CPSC 540 Project 1

Hamed Helisaz

September 15, 2025

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: $\begin{bmatrix} 2 \\ 10 \\ 0 \end{bmatrix}$

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: False

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

   Answer: True

5. $AB = BA$.

   Answer: False

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

   Answer: True

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

   Answer: False

8. $x^T A y = y^T A^T x$.

   Answer: True

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

   Answer: True

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

    Answer: False

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: 0.92

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: 1.01%

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

   Answer: False Positive

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

   Answer: 0.96%

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: 0.96%

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: Reduce false positive rates (increase specificity) as most of positives are not detected

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: $f'(x) = 15x^2 - 2$

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

   Answer: $f'(x) = 1 - 2x$

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

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: $\begin{bmatrix} \exp(x_1)/Z \\ \exp(x_2)/Z \\ \exp(x_3)/Z \end{bmatrix}$

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: $\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$

4. $f(x) = \frac{1}{2}x^\top A x$ 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: 14/3

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

   Answer: 1/4

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

   Answer: 0

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

   Answer: 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: 1

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, $0 \le x \le 1$.

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:

```python
def foo_grad(x):
    x = np.asarray(x)
    return 4 * x**3
```

```python
def bar_grad(x):
    x = np.asarray(x)
    if_zeros = (x == 0)
    zero_counts = if_zeros.sum()
    if zero_counts == 0:
         return np.prod(x)/x
    grad = np.zeros_like(x)
    if zero_counts == 1:
        grad[if_zeros] =  np.prod(x[~if_zeros])
    return grad
```

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(log(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(d)

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

   Answer: O(d^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:

- func1(N) $\implies O$(N)
- func2(N) $\implies O$(N)
- func3(N) $\implies O$(1)
- func4(N) $\implies O$(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:

   - Min is 0.35
   - Max is 4.86
   - Mean is 1.32
   - Median is 1.16
   - Mode is 0.77

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

   Answer:

   - 5% Quantile is 0.46
   - 25% Quantile is 0.72
   - 50% Quantile is 1.16
   - 75% Quantile is 1.81
   - 95% Quantile is 2.62

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

   Answer:

   - Region with the highest mean is WtdILI
   - Region with the lowest mean is Pac
   - Region with the highest variance is Mtn
   - Region with the lowest variance is Pac

Answer:

```
def q5_1():
    df = pd.read_csv(Path("..", "data", "fluTrends.csv"))
    X = df.values
    names = df.columns.values

    print("5.1.1 \n")
    print(f"Min is {X.min():.2f}\n")
    print(f"Max is {X.max():.2f}\n")
    print(f"Mean is {X.mean():.2f}\n")
```

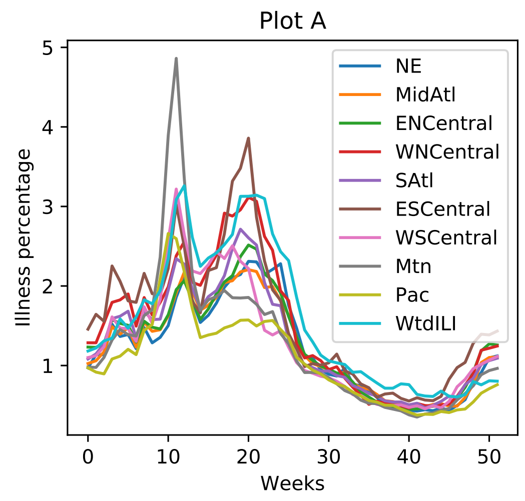
```python
    print(f"Median is {np.median(X):.2f}\n")
    print(f"Mode is {mode(X):.2f}\n")
    print("5.1.2 \n")
    print(f"5% Quantile is {np.quantile(X,0.05):.2f}\n")
    print(f"25% Quantile is {np.quantile(X,0.25):.2f}\n")
    print(f"50% Quantile is {np.quantile(X,0.5):.2f}\n")
    print(f"75% Quantile is {np.quantile(X,0.75):.2f}\n")
    print(f"95% Quantile is {np.quantile(X,0.95):.2f}\n")
    print("5.1.3 \n")
    print(f"Region with the highest mean is {df.mean().idxmax()}")
    print(f"Region with the lowest mean is {df.mean().idxmin()}")
    print(f"Region with the highest variance is {df.var().idxmax()}")
    print(f"Region with the lowest variance is {df.var().idxmin()}")
```

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.

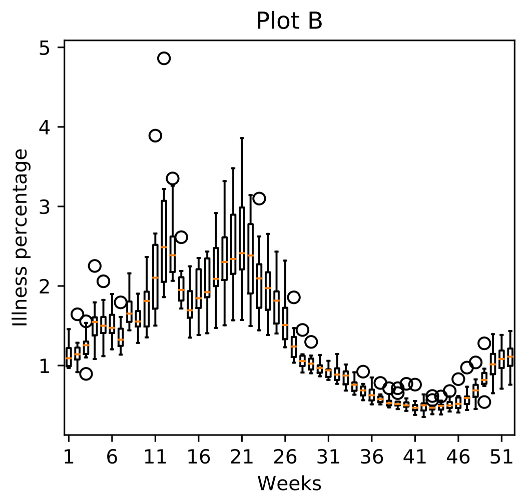
Answer: No, since data is continuous, data that are very close but not *exactly* identical are treated as distinct in `mode` function. A better way is to group the data into intervals (bins) and report the bin with the highest frequency.

5.2 Data Visualization [3 points]

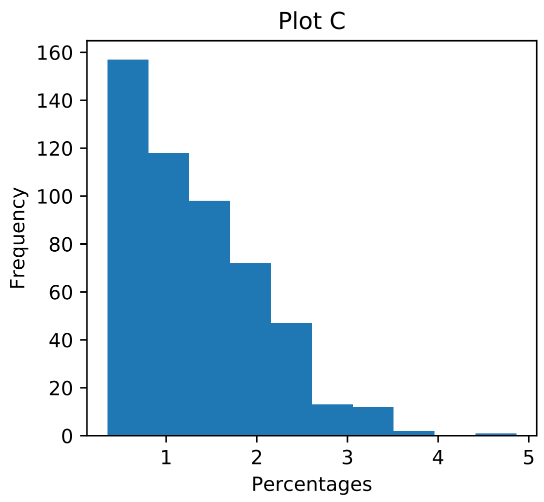
Consider the figure below.



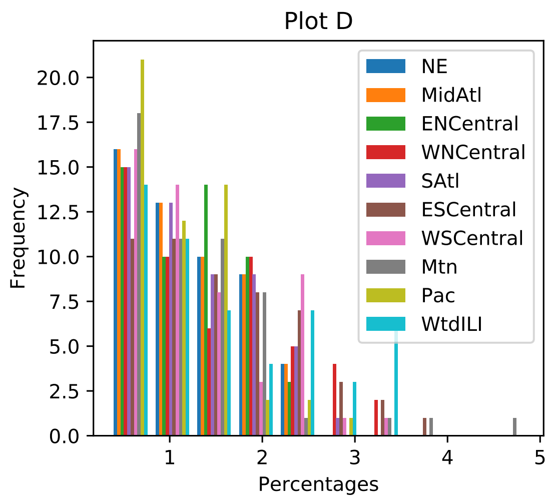
Plot A
Illness percentage
Weeks
NE
MidAtl
ENCentral
WNCentral
SAtl
ESCentral
WSCentral
Mtn
Pac
WtdILI



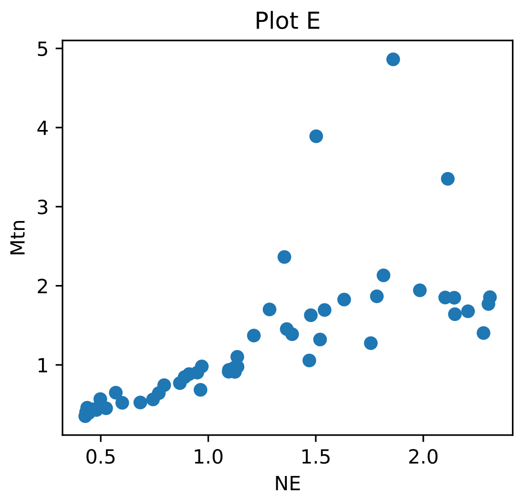
Plot B
Illness percentage
Weeks



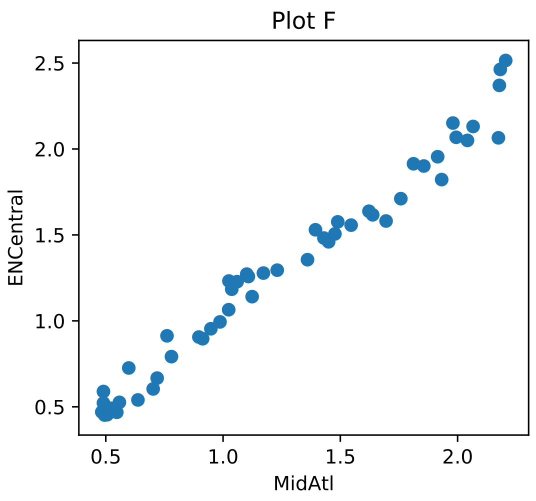
Plot C
Frequency
Percentages



Plot D
Frequency
Percentages
NE
MidAtl
ENCentral
WNCentral
SAtl
ESCentral
WSCentral
Mtn
Pac
WtdILI



Plot E
Mtn
NE



Plot F
ENCentral
MidAtl

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

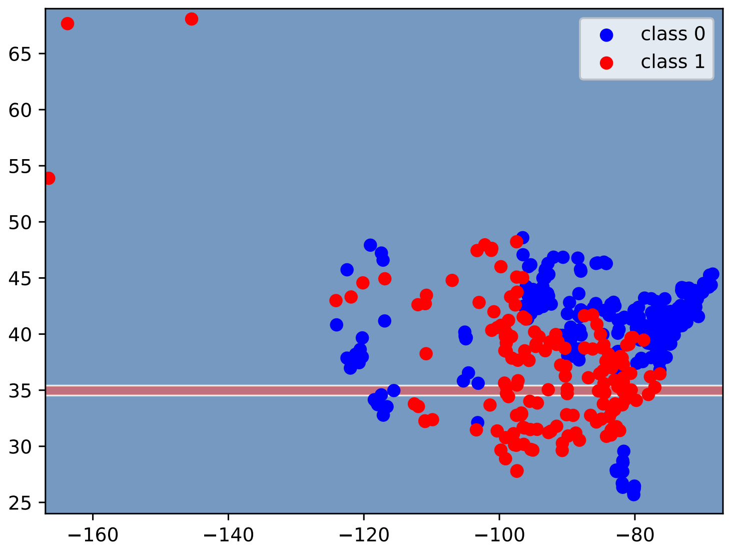
1. A single histogram showing the distribution of *each* column in X.
2. A histogram showing the distribution of each the values in the matrix X.
3. A boxplot grouping data by weeks, showing the distribution across regions for each week.
4. A plot showing the illness percentages over time.
5. A scatterplot 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:
Plot A → (4) Percentages over time (multi-line time series vs. weeks).
Plot B → (3) Boxplots by week (distribution for all regions for each week).
Plot C → (2) Histogram of all values in X.
Plot D → (1) Per-column histograms (one per region of X).
Plot E → (6) Scatterplot of two regions with lowest correlation, with weak linear relation
Plot F → (5) Scatterplot of two regions with highest correlation, with tight near-line relation

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: No, the data do not seem to be split well using an equality-based splitting rule. Threshold-based splits seem to be more appropriate

6.2 Decision Stump Implementation [8 points]

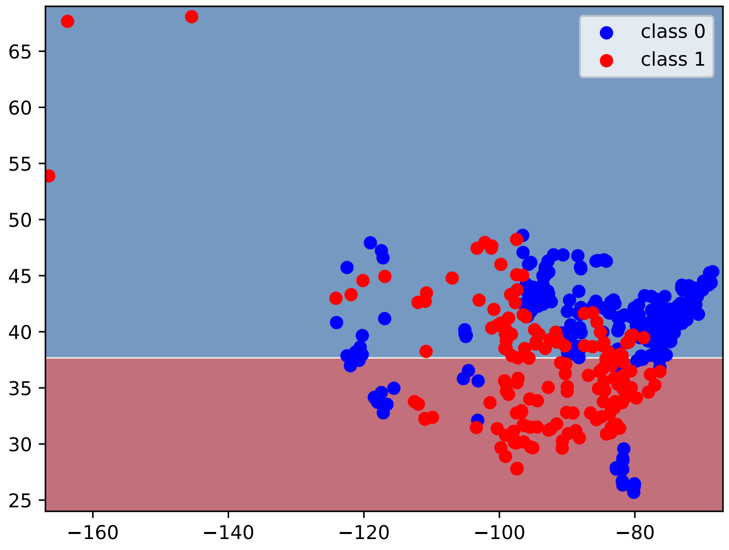
The file `decision_stump.py` contains the class `DecisionStumpEquality` which finds the best decision stump using the equality rule and then makes predictions using that rule. Instead of discretizing the data and using a rule based on testing an equality for a single feature, we want to check whether 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. 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. 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:



Error Rate = 0.253

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

    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 i in range(n):
                # Choose value to equate to
                t = X[i, j]

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

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

        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]

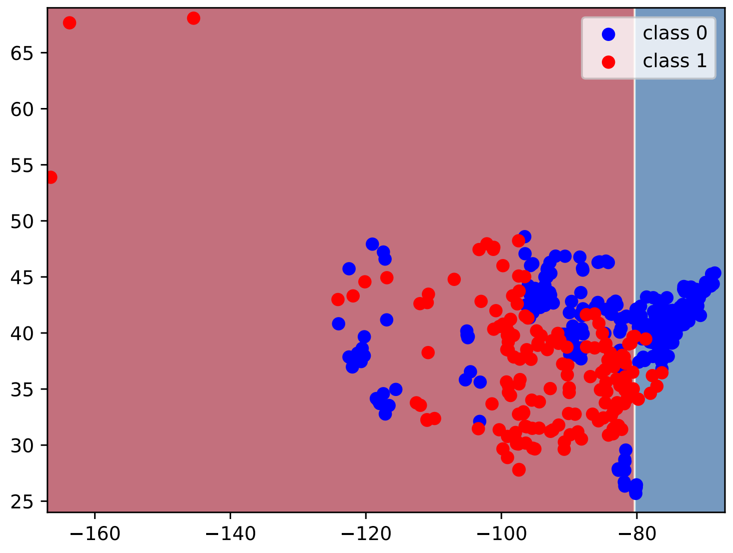
In `decision_stump.py`, create a `DecisionStumpInfoGain` class that fits using the information gain criterion discussed in lecture. Report the updated error you obtain. Submit your class definition code as a screenshot or using the `lstlisting` environment. Submit the classification boundary figure.

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.

Answer:



Error Rate = 0.325!

```
class DecisionStumpInfoGain(DecisionStumpErrorRate):
    # This is not required, but one way to simplify the code is
    # to have this class inherit from DecisionStumpErrorRate.
    # Which methods (init, fit, predict) do you need to overwrite?
    y_hat_yes = None
    y_hat_no = None
    j_best = None
    t_best = None

    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

        maxGain = 0

        # Loop over features looking for the best split
        for j in range(d):
            for i in range(n):
                # Choose value to equate to
                t = X[i, j]

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

                # Make predictions
                y_pred = y_yes_mode * np.ones(n)
                y_pred[X[:, j] <= t] = y_no_mode

                # Compute gain
                count_yes = np.bincount(y[is_more], minlength = len(count))
                n_yes = len(y[is_more]) ## n_less is always positive
                count_no = np.bincount(y[~is_more], minlength = len(count))
                n_no = len(y[~is_more])
                if n_yes == 0 or n_no == 0:
                    continue
                # Find the gain for each possible split
                gain = entropy(count/n) - n_no/n*entropy(count_no/n_no) -
                ↪  n_yes/n*entropy(count_yes/n_yes)

                # Compare to maximum gain so far
                if gain > maxGain:
                    # This is the lowest error, store this value
                    maxGain = gain
                    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). 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.

Answer:

```python
def predict_hard_coded(X):
    n, d = X.shape
    y_hat = np.zeros(n)
    for i in range(n):
        if X[:,0] > -80.30510 and X[:,1] > 36.453576:
            y_hat[i] = 0
        elif X[:,0] > -80.30510 and X[:,1] <= 36.453576:
            y_hat[i] = 1
        elif X[:,0] <= -80.30510 and X[:,1] > 37.669007:
            y_hat[i] = 0
        elif X[:,0] <= -80.30510 and X[:,1] <= 37.669007:
            y_hat[i] = 1
    return y_hat
```

6.5 Decision Tree Training Error [2 points]

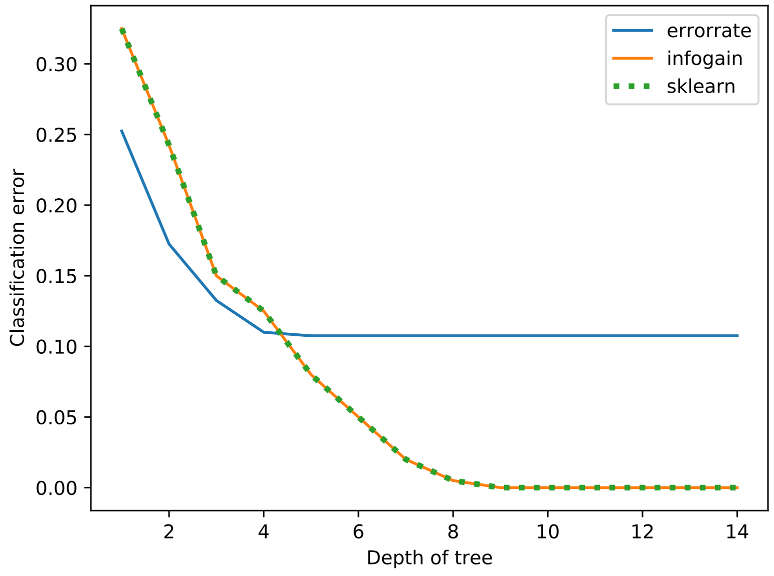
Running `python main.py 6.5` fits decision trees of different depths using the following different implementations:

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

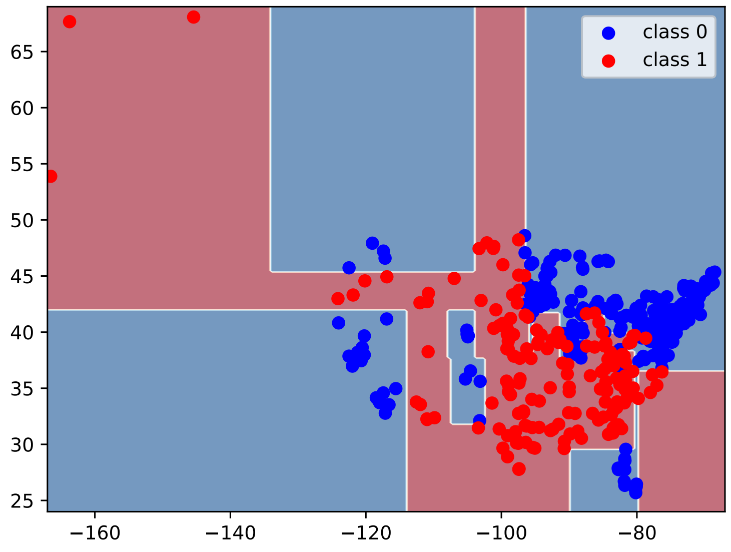
Run the code and look at the figure. Describe what you observe. Can you explain the results? Why is approach (1) so disappointing? Also, submit a classification boundary plot of the model with the lowest training error.

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

Answer:



The error-rate tree starts a bit better at low depths, but it stops improvement early (depth 4). The info-gain tree and sklearn, however, keeps improving and basically hit around zero training error by depth 9.
With error rate, only the majority matters. A split that makes [0.6, 0.4] is treated the same as [0.9,0.1]. Therefore, many candidate splits tie, and the tree stops splitting which caeses no more progress.
With information gain, uncertainty gets penalized and [0.6,0.4] is worse than [0.9,0.1], so the algorithm has a clear favorite and keeps splitting until leaves are (almost) pure.
Classification boundary plot for sklearn (and infogain) is shown below:



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 [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: No; it does not. First, the running time for our information gain tree is about 4 seconds while scikit-learn's took 0.02 seconds. Also, while error rates for the models are the same, they might have different thresholds and misclassify different examples. To ensure the implementations are the same, predictions should be compared to see if they match. The three structure should also be compared to ensure the nodes make the same split on similar features.