Algorithms and Data Structures Review

For these questions you may find it helpful to review these notes on big-O notation:
`http://www.cs.ubc.ca/~schmidtm/Courses/Notes/bigO.pdf`

4.1 Trees

Answer the following questions. You do not need to show your work.

1. What is the maximum number of *leaves* you could have in a binary tree of depth l?
2. What is the maximum number of *internal nodes* (excluding leaves) you could have in a binary tree of depth l?

Note: we'll use the standard convention that the leaves are not included in the depth, so a tree with depth 1 has 3 nodes with 2 leaves.

4.2 Common Runtimes

Answer the following questions using big-O notation. You do not need to show your work.

1. What is the cost of running the mergesort algorithm to sort a list of n numbers?
2. What is the cost of finding the third-largest element of an unsorted list of n numbers?
3. What is the cost of finding the smallest element greater than 0 in a *sorted* list with n numbers?
4. What is the cost of finding the value associated with a key in a hash table with n numbers? (Assume the values and keys are both scalars.)
5. What is the cost of computing the matrix-vector product Ax when A is $n \times d$ and x is $d \times 1$?
6. What is the cost of computing the quadratic form $x^T Ax$ when A is $d \times d$ and x is $d \times 1$?
7. How does the answer to the previous question change if A has only z non-zeroes? (You can assume $z \geq d$)

4.3 Running Times of Code

Your repository contains a file named `bigO.py`, which defines several functions that take an integer argument N. For each function, state the running time as a function of N, using big-O notation.

5 Data Exploration

Your repository contains the file `fluTrends.csv`, which contains estimates of the influenza-like illness percentage over 52 weeks on 2005-06 by Google Flu Trends. Your `main.py` loads this data for you and stores it in a pandas DataFrame `X`, where each row corresponds to a week and each column corresponds to a different region. If desired, you can convert from a DataFrame to a raw numpy array with `X.values()`.

5.1 Summary Statistics

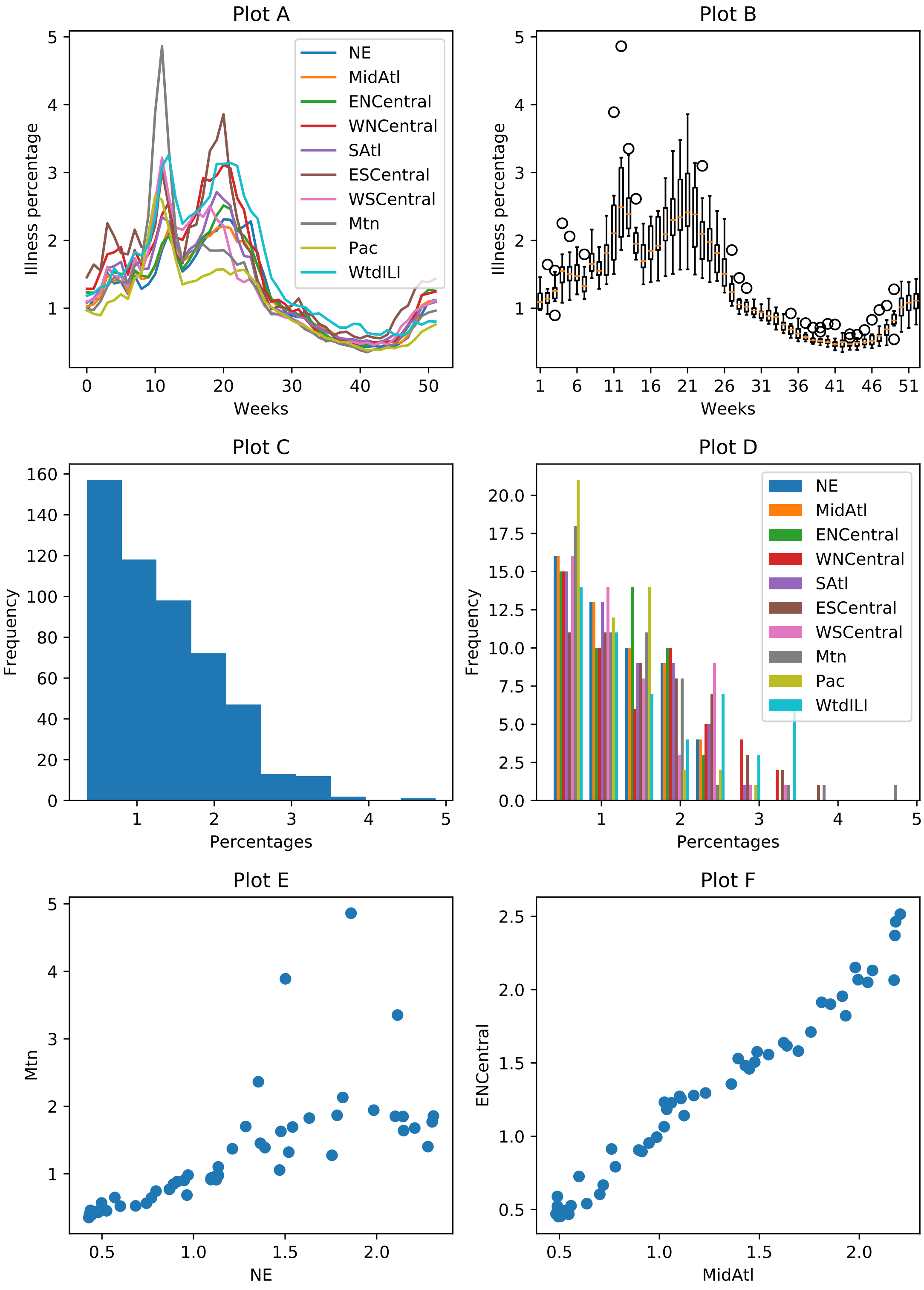
Report the following statistics:

1. The minimum, maximum, mean, median, and mode of all values across the dataset.
2. The 5%, 25%, 50%, 75%, and 95% quantiles of all values across the dataset.
3. The names of the regions with the highest and lowest means, and the highest and lowest variances.

In light of the above, is the mode a reliable estimate of the most "common" value? Describe another way we could give a meaningful "mode" measurement for this (continuous) data. Note: the function `utils.mode()` will compute the mode value of an array for you.

5.2 Data Visualization

Consider the figure below.



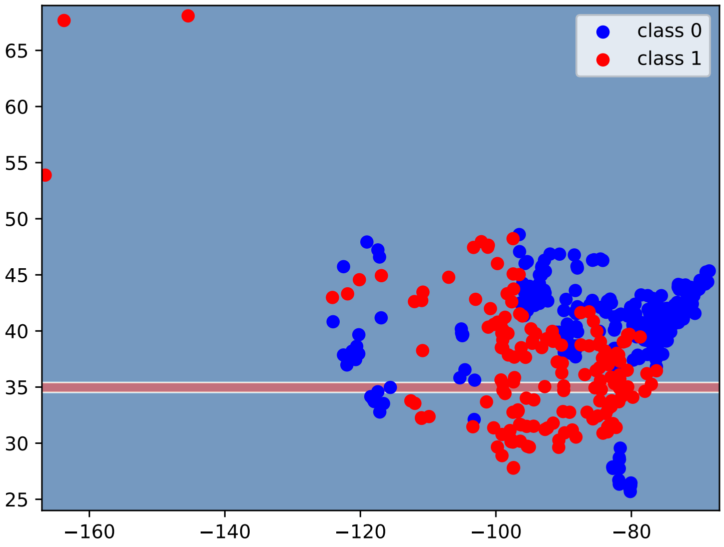
The figure contains the following plots, in a shuffled order:

1. A single histogram showing the distribution of *each* column in X.
2. A histogram showing the distribution of each the values in the matrix X.
3. A boxplot grouping data by weeks, showing the distribution across regions for each week.
4. A plot showing the illness percentages over time.
5. A scatterplot between the two regions with highest correlation.
6. A scatterplot between the two regions with lowest correlation.

Match the plots (labeled A-F) with the descriptions above (labeled 1-6), with an extremely brief (a few words is fine) explanation for each decision.

6 Decision Trees

If you run `python main.py 6`, it will load a dataset containing longitude and latitude data for 400 cities in the US, along with a class label indicating whether they were a "red" state or a "blue" state in the 2012 election.[1] Specifically, the first column of the variable X contains the longitude and the second variable contains the latitude, while the variable y is set to 0 for blue states and 1 for red states. After it loads the data, it plots the data and then fits two simple classifiers: a classifier that always predicts the most common label (0 in this case) and a decision stump that discretizes the features (by rounding to the nearest integer) and then finds the best equality-based rule (i.e., check if a feature is equal to some value). It reports the training error with these two classifiers, then plots the decision areas made by the decision stump. The plot is shown below:



As you can see, it is just checking whether the latitude equals 35 and, if so, predicting red (Republican). This is not a very good classifier.

6.1 Splitting rule

Is there a particular type of features for which it makes sense to use an equality-based splitting rule rather than the threshold-based splits we discussed in class?

[1] The cities data was sampled from `http://simplemaps.com/static/demos/resources/us-cities/cities.csv`. The election information was collected from Wikipedia.

6.2 Decision Stump Implementation

The file decision_stump.py contains the class DecisionStumpEquality which finds the best decision stump using the equality rule and then makes predictions using that rule. Instead of discretizing the data and using a rule based on testing an equality for a single feature, we want to check whether a feature is above or below a threshold and split the data accordingly (this is a more sane approach, which we discussed in class). Create a DecisionStumpErrorRate class to do this, and report the updated error you obtain by using inequalities instead of discretizing and testing equality. Submit your class definition code as a screenshot or using the lstlisting environment. Also submit the generated figure of the classification boundary.

Hint: you may want to start by copy/pasting the contents DecisionStumpEquality and then make modifications from there.

6.3 Decision Stump Info Gain Implementation

In decision_stump.py, create a DecisionStumpInfoGain class that fits using the information gain criterion discussed in lecture. Report the updated error you obtain. Submit your class definition code as a screenshot or using the lstlisting environment. Submit the classification boundary figure.

Notice how the error rate changed. Are you surprised? If so, hang on until the end of this question!

Note: even though this data set only has 2 classes (red and blue), your implementation should work for any number of classes, just like DecisionStumpEquality and DecisionStumpErrorRate.

Hint: take a look at the documentation for np.bincount, at https://docs.scipy.org/doc/numpy/reference/generated/numpy.bincount.html. The minlength argument comes in handy here to deal with a tricky corner case: when you consider a split, you might not have any cases of a certain class, like class 1, going to one side of the split. Thus, when you call np.bincount, you'll get a shorter array by default, which is not what you want. Setting minlength to the number of classes solves this problem.

6.4 Hard-coded Decision Trees

Once your `DecisionStumpInfoGain` class is finished, running `python main.py 6.4` will fit a decision tree of depth 2 to the same dataset (which results in a lower training error). Look at how the decision tree is stored and how the (recursive) `predict` function works. Using the splits from the fitted depth-2 decision tree, write a hard-coded version of the `predict` function that classifies one example using simple if/else statements (see the Decision Trees lecture). Submit this code as a plain text, as a screenshot or using the `lstlisting` environment.

Note: this code should implement the specific, fixed decision tree which was learned after calling `fit` on this particular data set. It does not need to be a learnable model. You should just hard-code the split values directly into the code. Only the `predict` function is needed.

Hint: if you plot the decision boundary you can do a quick visual check to see if your code is consistent with the plot.

6.5 Decision Tree Training Error

Running `python main.py 6.5` fits decision trees of different depths using the following different implementations:

1. A decision tree using `DecisionStumpErrorRate`
2. A decision tree using `DecisionStumpInfoGain`
3. The `DecisionTreeClassifier` from the popular Python ML library *scikit-learn*

Run the code and look at the figure. Describe what you observe. Can you explain the results? Why is approach (1) so disappointing? Also, submit a classification boundary plot of the model with the lowest training error.

Note: we set the `random_state` because sklearn's `DecisionTreeClassifier` is non-deterministic. This is probably because it breaks ties randomly.

Note: the code also prints out the amount of time spent. You'll notice that sklearn's implementation is substantially faster. This is because our implementation is based on the $O(n^2d)$ decision stump learning algorithm and sklearn's implementation presumably uses the faster $O(nd \log n)$ decision stump learning algorithm that we discussed in lecture.

6.6 Comparing implementations

In the previous section you compared different implementations of a machine learning algorithm. Let's say that two approaches produce the exact same curve of classification error rate vs. tree depth. Does this conclusively demonstrate that the two implementations are the same? If so, why? If not, what other experiment might you perform to build confidence that the implementations are probably equivalent?

HAVE YOU DOUBLE CHECKED THAT YOU'RE FOLLOWING ALL THE ASSIGNMENT SUBMISSION INSTRUCTIONS POSTED ON PIAZZA???