CAP 4611 Assignment 1

Broc Weselmann (br142931)

Commentary on Assignment 1: CAP 4611 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:

- Basic Python programming, including NumPy and plotting with matplotlib.
- Math to the level of the course prerequisites: linear algebra, multivariable calculus, some probability.
- 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.

This assignment will help you assess whether 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 warning sign that you may not be prepared to take CAP 4611 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 new CAP 4611 material from the first few lectures.

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, 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).

Instructions [5 points]

The above points are allocated for following the submission instructions. Both you and the person marking your assignment will be much happier if you do. The instructions is poseted on the course website and webcourses, and you should read them carefully. Here is a direct link to the instructions.

We use blue to highlight the deliverables that you must answer/do/submit with the assignment.

1 Linear Algebra Review [17 points]

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 [7 points]

Use the definitions below,

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

and use x_i to denote element i of vector x. Evaluate the following expressions (you do not need to show your work).

1. $\sum_{i=1}^{n} x_i y_i$ (inner product).

   Answer: 14

2. $\sum_{i=1}^{n} x_i z_i$ (inner product between orthogonal vectors).

   Answer: 0

3. $\alpha(x + z)$ (vector addition and scalar multiplication)

   Answer: $\begin{bmatrix} 6 \\ 10 \\ 14 \end{bmatrix}$

4. $x^T z + \|x\|$ (inner product in matrix notation and Euclidean norm of x).

   Answer: $\sqrt{5}$

5. Ax (matrix-vector multiplication).

   Answer: $\begin{bmatrix} 6 \\ 5 \\ 7 \end{bmatrix}$

6. $x^T A x$ (quadratic form).

   Answer: 19

7. $A^T A$ (matrix tranpose and matrix multiplication).

   Answer: $\begin{bmatrix} 11 & 10 & 10 \\ 10 & 14 & 10 \\ 10 & 10 & 14 \end{bmatrix}$

1.2 Matrix Algebra Rules [10 points]

Assume that $\{x, y, z\}$ are $n \times 1$ column vectors, $\{A, B, C\}$ 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. $x^T y = \sum_{i=1}^{n} x_i y_i$.

   Answer: True

2. $x^T x = \|x\|^2$.

   Answer: True

3. $x^T x = x x^T$.

   Answer: True

4. $(x-y)^T(x-y) = \|x\|^2 - 2x^Ty + \|y\|^2$.

   Answer: True

5. $AB = BA$.

   Answer: True

6. $A^T(B + IC) = A^TB + A^TC$.

   Answer: False

7. $(A + BC)^T = A^T + B^TC^T$.

   Answer: True

8. $x^TAy = y^TA^Tx$.

   Answer: True

9. $A^TA = AA^T$ if A is a symmetric matrix.

   Answer: True

10. $A^TA = 0$ if the columns of A are orthonormal.

    Answer: True

2 Probability Review [16 points]

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 [6 points]

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

1. You are offered the opportunity to play the following game: your opponent rolls 2 regular 6-sided dice. If the difference between the two rolls is at least 3, you win $30. Otherwise, you get nothing. What is a fair price for a ticket to play this game once? In other words, what is the expected value of playing the game?

   Answer: $10

2. Consider two events A and B such that $\Pr(A \cap B) = 0$ (they are mutually exclusive). If $\Pr(A) = 0.4$ and $\Pr(A \cup B) = 0.95$, what is $\Pr(B)$? Note: $\Pr(A \cap B)$ 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.

   Answer: 0.55

3. Instead of assuming that A and B are mutually exclusive ($\Pr(A \cap B) = 0$), what is the answer to the previous question if we assume that A and B are independent?

   Answer: 0.55 - p(A, B)

2.2 Bayes Rule and Conditional Probability [10 points]

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

Suppose a drug test produces a positive result with probability 0.97 for drug users, $P(T = 1 \mid D = 1) = 0.97$. It also produces a negative result with probability 0.99 for non-drug users, $P(T = 0 \mid D = 0) = 0.99$. 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, $P(T = 1)$?

   Answer: 0.010096

2. In the above, do most of these positive tests come from true positives or from false positives?

   Answer: False positives.

3. What is the probability that a random person who tests positive is a user, $P(D = 1 \mid T = 1)$?

   Answer: 0.009608

4. Suppose you have given this test to a random person and it came back positive, are they likely to be a drug user?

   Answer: Yes, it is very likely they are a drug user.

5. Suppose you are the designer of this drug test. You may change how the test is conducted, which may influence factors like false positive rate, false negative rate, and the number of samples collected. What is one factor you could change to make this a more useful test?

   Answer: Increase the number of samples to reduce the possibility of false positives and negatives.

3 Calculus Review [23 points]

3.1 One-variable derivatives [8 points]

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

1. Find the derivative of the function $f(x) = 5x^3 - 2x + 5$.

   Answer: $15x^2 - 2$

2. Find the derivative of the function $f(x) = x(1 - x)$.

   Answer: $-2x + 1$

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 $p(x)$.

   Answer: $p(x)$

Remember that in this course we will $\log(x)$ 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 [5 points]

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

1. $f(x) = x_1^2 + \exp(x_1 + 3x_2)$ where $x \in \mathbb{R}^2$.

   Answer: $\begin{bmatrix} 2x_1 + exp(x_1 + 3x_2) \\ 3exp(x_1 + 3x_2) \end{bmatrix}$

2. $f(x) = \log\left(\sum_{i=1}^3 \exp(x_i)\right)$ where $x \in \mathbb{R}^3$ (simplify the gradient by defining $Z = \sum_{i=1}^3 \exp(x_i)$).

   Answer: $\frac{1}{Z}\begin{bmatrix} exp(x_1) \\ exp(x_2) \\ exp(x_3) \end{bmatrix}$

3. $f(x) = a^T x + b$ where $x \in \mathbb{R}^3$ and $a \in \mathbb{R}^3$ and $b \in \mathbb{R}$.

   Answer: a

4. $f(x) = \frac{1}{2}x^\top Ax$ where $A = \begin{bmatrix} 4 & -1 \\ -1 & 4 \end{bmatrix}$ and $x \in \mathbb{R}^2$.

   Answer: $\begin{bmatrix} 4x_1 - x_2 \\ -x_1 + 4x_2 \end{bmatrix}$

5. $f(x) = \frac{1}{2}\|x\|^2$ where $x \in \mathbb{R}^d$.

   Answer: x

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

3.3 Optimization [6 points]

Find the following quantities. You do not need to show your work.

1. $\min 3x^2 - 2x + 5$, or, in words, the minimum value of the function $f(x) = 3x^2 - 2x + 5$ for $x \in \mathbb{R}$.

   Answer: $\frac{14}{3}$

2. $\max_{x \in [0,1]} x(1-x)$

   Answer: $\frac{1}{4}$

3. $\min_{x \in [0,1]} x(1-x)$

   Answer: 0

4. $\arg\max_{x \in [0,1]} x(1-x)$

   Answer: $\frac{1}{2}$

5. $\min_{x \in [0,1]^2} x_1^2 + \exp(x_2)$ – in other words $x_1 \in [0,1]$ and $x_2 \in [0,1]$

   Answer: $(0, 0)$

6. $\arg\min_{x \in [0,1]^2} x_1^2 + \exp(x_2)$ where $x \in [0,1]^2$.

   Answer: $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Note: the notation $x \in [0,1]$ means "x is in the interval $[0,1]$", or, also equivalently, $0 \leq x \leq 1$.

Note: the notation "$\max f(x)$" means "the value of $f(x)$ where $f(x)$ is maximized", whereas "$\arg\max f(x)$" means "the value of x such that $f(x)$ is maximized". Likewise for min and arg min. For example, the min of the function $f(x) = (x-1)^2$ is 0 because the smallest possible value is $f(x) = 0$, whereas the arg min is 1 because this smallest value occurs at $x = 1$. The min is always a scalar but the arg min is a value of x, so it's a vector if x is vector-valued.

3.4 Derivatives of code [4 points]

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.

Answer:

```
def foo_grad(x):
  lam = 4
  return lam * (x ** (lam - 1))
```

```
def bar_grad(x):
  zeros = np.where(x == 0)[0]
  num_zeros = len(zeros)

  if num_zeros > 1:
      # All zeros
      return np.zeros_like(x, dtype=float)
  elif num_zeros == 1:
      k = zeros[0]
      grad = np.zeros_like(x, dtype=float)
      temp_x = x.copy()
      # Where 0 is
      temp_x[k] = 1
      grad[k] = np.prod(temp_x)
      return grad
  else:
      product = np.prod(x)
      return product / x
```

4 Algorithms and Data Structures Review [11 points]

4.1 Trees [2 points]

Answer the following questions. You do not need to show your work. We'll define "depth" as the maximum number of edges you need to traverse to get from the root of the tree to a leaf of the tree. In other words, if you're thinking about nodes, then the leaves are not included in the depth, so a complete tree with depth 1 has 3 nodes with 2 leaves.

1. What is the minimum depth of a binary tree with 128 leaf nodes?

   Answer: 7

2. What is the minimum depth of binary tree with 128 nodes (including leaves and all other nodes)?

   Answer: 7

4.2 Common Runtimes [5 points]

Answer the following questions using big-O notation You do not need to show your work. Here, the word "list" means e.g. a Python `list` – an array, not a linked list.

1. What is the cost of finding the largest number in an unsorted list of n numbers?

   Answer: $O(n)$

2. What is the cost of finding the smallest element greater than 0 in a *sorted* list with n numbers.

   Answer: $O(n)$

3. 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.)

   Answer: $O(1)$

4. What is the cost of computing the inner product $a^T x$, where a is $d \times 1$ and x is $d \times 1$?

   Answer: $O(n^2)$

5. What is the cost of computing the quadratic form $x^T A x$ when A is $d \times d$ and x is $d \times 1$?

   Answer: $O(n^2)$

4.3 Running times of code [4 points]

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.

Answer:

1. func1: $O(N)$
2. func2: $O(N)$
3. func3: $O(1000)$
4. func4: $O(N^2)$

5 Data Exploration [5 points]

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.

5.1 Summary Statistics [2 points]

Report the following statistics:

1. The minimum, maximum, mean, median, and mode of all values across the dataset. **Note:** A mode function is defined for you it `utils.py`.

   Answer:
   Minimum: 0.352
   Maximum: 4.862
   Mean: 1.3246250000000002
   Median: 1.1589999999999998
   Mode: 0.77

2. The 5%, 25%, 50%, 75%, and 95% quantiles of all values across the dataset.

   Answer:
   5%: 0.46495000000000003
   25%: 1.81325
   50%: 1.1589999999999998
   75%: 1.81325
   95%: 2.624049999999999

3. The names of the regions with the highest and lowest means, and the highest and lowest variances.

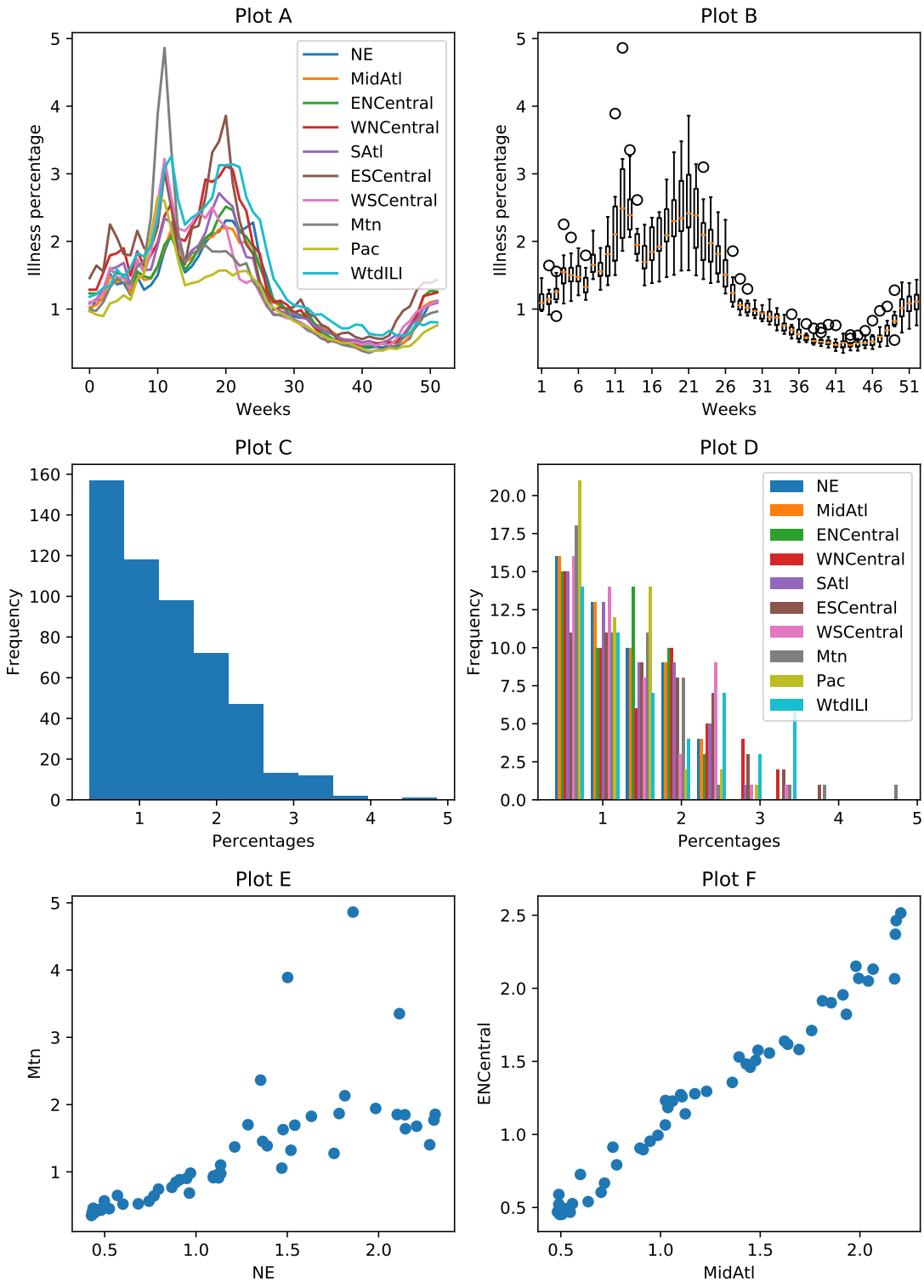
   Answer:
   Highest Mean: WtdILI
   Lowest Mean: Pac
   Highest Variance: Mtn
   Lowest Variance: Pac

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.

Answer: No, it isn't. We can see that the mode, 0.77, is below the 25% quartile, and so does not represent the majority of the data. The most 'common' value in this dataset would probably be closer to the median, the 50% quartile.

5.2 Data Visualization [3 points]

Consider the figure below.


Plot A
Illness percentage
Weeks
NE
MidAtl
ENCentral
WNCentral
SAtl
ESCentral
WSCentral
Mtn
Pac
WtdILI
Plot B
Illness percentage
Weeks
Plot C
Frequency
Percentages
Plot D
Frequency
Percentages
Plot E
Mtn
NE
Plot F
ENCentral
MidAtl

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

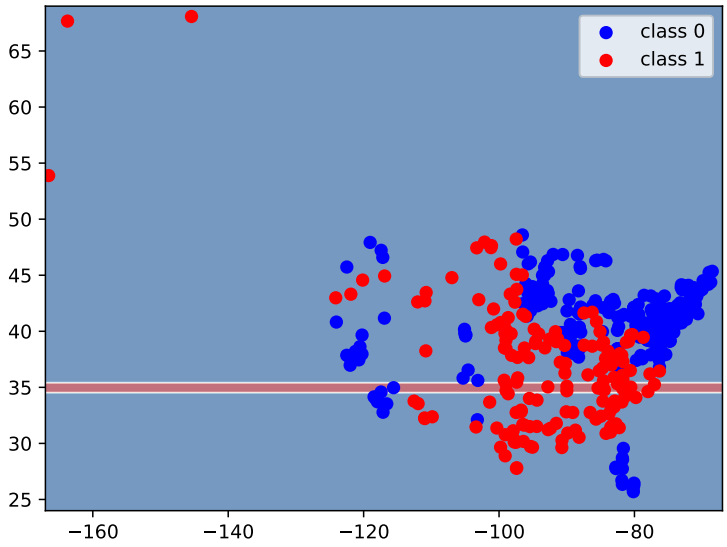
1. A single histogram showing the distribution of *each* column in X.
2. A histogram showing the distribution of each the values in the matrix X.
3. A boxplot grouping data by weeks, showing the distribution across regions for each week.
4. A plot showing the illness percentages over time.
5. A scatterplot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

Answer:
A - 4
This plot shows illness percentages over time, and is not a boxplot.
B - 3
This is the only boxplot.
C - 1
This is the only single histogram.
D - 2
This is the remaining histogram.
E - 6
This is a scatterplot, and is less correlated than the other scatterplot.
F - 5
This is a scatterplot, and is more correlated than the other scatterplot.

6 Decision Trees [23 points]

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 [1 points]

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?

Answer: Features that can be completely seperated from one another via a horizontal or vertical line. (Learning Decision Trees, slide 29)

6.2 Decision Stump Implementation [8 points]

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

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

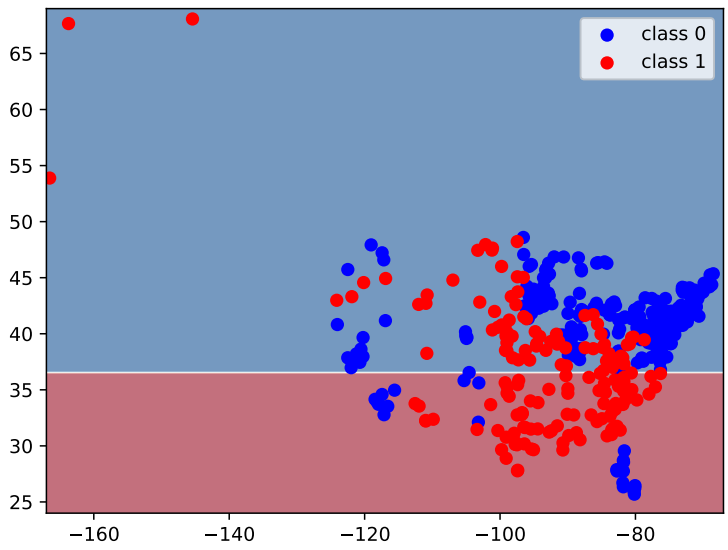
a `DecisionStumpErrorRate` class to do this, and report the updated error you obtain by using inequalities instead of discretizing and testing equality. See submission instructions for how to submit your code. Also submit the generated figure of the classification boundary.

Hint: you may want to start by copy/pasting the contents `DecisionStumpEquality` and then make modifications from there. Hint: A correct implementation will achieve an error in the neighbourhood of 0.250. Our reference implementation gets 0.253. Note: please keep the same variable names, as subsequent parts of this assignment rely on this!

Answer:

```
class DecisionStumpErrorRate:
  y_hat_yes = None
  y_hat_no = None
  j_best = None
  t_best = None

  t_range = 150

  def fit(self, X, y):
    n, d = X.shape

    # Get an array with the number of 0's, number of 1's, etc.
    count = np.bincount(y)

    # Get the index of the largest value in count.
    # Thus, y_mode is the mode (most popular value) of y
    y_mode = np.argmax(count)

    self.y_hat_yes = y_mode
    self.y_hat_no = None
    self.j_best = None
    self.t_best = None

    # If all the labels are the same, no need to split further
    if np.unique(y).size <= 1:
        return

    minError = np.sum(y != y_mode)

    # Loop over features looking for the best split
    for j in range(d):
        # For each threshold
        for t_test in range(-self.t_range, self.t_range, 1):
            # Choose value to equate to
            t = t_test

            # Find most likely class for each split
            is_almost_equal = np.round(X[:, j]) > t
            y_yes_mode = utils.mode(y[is_almost_equal])
            y_no_mode = utils.mode(y[~is_almost_equal])  # ~ is "logical not"

            # Make predictions
            y_pred = y_yes_mode * np.ones(n)
```

```
            y_pred[np.round(X[:, j]) <= t] = y_no_mode

            # Compute error
            errors = np.sum(y_pred != y)

            # Compare to minimum error so far
            if errors < minError:
                # This is the lowest error, store this value
                minError = errors
                self.j_best = j
                self.t_best = t
                self.y_hat_yes = y_yes_mode
                self.y_hat_no = y_no_mode

def predict(self, X):
    n, d = X.shape
    X = np.round(X)

    if self.j_best is None:
        return self.y_hat_yes * np.ones(n)

    y_hat = np.zeros(n)

    for i in range(n):
        if X[i, self.j_best] > self.t_best:
            y_hat[i] = self.y_hat_yes
        else:
            y_hat[i] = self.y_hat_no

    return y_hat
```



6.3 Decision Stump Info Gain Implementation [8 points]

In `decision_stump.py`, create a `DecisionStumpInfoGain` class that fits using the information gain criterion discussed in lecture. Report the updated error you obtain. See submission instructions for how to submit your code. Submit the classification boundary figure.

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

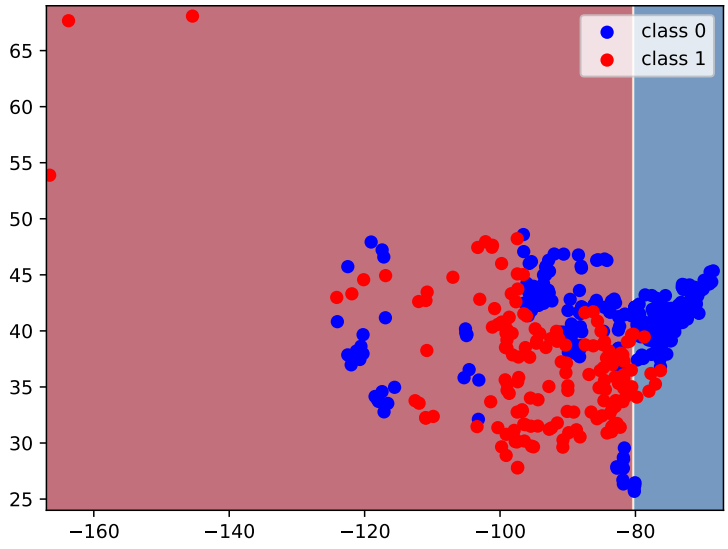
Note: even though this data set only has 2 classes (red and blue), your implementation should work for any number of classes, just like `DecisionStumpEquality` and `DecisionStumpErrorRate`.

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.

Answer:

```
class DecisionStumpInfoGain(DecisionStumpErrorRate):
  def fit(self, X, y):
    n, d = X.shape

    # Get an array with the number of 0's, number of 1's, etc.
    count = np.bincount(y)

    # Get the index of the largest value in count.
    # Thus, y_mode is the mode (most popular value) of y
    y_mode = np.argmax(count)

    self.y_hat_yes = y_mode
    self.y_hat_no = None
    self.j_best = None
    self.t_best = None

    # If all the labels are the same, no need to split further
    if np.unique(y).size <= 1:
        return

    minScore = entropy(np.bincount(y) / n)

    # Loop over features looking for the best split
    for j in range(d):
        # For each threshold
        for t_test in range(-self.t_range, self.t_range, 1):
            # Choose value to equate to
            t = t_test

            # Find most likely class for each split
            is_almost_equal = np.round(X[:, j]) > t

            # No None
            if not np.any(is_almost_equal) or not np.any(~is_almost_equal):
                continue
```

```
y_yes_mode = utils.mode(y[is_almost_equal])
y_no_mode = utils.mode(y[~is_almost_equal])  # ~ is "logical not"

# Score calculation
left_examples = np.bincount(y[is_almost_equal], minlength=np.max(y) + 1)
right_examples = np.bincount(
    y[~is_almost_equal], minlength=np.max(y) + 1
)

entropy_left = entropy(left_examples / len(y[is_almost_equal]))
entropy_right = entropy(right_examples / len(y[~is_almost_equal]))

weight_left = len(y[is_almost_equal]) / n
weight_right = len(y[~is_almost_equal]) / n

score = weight_left * entropy_left + weight_right * entropy_right

# Compare to minimum score so far
if score < minScore:
    # This is the lowest score, store this value
    minScore = score
    self.j_best = j
    self.t_best = t
    self.y_hat_yes = y_yes_mode
    self.y_hat_no = y_no_mode
```



6.4 Hard-coded Decision Trees [2 points]

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). See submission instructions for how to submit your code.

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.

Answer:

```
# where X is {lat : float, lon : float}
def hard_coded_predict(X):
  if X[0] > -81.0:
      return 0
  else:
      if X[1] > 39:
          return 0
      else:
          return 1
```

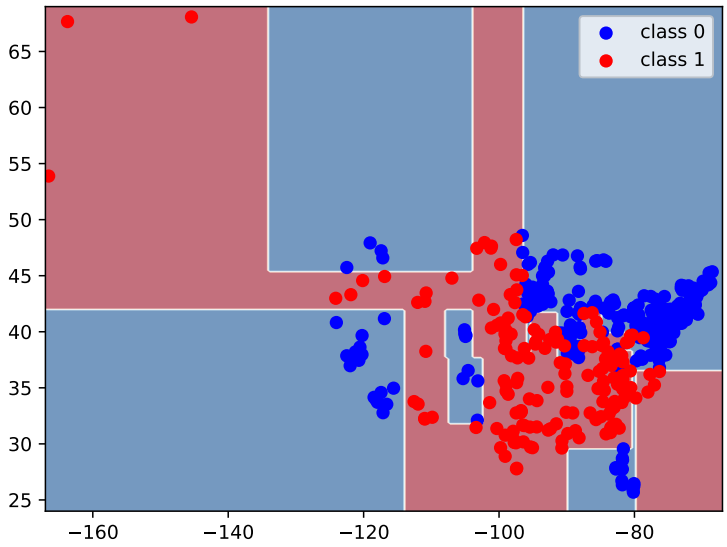
6.5 Decision Tree Training Error [2 points]

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.

Answer: I observe that both of our solutions, DecisionStumpErrorRate and DecisionStumpInfoGain, both 'plateau' relatively quickly relating to the depth of a tree, while DecisionTreeClassifier uses depth to a much greater effect. I now appreciate the Information Gain method of scoring significantly more than the 'error' rate scoring, as DecisionStumpInfoGain gets a much lower classification error than DecisionStumpErrorRate. DecisionStumpErrorRate also does not bode well with trees of a depth greater than 4.



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 $O(n^2 d)$ 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 [2 points]

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?

Answer: The two implementations are the same only on this particular dataset. If you want to definitively prove that the two implementations are the same, compare their results to (many) other datasets (e.x. the 2000, 2008, etc. Elections)