CPSC 340 Assignment 2 (due Friday September 27 at 11:55pm)

The assignment instructions are the same as Assignment 1, except you have the option to work in a group of 2. If you work in a group, please only hand in *one* assignment. It is recommended that you work in groups as the assignment is quite long, but please only submit one assignment for the group and make sure that everyone's name/ID is on the front page.

Answer:
Name:Yixuan LI
ID:61261814
Name:Kexin Wen
ID:76020940

1 Training and Testing

1.1 Traning Error

Running `example_train.jl` fits decision trees of different depths using two different implementations: the "decisionStump" function from Assignment 1, and using a variant using a more sophisticated splitting criterion called the information gain. Describe what you observe. Can you explain the results?

Answer: As the training depth increase, the training error of information gain-based decision tree reduced to zero, but the accuracy-based decision tree training error keep decrease until reach a certain value

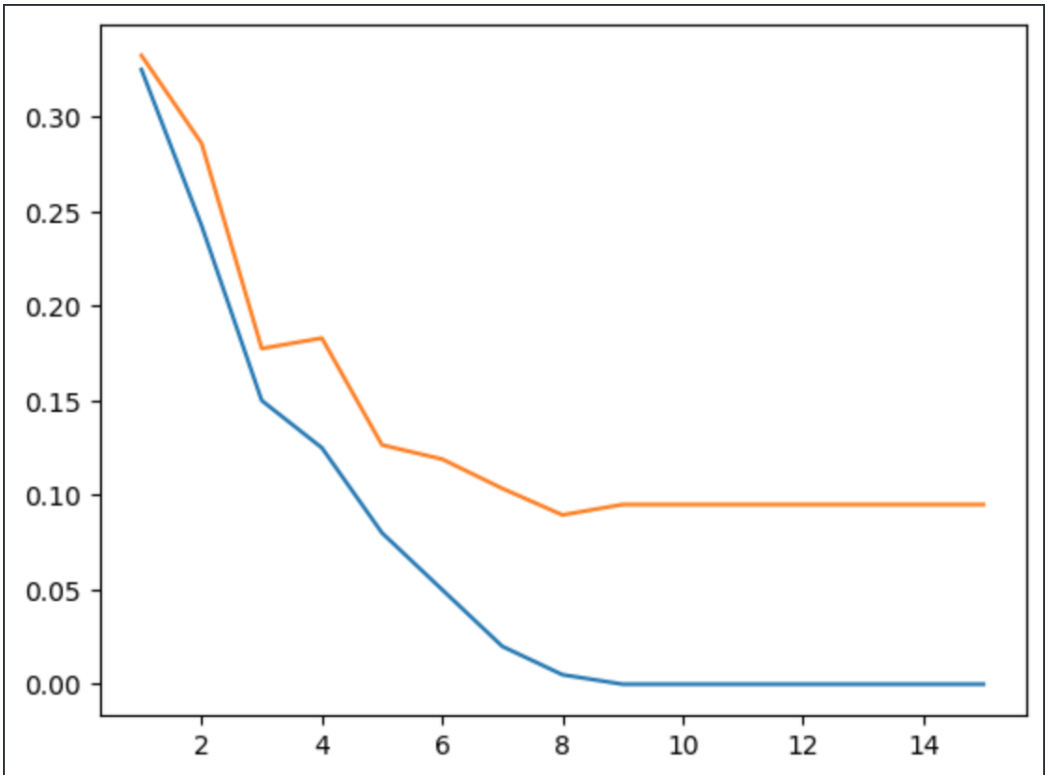
1.2 Training and Testing Error Curves

Notice that the *citiesSmall.mat* file also contains test data, "Xtest" and "ytest". Running *example_trainTest* trains a depth-2 decision tree and evaluates its performance 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.

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: use the provided infogain-based decision tree code from the previous subsection.

Answer:
Orange: Test Error
Blue: Training Error



as the graph shows, the training error keeps going down until it reaches to 0, and total trending of test error is also going down to a certain value.

1.3 Validation Set

Suppose we're in the typical case where we don't have the labels for the test data. 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 if we minimized the validation set error? Does the answer change if you switch the training and validation set? How could we use more of our data to estimate the depth more reliably?

Note: use the provided infogain-based decision tree code from the previous subsection.

Answer: In order to minimized the validation set error, the depth of decision tree should be equal to 3 (Validation error = 0.145).
The answer change when switching the training and validation setThe most ideal depth now is 6, which lead the validation error = 0.12)
We can use k-fold validation to estimate the depth more reliably.

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

Compute the estimates of the class prior probabilities (you don't need to show any work):

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

Answer:

- $p(\text{spam}) = 0.6$.
- $p(\text{not spam}) = 0.4$.

2.1.2 Conditional probabilities

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

- $p(\text{<your name>} = 1 \mid \text{spam})$.
- $p(\text{pharmaceutical} = 1 \mid \text{spam})$.
- $p(\text{PayPal} = 0 \mid \text{spam})$.
- $p(\text{<your name>} = 1 \mid \text{not spam})$.
- $p(\text{pharmaceutical} = 1 \mid \text{not spam})$.
- $p(\text{PayPal} = 0 \mid \text{not spam})$.

Answer:

- $p(\text{<your name>} = 1 \mid \text{spam}) = 0.167$
- $p(\text{pharmaceutical} = 1 \mid \text{spam}) = 0.833$

- $p(\text{PayPal} = 0 \mid \text{spam}) = 0.333$
- $p(\text{<your name>} = 1 \mid \text{not spam}) = 1$
- $p(\text{pharmaceutical} = 1 \mid \text{not spam}) = 0.25$
- $p(\text{PayPal} = 0 \mid \text{not spam}) = 0.75$

2.1.3 Prediction

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:
The numerator of $p(\text{not spam} \mid \text{<your name>} = 1, \text{pharmaceutical} = 1, \text{PayPal} = 0)$
$= p(\text{<your name>} = 1 \mid \text{not spam}) * p(\text{pharmaceutical} = 1 \mid \text{not spam}) * p(\text{PayPal} = 0 \mid \text{not spam}) * p(notspam)$
$= 0.167 * 0.833 * 0.333 * 0.4$
$= 0.01852$

The numerator of $p(\text{spam} \mid \text{<your name>} = 1, \text{pharmaceutical} = 1, \text{PayPal} = 0)$
$= p(\text{<your name>} = 1 \mid \text{spam}) * p(\text{pharmaceutical} = 1 \mid \text{spam}) * p(\text{PayPal} = 0 \mid \text{spam}) * p(spam)$
$= 1 * 0.25 * 0.75 * 0.6$
$= 0.1125$

Because $0.1125 > 0.01852$, the "spam" is the most likely label.

2.2 Bag of Words

If you run the script *example_bagOfWods.jl*, it will load the following dataset:

1. X: A sparse 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 1 through 4, with the value corresponding to the newsgroup that the post came from.
4. *groupnames*: The names of the four newsgroups.
5. $Xvalid$ and $yvalid$: the word lists and newsgroup labels for additional newsgroup posts.

Answer the following:

1. Which word is present in the newsgroup post if there is a 1 in column 30 of X?

   Answer: "$format$"

2. Which words are present in training example 200?

   Answer: "$food$" and "$world$"

3. Which newsgroup name does training example 200 come from?

   Answer: $talk.*$

2.3 Naive Bayes Implementation

If you run the function *example_decisionTree_newsgroups.jl* it will load the newsgroups dataset and report the test error for decision trees of different sizes (it may take a while for the deeper trees, as this is a sub-optimal implementation). On other other hand, if you run the function *example_naiveBayes.jl* it will fit the basic naive Bayes model and report the test error.

While the *predict* function of the naive Bayes classifier is already implemented, 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. Hand in your code and report the test error that you obtain.

Answer:

```
22    # Add code for Q2.3
23    countp=zeros(d,k)
24    countn=zeros(d,k)
25
26    for m in 1:k
27        for j in 1:d
28           for i in 1:n
29               if X[i,j]==1 && y[i]==m
30                   countp[j,m]+=1
31               elseif X[i,j]==0 && y[i]==m
32                   countn[j,m]+=1
33               end
34           end
35       end
36     end
37
38    p_xy=zeros(2,d,k)
39    for m in 1:k
40        for j in 1:d
41            p_xy[1,j,m]=countp[j,m]/counts[m]
42            p_xy[2,j,m]=countn[j,m]/counts[m]
43        end
44    end
45    # End of Add code for Q2.3
```

```
julia> include("example_naiveBayes.jl")
Test error with naive Bayes: 0.188
```

2.4 Runtime of Naive Bayes for Discrete Data

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?

Answer: $O(ndk)$

3 K-Nearest Neighbours

In *citiesSmall* dataset, nearby points tend to receive the same class label because they are part of the same state. This indicates that a $k-$nearest neighbours classifier might be a better choice than a decision tree (while naive Bayes would probably work poorly on this dataset). The file *knn.jl* has implemented the training function for a $k-$nearest neighbour classifier (which is to just memorize the data) but the predict function always just predicts 1.

3.1 KNN Prediction

Fill in the *predict* function in *knn.jl* so that the model file implements the k-nearest neighbour prediction rule. You should use Euclidean distance.

Hint: although it is not necessary, you may find it useful to pre-compute all the distances (using the *distancesSquared* function in *misc.jl*) and to use the *sortperm* command.

1. Hand in the predict function.
2. Report the training and test error obtained on the *citiesSmall.mat* dataset for $k = 1$, $k = 3$, and $k = 10$. (You can use *example_knn.jl* to get started.)
3. Hand in the plot generatied by classifier2Dplot on the *citiesSmall.mat* dataset for $k = 1$ on the training data.
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?

Hint: when writing a function, it is typically a good practice to write one step of the code at a time and check if the code matches the output you expect. You can then proceed to the next step and at each step test is if the function behaves as you expect. You can also use a set of inputs where you know what the output should be in order to help you find any bugs. These are standard programming practices: it is not the job of the TAs or the instructor to find the location of a bug in a long program you've written without verifying the individual parts on their own.

Answer:

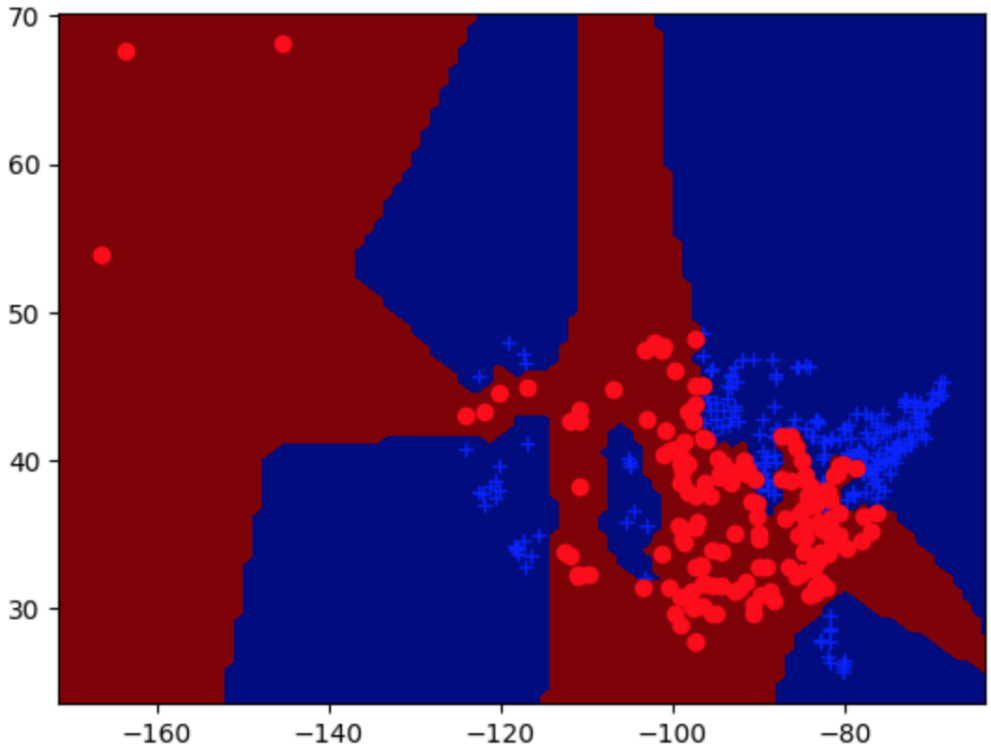
1.

```
function knn_predict(Xhat,X,y,k)
  (n,d) = size(X)
  (t,d) = size(Xhat)
  k = min(n,k) # To save you some debuggin

 # Modified code for Q3.1
  DisM=distancesSquared(X,Xhat)
  p=zeros(n,t)
  yhat=zeros(t)
  tobemode=zeros(t,k)
  for i in 1:t
     p[:,i]=sortperm(DisM[:,i])
     yhat[i]=mode(y[p[1:k,i].|>Int])
 end

  return yhat
  # End of modified code for Q3.1
end

```

2.

```
Train Error with 1-nearest neighbours: 0.000
Test Error with 1-nearest neighbours: 0.065
Train Error with 3-nearest neighbours: 0.028
Test Error with 3-nearest neighbours: 0.066
Train Error with 10-nearest neighbours: 0.072
Test Error with 10-nearest neighbours: 0.097
```

3.



4. Training error is zero because when k = 1, it will choose the closest sample. And since the test sample is in the training set, it will choose itself as the closest sample.

5. We can separate the training set into training and validation set, such as k-fold cross validation. Based on different validation set, we pick the k that has lowest validation error.

3.2 Condensed Nearest Neighbours

The file *citiesBig1.mat* contains a version of this dataset with more than 30 times as many cities. KNN can obtain a lower test error if it's trained on this dataset, but the prediction time will be very slow. A common strategy for applying KNN to huge datasets is called *condensed nearest neighbours*, and the main idea is to only store a *subset* of the training examples (and to only compare to these examples when making predictions). A simple variation of this algorithm would be:

initialize subset with first training example;
for *each training example* **do**
  if *the example is incorrectly classified by the KNN classifier using the current subset* **then**
    add the current example to the subset;
  else
    do *not* add the current example to the subset (do nothing);
  end
end

Algorithm 1: Condensed Nearest Neighbours

You are provided with an implementation of this *condensed nearest neighbours* algorithm in *knn.jl*.

1. The point of this algorithm is to be faster than KNN. Try running the condensed nearest neighbours

(called "cknn" in the code) on the *citiesBig1* dataset and report how long it takes to make a prediction. What about if you try to use KNN for this dataset?

2. Report the training and testing errors for condensed NN, as well as the number of training examples in the subset, on the *citiesBig1* dataset with $k = 1$.
3. Why is the training error with $k = 1$ now greater than 0?
4. If we entered the coordinates of Vancouver into the predict function, would it be predicted to be in a blue state or a red state?
5. If you have s examples in the subset, what is the cost of running the predict function on t test examples in terms of n, d, t, and s?
6. Try out your function on the dataset *citiesBig2*. Why are the test error *and* training error so high (even for $k = 1$) for this method on this dataset?

Answer:

1. Running time for CNN:

```
  5.290659 seconds (366.93 k allocations: 382.928 MiB, 23.32% gc time)
```

   Running time for KNN:

```
263.962741 seconds (207.13 k allocations: 11.336 GiB, 8.55% gc time)
```

2.

```
size(Xcond) = (457, 2)
Train Error with 1-nearest neighbours: 0.008
Test Error with 1-nearest neighbours: 0.018
```

3. Because in the training process, it doesn't iterate all training samples, but the most condense ones.
4. Blue
5. Assume the running time of sortsperm() function in knn.jl is $s \log s$, the total running time is $O(MAX(dst, st \log s))$
6. Because all the training samples in dataset citiesBig2 are similar to each other. Therefore, the size of the subset is way smaller than the citiesBig1, which contain more diverse samples.

4 Random Forests

4.1 Implementation

The file *vowels.jld* 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 a `randomTree` function in *randomTree.jl* (based on information gain). The random tree model differs from the decision tree model in two ways: it takes a bootstrap sample of the data before fitting and when fitting individual stumps it only considers $\lfloor\sqrt{d}\rfloor$ randomly-chosen features.[1] In other words, `randomTree` is the model we discussed in class that is combined to make up a random forest.

If you run *example_randomTree.jl*, it will fit both models to the dataset, and you will notice that it overfits badly.

[1] The notation $\lfloor x \rfloor$ means the "floor" of x, or "x rounded down".

1. If you set the *depth* parameter to *Inf*, why do the training functions terminate?

   Answer: When all the training samples have been put into distinct node

2. Why does the random tree model, using infoGain and a depth of *Inf*, have a training error greater 0?

   Answer: Random tree model would boothtrap the training sample. Therefore, some sample will be missing, and some sample will be duplicated.

3. Create a function `randomForest` that takes in hyperparameters `depth` and `nTrees` (number of trees), and fits `nTrees` random trees each with maximum depth `depth`. For prediction, have all trees predict and then take the mode. Hand in your function. Hint: you can define an array for holding 10 *GenericModel* types using:
   `subModels = Array{GenericModel}(undef,10)`.

   Answer:

```
function randomForest(X,y,depth,k)
(n,d)=size(X)
#select n sample using bootstrapping from all X
bootstrap=zeros(k,n)
subModels=Array{GenericModel}(undef,k)
for i in 1:k
    # k: #training set
    bootstrap[i,:]=rand(1:n,n)
    media=bootstrap[i,:].|>Int
    #randomTree_sub(X[media,:],y[media],depth)
    subModels[i]=randomTree_sub(X[media,:],y[media],depth)
end

return subModels
end
```

```
modelforest=randomForest(X,y,depth,k)

prehat=zeros(n,k)
yhatt=zeros(n)
for i in 1:k
 prehat[:,i]=modelforest[i].predict(X)
end
for j in 1:n
    yhatt[j]=mode(prehat[j,:])
end
trainError=mean(yhatt .!=y)
@printf("Train Error with depth-%d random forest: %.3f\n",depth,trainError)

prehattest=zeros(t,k)
yhattest=zeros(n)
 for i in 1:k
     prehattest[:,i]=modelforest[i].predict(Xtest)
end
for j in 1:t
    yhattest[j]=mode(prehattest[j,:])
end

testError = mean(yhattest .!= ytest)
@printf("Test Error with depth-%d random forest: %.3f\n",depth,testError)
```

Change in *example_randomTree.jl*

4. Using 50 trees, and a 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:

```
Train Error with depth-Inf decision tree: 0.000
Test Error with depth-Inf decision tree: 0.367
Train Error with depth-Inf random tree: 0.189
Test Error with depth-Inf random tree: 0.504
Train Error with depth-Inf random forest: 0.000
Test Error with depth-Inf random forest: 0.152
julia>
```

The results are we expected. As using 50 trees and depth of infinity, the training error or decision tree

and random forest are both zero. And the test error of random forest is the lowest among these three model.

5. Why does a random forest typically have a training error of 0, even though random trees typically have a training error greater than 0?

   Answer: For Random Forest model, the classifying rules are all derived exactly from the training set and thus has the capability to classify perfectly to the training set given.

5 K-Means Clustering

If you run the function *example_Kmeans*, 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 due to the label switching problem.) But the 'correct' clustering (that was used to make the data) is something more like this:

5.1 Selecting among k-means Initializations

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. Write a new function called *kMeansError* that takes in a dataset X, a set of cluster assignments y, and a set of cluster means W, and computes this objective function. Hand in your code.

   Answer:

```
function kMeansError(X,y,W)
    (n,d) = size(X)

    f = 0
    for i in 1:n
        for j = 1:d
            f += (X[i,j] - W[y[i],j])^2
        end
    end
    return f
end
```

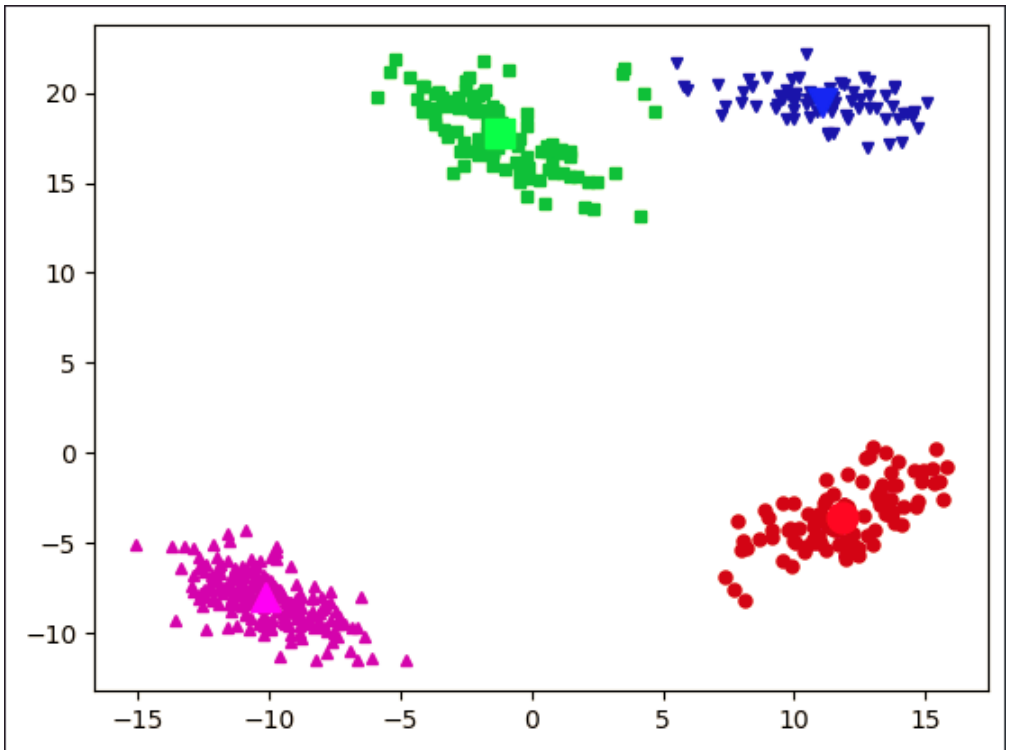
   The code is provided by the instructor.

2. Instead of printing the number of labels that change on each iteration, what trend do you observe if you print the value of *kMeansError* after each iteration of the k-means algorithm?

   Answer: The kMeansError value keep decreasing until it stop in the fix value

3. Using the *clustering2Dplot* file, output the clustering obtained by running k-means 50 times (with $k = 4$) and taking the one with the lowest error. Note that the k-means training function will run much faster if you set `doPlot = false` or just remove this argument.

   Answer:



5.2 Selecting k in k-means

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

1. Explain why the *kMeansError* function should not be used to choose k.

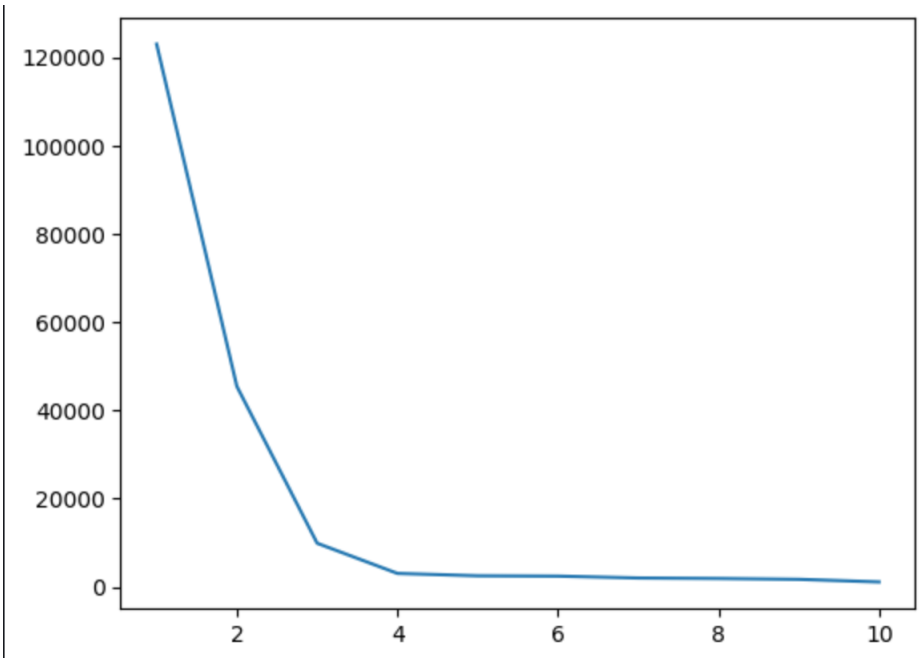
   Answer: Because the value of kMeansError is determined by the sum of distance between training samples and their corresponding cluster center, as the k increase, the number of sample in the cluster goes down, thus the kMeansError will always decreases.

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

   Answer: As the k keep increasing, there will be more cluster. For a fixed test set, each test sample would have a closer cluster center. Therefore, the kMeansError(the distance between test sample and cluster center) would goes down.

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

Answer:



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: Based on the above plot, $k = 3$ has the sharpest "elbow" (greatest change in slope)

6 Very-Short Answer Questions

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 a feature transformation that you might do to address a "coupon collecting" problem in your data?

   Answer: Combine those city features that are in the same province into a province feature, since there are fewer province coupons to collect than city coupons.

2. What is one reason we would want to look at scatterplots of the data before doing supervised learning?

   Answer: Scatterplots gives us a nice quick overview of the data, might discover certain pattern by human vision

3. When we fit decision stumps, why do we only consider $>$ (for example) and not consider $<$ or $\geq$?

   Answer: Because it would give equivalent rules.

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

   Answer: not all the examples come from the same distribution

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

Answer: Validation set is used to tune the parameters of a classifier and find the optimal in training step. Test set is used only to assess the performance of a fully-trained classifier

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

   Answer: training data are used to find the the best rule values of the model. Hyper-parameter controls how complex our model is. One can always fit training data better by making the model more complicated, which will cause over-fitting

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

   Answer: Larger K means less bias towards overestimating the true expected error (as training folds will be closer to the total dataset) but higher variance and higher running time (as we are getting closer to the limit case)

8. Why is naive Bayes called "naive"?

   Answer: It's called naive because it makes the assumption that all attributes are independent of each other. This assumption is in lots of real world situations this does not fit.

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: As k grows, training error increase and approximation error decreases.

10. For any parametric model, how does increasing number of training examples n affect the two parts of the fundamental trade-off.

    Answer: approximation error tends to get smaller as n gets larger, grow as model get more complicated.

11. 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 transformed data during training, such as translated/ change of volume of training audio.

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

    Answer: The former one y_i contain the predict label, based on its rule; the y_i produced by clustering model is not label but the index of cluster.

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: No, just look at the plot diagram of question 3-1-3,it is separated by multiple regions with a non-convex boundary made up of piecewise linear hyperplanes