CPSC 340 Assignment 2 (due 2020-01-29 at 11:55pm)

Instructions

Rubric: {mechanics:5}

IMPORTANT!!! Before proceeding, please carefully read the general homework instructions at `https://www.cs.ubc.ca/~fwood/CS340/homework/`. The above 5 points are for following the submission instructions. You can ignore the words "mechanics", "reasoning", etc.

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

1 Training and Testing

If you run `python main.py -q 1`, it will load the *citiesSmall.pkl* data set from Assignment 1. Note that this file contains not only training data, but also test data, `X_test` and `y_test`. After training a depth-2 decision tree with the information gain splitting rule, it will evaluate the performance of the classifier on the test data. With a depth-2 decision tree, the training and test error are fairly close, so the model hasn't overfit much.

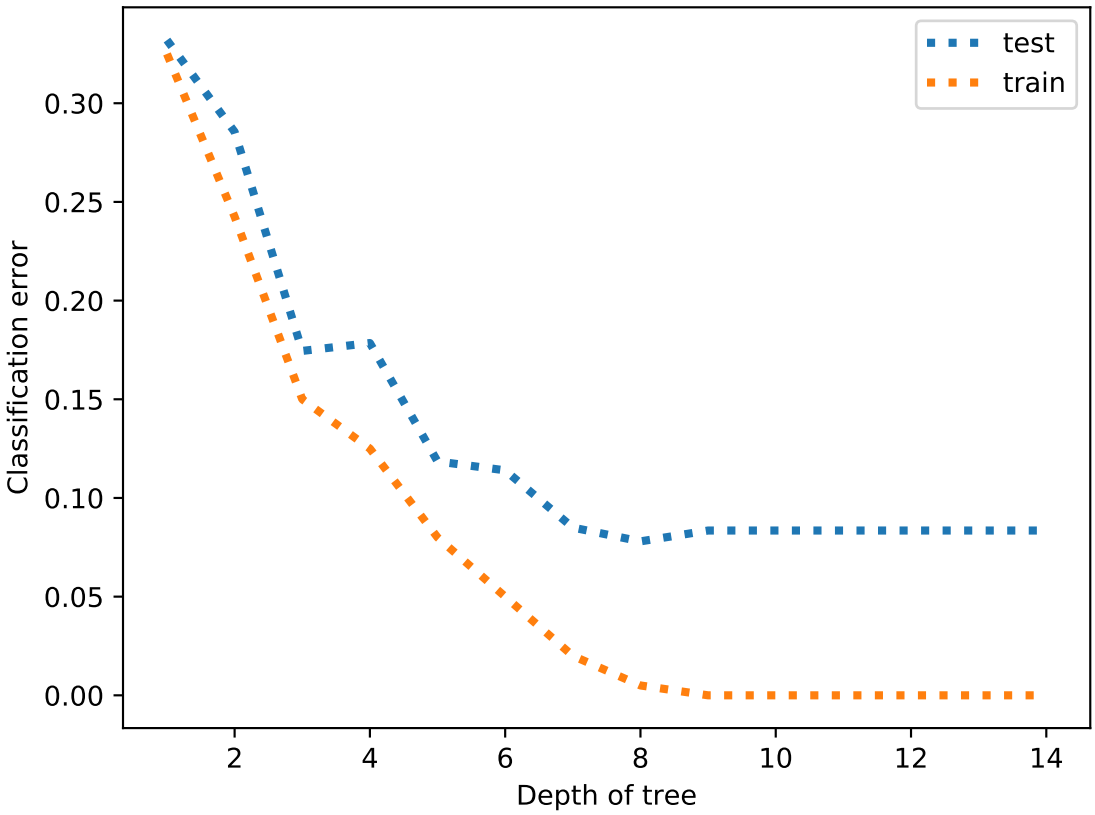
1.1 Training and Testing Error Curves

Rubric: {reasoning:2}

Make a plot that contains the training error and testing error as you vary the depth from 1 through 15. How do each of these errors change with the decision tree depth?

Note: it's OK to reuse code from Assignment 1.

Answer: The training error decreases all the way to 0 as depth of tree increases. The testing error stops decreasing as the depth comes to 8.



1.2 Validation Set

Rubric: {reasoning:3}

Suppose that we didn't have an explicit test set available. In this case, we might instead use a *validation* set. Split the training set into two equal-sized parts: use the first $n/2$ examples as a training set and the second $n/2$ examples as a validation set (we're assuming that the examples are already in a random order). What depth of decision tree would we pick to minimize the validation set error? Does the answer change if you switch the training and validation set? How could use more of our data to estimate the depth more reliably?

Answer: I would choose 8, since the validation error is smallest at that point. My answer will change to 6 when two sets swap. To make depth more reliabel, we can used K-fold cross validation.

2 Naive Bayes

In this section we'll implement naive Bayes, a very fast classification method that is often surprisingly accurate for text data with simple representations like bag of words.

2.1 Naive Bayes by Hand

Consider the dataset below, which has 10 training examples and 3 features:

$$X = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} \text{spam} \\ \text{spam} \\ \text{spam} \\ \text{spam} \\ \text{spam} \\ \text{spam} \\ \text{not spam} \\ \text{not spam} \\ \text{not spam} \\ \text{not spam} \end{bmatrix}.$$

The feature in the first column is <your name> (whether the e-mail contained your name), in the second column is "pharmaceutical" (whether the e-mail contained this word), and the third column is "PayPal" (whether the e-mail contained this word). Suppose you believe that a naive Bayes model would be appropriate for this dataset, and you want to classify the following test example:

$$\hat{x} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}.$$

2.1.1 Prior probabilities

Rubric: {reasoning:1} Compute the estimates of the class prior probabilities (you don't need to show any work):

- $p(\text{spam})$.

  Answer: $\frac{3}{5}$

- $p(\text{not spam})$.

  Answer: $\frac{2}{5}$

2.1.2 Conditional probabilities

Rubric: {reasoning:1}

Compute the estimates of the 6 conditional probabilities required by naive Bayes for this example (you don't need to show any work):

- $p(\text{<your name>} = 1 \mid \text{spam})$.

  Answer: $\frac{1}{6}$

- $p(\text{pharmaceutical} = 1 \mid \text{spam})$.

  Answer: $\frac{5}{6}$

- $p(\text{PayPal} = 0 \mid \text{spam})$.

  Answer: $\frac{1}{3}$

- $p(\text{<your name>} = 1 \mid \text{not spam})$.

  Answer: 1

- $p(\text{pharmaceutical} = 1 \mid \text{not spam})$.

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

- $p(\text{PayPal} = 0 \mid \text{not spam})$.

  Answer: $\frac{3}{4}$

2.1.3 Prediction

Rubric: {reasoning:1}

Under the naive Bayes model and your estimates of the above probabilities, what is the most likely label for the test example? (Show your work.)

Answer:
Assume that all x_i are indenpendent.

1. $p(\text{spam} \mid 1, 1, 0) \propto p(\text{<name>} = 1, \text{phar} = 1, \text{PayPal} = 0) * p(\text{spam})$
   $= p(\text{<name>} = 1 \mid \text{spam}) * p(\text{phar} = 1 \mid \text{spam}) * p(\text{PayPal} = 0 \mid \text{spam}) * p(\text{spam})$
   $= \frac{1}{6} * \frac{5}{6} * \frac{1}{3} * \frac{3}{5} = \frac{1}{36}$
2. $p(\text{not spam} \mid 1, 1, 0) \propto p(\text{<name>} = 1, \text{phar} = 1, \text{PayPal} = 0) * p(\text{not spam})$
   $= p(\text{<name>} = 1 \mid \text{not spam}) * p(\text{phar} = 1 \mid \text{not spam}) * p(\text{PayPal} = 0 \mid \text{not spam}) * p(\text{not spam})$
   $= 1 * \frac{1}{4} * \frac{3}{4} * \frac{2}{5} = \frac{3}{40}$
   Since $\frac{1}{36} > \frac{3}{40}$, it is more likely to be not spam.

2.1.4 Laplace smoothing

Rubric: {reasoning:2}

One way to think of Laplace smoothing is that you're augmenting the training set with extra counts. Consider the estimates of the conditional probabilities in this dataset when we use Laplace smoothing (with $\beta = 1$). Give a set of extra training examples that we could add to the original training set that would make the basic estimates give us the estimates with Laplace smoothing (in other words give a set of extra training examples that, if they were included in the training set and we didn't use Laplace smoothing, would give the same estimates of the conditional probabilities as using the original dataset with Laplace smoothing). Present your answer in a reasonably easy-to-read format, for example the same format as the data set at the start of this question.

Answer:

$$X = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad y = \begin{bmatrix} \text{spam} \\ \text{not spam} \\ \text{spam} \\ \text{not spam} \end{bmatrix}.$$

2.2 Bag of Words

Rubric: {reasoning:3}

If you run `python main.py -q 2.2`, it will load the following dataset:

1. X: A binary matrix. Each row corresponds to a newsgroup post, and each column corresponds to whether a particular word was used in the post. A value of 1 means that the word occured in the post.
2. *wordlist*: The set of words that correspond to each column.
3. y: A vector with values 0 through 3, with the value corresponding to the newsgroup that the post came from.
4. *groupnames*: The names of the four newsgroups.

5. *Xvalidate* and *yvalidate*: the word lists and newsgroup labels for additional newsgroup posts.

Answer the following:

1. Which word corresponds to column 51 of X? (This is column 50 in Python.)

   Answer:
   print(wordlist[50])
   output: lunar

2. Which words are present in training example 501?

   Answer:
   x_frame = pandas.DataFrame('bin': X[500,:],'name': wordlist, columns=['bin','name'])
   x_filter = x_frame[x_frame['bin']==1]
   print(x_filter)
   output: car,fact,gun,video

3. Which newsgroup name does training example 501 come from?

   Answer:
   print(groupnames[int(y[500])])
   output: talk.*

2.3 Naive Bayes Implementation

Rubric: {code:5}

If you run `python main.py -q 2.3` it will load the newsgroups dataset, fit a basic naive Bayes model and report the validation error.

The `predict()` function of the naive Bayes classifier is already implemented. However, in `fit()` the calculation of the variable `p_xy` is incorrect (right now, it just sets all values to 1/2). Modify this function so that `p_xy` correctly computes the conditional probabilities of these values based on the frequencies in the data set. Include your code and the validation error that you obtain in you pdf GradeScope submission. Also, compare your validation error to what you obtain with scikit-learn's implementation, `BernoulliNB`.

Answer: The validation error is 0.188 and the BernoulliNB validation error is 0.187

```
def predict(self, X):

N, D = X.shape
C = self.num_classes
p_xy = self.p_xy
p_y = self.p_y

y_pred = np.zeros(N)
for n in range(N):

probs = p_y.copy() # initialize with the p(y) terms
for d in range(D):
if X[n, d] != 0:
probs *= p_xy[d, :]
else:
probs *= (1-p_xy[d, :])

y_pred[n] = np.argmax(probs)

return y_pred
```

2.4 Runtime of Naive Bayes for Discrete Data

Rubric: {reasoning:3}

For a given training example i, the predict function in the provided code computes the quantity

$$p(y_i \mid x_i) \propto p(y_i) \prod_{j=1}^{d} p(x_{ij} \mid y_i),$$

for each class y_i (and where the proportionality constant is not relevant). For many problems, a lot of the $p(x_{ij} \mid y_i)$ values may be very small. This can cause the above product to underflow. The standard fix for this is to compute the logarithm of this quantity and use that $\log(ab) = \log(a) + \log(b)$,

$$\log p(y_i \mid x_i) = \log p(y_i) + \sum_{j=1}^{d} \log p(x_{ij} \mid y_i) + (\text{irrelevant propportionality constant}) .$$

This turns the multiplications into additions and thus typically would not underflow.

Assume you have the following setup:

- The training set has n objects each with d features.
- The test set has t objects with d features.
- Each feature can have up to c discrete values (you can assume $c \leq n$).
- There are k class labels (you can assume $k \leq n$)

You can implement the training phase of a naive Bayes classifier in this setup in $O(nd)$, since you only need to do a constant amount of work for each $X(i, j)$ value. (You do not have to actually implement it in this way for the previous question, but you should think about how this could be done.) What is the cost of classifying t test examples with the model and this way of computing the predictions?

Answer: It's O(tdk), since the function need to go through three for loops (t eamples, d features and k classes).

3 K-Nearest Neighbours

Rubric: {code:3, reasoning:4}

In the *citiesSmall* dataset, nearby points tend to receive the same class label because they are part of the same U.S. state. For this problem, perhaps a k-nearest neighbours classifier might be a better choice than a decision tree. The file *knn.py* has implemented the training function for a k-nearest neighbour classifier (which is to just memorize the data).

Fill in the `predict` function in `knn.py` so that the model file implements the k-nearest neighbour prediction rule. You should Euclidean distance, and may numpy's `sort` and/or `argsort` functions useful. You can also use `utils.euclidean_dist_squared`, which computes the squared Euclidean distances between all pairs of points in two matrices.

1. Write the `predict` function. Include this code in your GradeScope submission.
2. Report the training and test error obtained on the *citiesSmall* dataset for $k = 1$, $k = 3$, and $k = 10$. How do these numbers compare to what you got with the decision tree?
3. Include the plot generated by `utils.plotClassifier` on the *citiesSmall* dataset for $k = 1$, using both your implementation of KNN and the KNeighborsClassifier from scikit-learn.

4. Why is the training error 0 for $k = 1$?

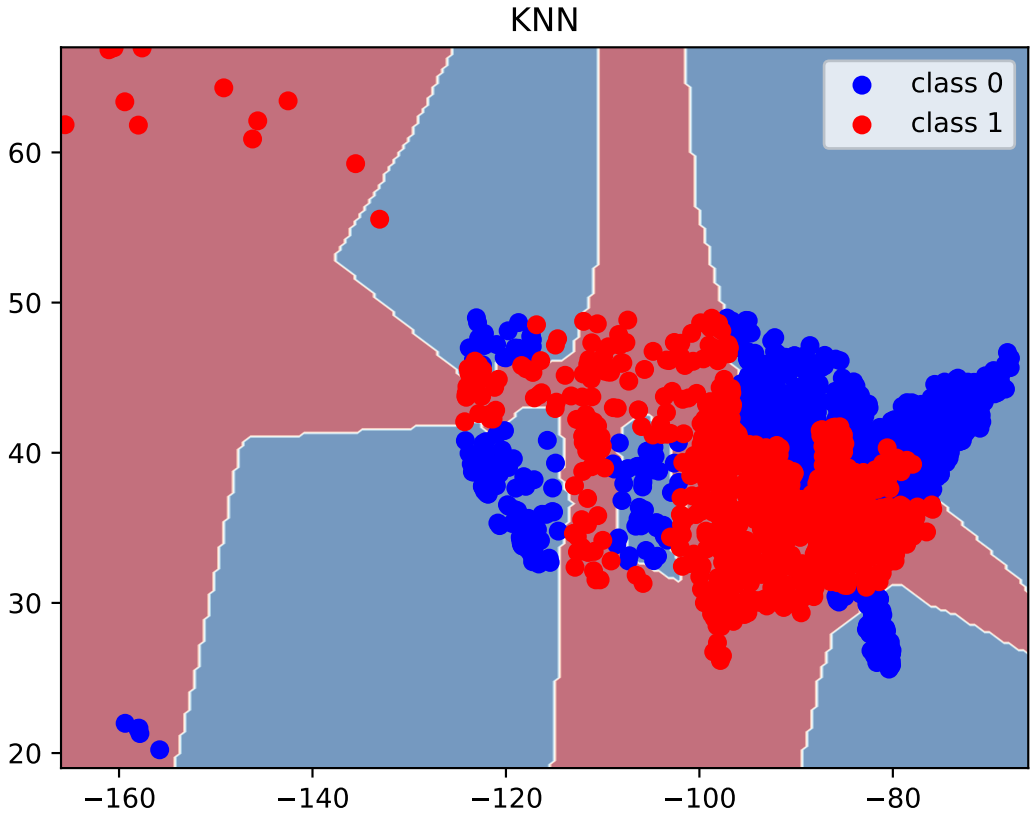
5. If you didn't have an explicit test set, how would you choose k?

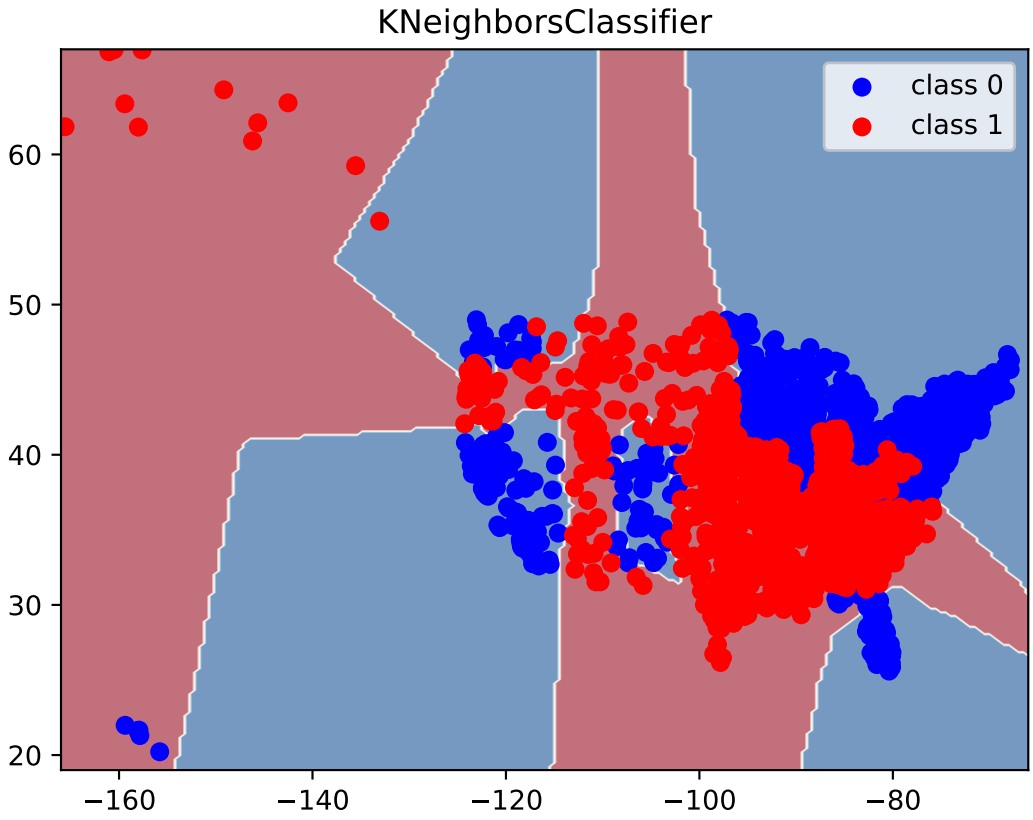
Answer: Q1.

```
def predict(self, Xtest):
# raise NotImplementedError()
n,d = self.X.shape
t,d = Xtest.shape

ytest = np.zeros(t)
Dis = utils.euclidean_dist_squared(self.X, Xtest)


for i in range(t):

neighbor = np.argsort(Dis[:, i])
ytest[i] = utils.mode(self.y[neighbor[0:self.k]])

return ytest
```

Answer: Q2.
when k = 1, test error = 0, test error = 0.0645
when k = 3, test error = 0.0275, test error = 0.066
when k = 10, test error = 0.0725, test error = 0.097
These error is relatively small compared with the error generated by decision tree.

Answer: Q3.


KNN
class 0
class 1
60
50
40
30
20
−160
−140
−120
−100
−80



Answer: Q4. When k = 1, every point's nearest neighbor is itself only. Therefore, it won't produce any error.

Answer: Q5. Use cross validation or randomly split the data into validation set and training set.

4 Random Forests

4.1 Implementation

Rubric: {code:4,reasoning:3}

The file *vowels.pkl* contains a supervised learning dataset where we are trying to predict which of the 11 "steady-state" English vowels that a speaker is trying to pronounce.

You are provided with a `RandomStump` class that differs from `DecisionStumpInfoGain` in that it only considers $\lfloor \sqrt{d} \rfloor$ randomly-chosen features.[1] You are also provided with a `RandomTree` class that is exactly the same as `DecisionTree` except that it uses `RandomStump` instead of `DecisionStump` and it takes a bootstrap sample of the data before fitting. In other words, `RandomTree` is the entity we discussed in class, which makes up a random forest.

If you run `python main.py -q 4` it will fit a deep `DecisionTree` using the information gain splitting criterion. You will notice that the model overfits badly.

[1] The notation $\lfloor x \rfloor$ means the "floor" of x, or "x rounded down". You can compute this with `np.floor(x)` or `math.floor(x)`.

1. Why doesn't the random tree model have a training error of 0?

   Answer: Ramdom forest use bootstrap sample, so that not all trainining data will be used.

2. Create a class `RandomForest` in a file called `random_forest.py` that takes in hyperparameters `num_trees` and `max_depth` and fits `num_trees` random trees each with maximum depth `max_depth`. For prediction, have all trees predict and then take the mode. Make sure to include the code you have written in your pdf GradeScope submission.

3. Using 50 trees, and a max depth of ∞, report the training and testing error. Compare this to what we got with a single `DecisionTree` and with a single `RandomTree`. Are the results what you expected? Discuss.

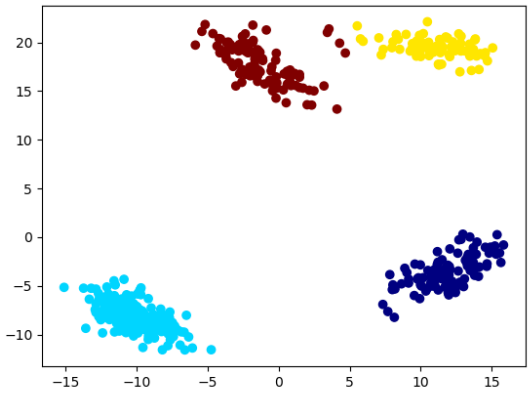
   Answer: training error = 0 and testing error = 0.170. Compared with other tree models, random forest has lower testing error. Even if the max depth is ∞, random forest prevents the model from overfitting the training data.

4. Compare your implementation with scikit-learn's `RandomForestClassifier` for both speed and accuracy, and briefly discuss. You can use all default hyperparameters if you wish, or you can try changing them.

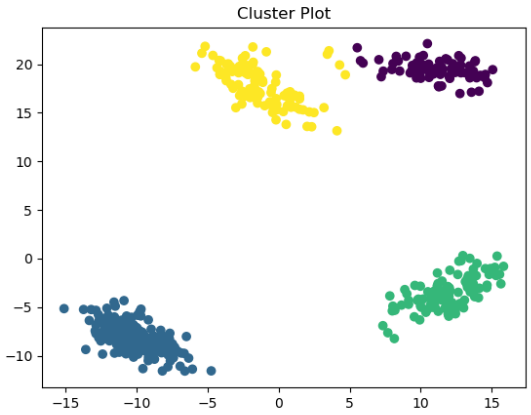
   Answer: Compared with the "Random Forest" we implemented, "RandomForestClassifier" takes much less time to run, and two methods have similar testing errors when their hyperparameters are the same.

5 Clustering

If you run `python main.py -q 5`, it will load a dataset with two features and a very obvious clustering structure. It will then apply the k-means algorithm with a random initialization. The result of applying the algorithm will thus depend on the randomization, but a typical run might look like this:



(Note that the colours are arbitrary – this is the label switching issue.) But the 'correct' clustering (that was used to make the data) is this:



5.1 Selecting among k-means Initializations

Rubric: {reasoning:5}

If you run the demo several times, it will find different clusterings. To select among clusterings for a *fixed* value of k, one strategy is to minimize the sum of squared distances between examples x_i and their means w_{y_i},

$$f(w_1, w_2, \ldots, w_k, y_1, y_2, \ldots, y_n) = \sum_{i=1}^{n} \|x_i - w_{y_i}\|_2^2 = \sum_{i=1}^{n} \sum_{j=1}^{d} (x_{ij} - w_{y_i j})^2.$$

where y_i is the index of the closest mean to x_i. This is a natural criterion because the steps of k-means alternately optimize this objective function in terms of the w_c and the y_i values.

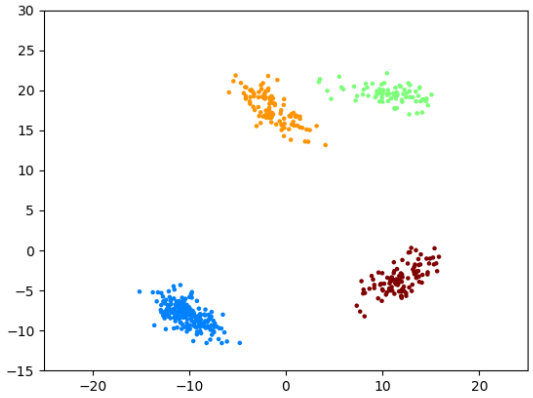
1. In the `kmeans.py` file, add a new function called `error` that takes the same input as the `predict` function but that returns the value of this above objective function. Make sure to include the code you have written in your pdf GradeScope submission.
2. What trend do you observe if you print the value of this error after each iteration of the k-means algorithm?
3. Using the code from question 5 in `main.py` (modify if needed), output the clustering obtained by running k-means 50 times (with $k = 4$) and taking the one with the lowest error. Submit your plot.
4. Looking at the hyperparameters of scikit-learn's `KMeans`, explain the first four (`n_clusters`, `init`, `n_init`, `max_iter`) very briefly.

Answer: Q1

```
def error(self, X):
means = self.means
y = self.predict(X)
N,D = X.shape
error = 0
for n in range(N):
for d in range(D):
error += (X[n,d] - means[y[n],:])^2
return error
```

Answer: Q2: The error keep decreasing

Answer: Q3



Answer: Q4:
n_cluster = the number of clusters to generate
init = a method to choose initial center
n_init = the number of time to run Kmeans with different initial center
max_iter = maximum number of iterations

5.2 Selecting k in k-means

Rubric: {reasoning:5}

We now turn to the task of choosing the number of clusters k.

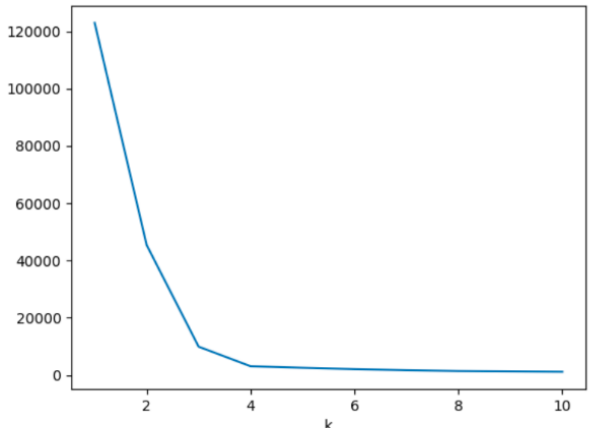
1. Explain why we should not choose k by taking the value that minimizes the `error` function.

   Answer: The function assumes that the k is known. The k with smallest error would be the largest value of k.

2. Explain why even evaluating the `error` function on test data still wouldn't be a suitable approach to choosing k.

   Answer: The future data could contain outliers that increase error with given k.

3. Hand in a plot of the minimum error found across 50 random initializations, as a function of k, taking k from 1 to 10.



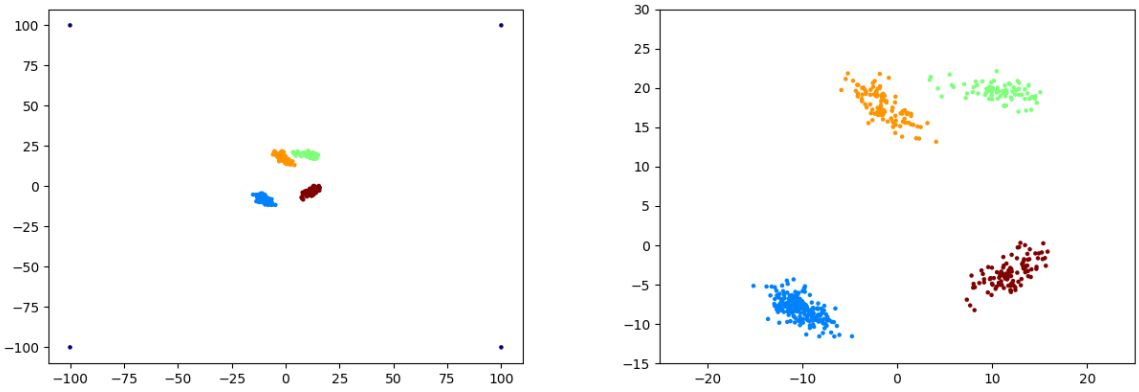
4. The *elbow method* for choosing k consists of looking at the above plot and visually trying to choose the k that makes the sharpest "elbow" (the biggest change in slope). What values of k might be reasonable according to this method? Note: there is not a single correct answer here; it is somewhat open to interpretation and there is a range of reasonable answers.

   Answer: Either 3 or 4 sould be an ideal k for this data.

5.3 Density-Based Clustering

Rubric: {reasoning:2}

If you run `python main.py -q 5.3`, it will apply the basic density-based clustering algorithm to the dataset from the previous part, but with some outliers added. The final output should look somewhat like this:



(The right plot is zoomed in to show the non-outlier part of the data.) Even though we know that each object was generated from one of four clusters (and we have 4 outliers), the algorithm finds 6 clusters and does not assign some of the original non-outlier objects to any cluster. However, the clusters will change if we change the parameters of the algorithm. Find and report values for the two parameters, `eps` (which we called the "radius" in class) and `minPts`, such that the density-based clustering method finds:

1. The 4 "true" clusters.

   Answer: eps in (2-3), minPts = 2

2. 3 clusters (merging the top two, which also seems like a reasonable interpretaition).

   Answer: eps in (4-12), minPts = 2

3. 2 clusters.

   Answer: eps in (13-15), minPts = 2

4. 1 cluster (consisting of the non-outlier points).

   Answer: eps >= 16 minPts = 2

6 Very-Short Answer Questions

Rubric: {reasoning:13}

Write a short one or two sentence answer to each of the questions below. Make sure your answer is clear and concise.

1. What is an advantage of using a boxplot to visualize data rather than just computing its mean and variance?

   Answer: Boxplot better describes the distibution of the data by indicating 100%, 75%, 50%, 25%, 0% quantiles and outliers of the data.

2. What is a reason that the the data may not be IID in the email spam filtering example from lecture?

   Answer: The email changes over time and their for the disturbution could change.

3. What is the difference between a validation set and a test set?

   Answer: A validation set is from training set to estimate the test error. A test set is another independent data set.

4. Why can't we (typically) use the training error to select a hyper-parameter?

   Answer: Doing that could construct a complicated model that overfits the training data

5. What is the effect of n on the optimization bias (assuming we use a parametric model).

   Answer: Optimization bias will decrease when increasing training examples.

6. What is an advantage and a disadvantage of using a large k value in k-fold cross-validation.

   Answer: The advantage is the error will be smaller. The disadvantage is that the CV will be expensive.

7. Why can we ignore $p(x_i)$ when we use naive Bayes?

   Answer: Based on Bayes formula $p(y_i = a \mid x_i) = \frac{p(x_i \mid y_i=a)*p(y_i=a)}{p(x_i)}$, the denominator won't change no matter what value a is. $p(y_i = a \mid x_i)$ is always proportional to $p(x_i \mid y_i = a) * p(y_i = a)$.

8. For each of the three values below in a naive Bayes model, say whether it's a parameter or a hyper-parameter:

   (a) Our estimate of $p(y_i)$ for some y_i.

   Answer: parameter

   (b) Our estimate of $p(x_{ij} \mid y_i)$ for some x_{ij} and y_i.

   Answer: parameter

   (c) The value β in Laplace smoothing.

   Answer: hyper-parameter

9. What is the effect of k in KNN on the two parts (training error and approximation error) of the fundamental trade-off. Hint: think about the extreme values.

   Answer: The larger the k, traning error will be larger, but the approximation error will be smaller.

10. Suppose we want to classify whether segments of raw audio represent words or not. What is an easy way to make our classifier invariant to small translations of the raw audio?

    Answer: Add translated verisons.

11. Both supervised learning and clustering models take in an input x_i and produce a label y_i. What is the key difference?

    Answer: When using supervised learning, we know the exact y_i value in traning set. When using clustering models, there is no y_i in origin data and we need to assign it based on clustring models.

12. Suppose you chose k in k-means clustering (using the squared distances to examples) from a validation set instead of a training set. Would this work better than using the training set (which just chooses the largest value of k)?

    Answer: Yes, because training error keeps decreases as k increase and will eventually lead the k to be n.

13. In k-means clustering the clusters are guaranteed to be convex regions. Are the areas that are given the same label by KNN also convex?

    Answer: Yes.