CPSC 340 Assignment Solutions

Sugun Machipeddy

1 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.

1.1 Naive Bayes by Hand

Rubric: {reasoning:3}

Consider the dataset below, which has 10 training examples and 2 features:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Suppose you believe that a naive Bayes model would be appropriate for this dataset, and you want to classify the following test example:

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

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

- $p(y = 1) = \frac{6}{10}$.
- $p(y = 0) = \frac{4}{10}$.

(b) Compute the estimates of the 4 conditional probabilities required by naive Bayes for this example (you don't need to show any work):

- $p(x_1 = 1|y = 1) = \frac{3}{6}$.
- $p(x_2 = 0|y = 1) = \frac{2}{6}$.
- $p(x_1 = 1|y = 0) = 1$.
- $p(x_2 = 0|y = 0) = \frac{3}{4}$.

(c) 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.)

- $P(x_1 = 1, x_2 = 0|y = 1) = P(x_1 = 1|y = 1) * P(x_2 = 0|y = 1) = \frac{1}{6}$

- $P(x_1 = 1, x_2 = 0|y = 0) = P(x_1 = 1|y = 0) * P(x_2 = 0|y = 0) = \frac{3}{4}$
  Since the probability of $P(x_1 = 1, x_2 = 0|y = 0)$ is higher, the likely label for $\hat{x} = \begin{bmatrix} 1 & 0 \end{bmatrix}$ is $y = [0]$

1.2 Bag of Words

Rubric: {reasoning:3}

If you run `python main.py -q 1.2`, 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. *Xvalidate* and *yvalidate*: the word lists and newsgroup labels for additional newsgroup posts.

Answer the following:

1. Which word corresponds to column 50 of X?
   - lunar
2. Which words are present in training example 500?
   - car, fact, gun and video
3. Which newsgroup name does training example 500 come from?
   - it comes form group 3, ie 'Sci.*'

1.3 Naive Bayes Implementation

Rubric: {code:5}

If you run `python main.py -q 1.3` it will load the newsgroups dataset and report the test error for a random forest, and also fit the basic naive Bayes model and report the test 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. Hand in your code and the validation error that you obtain. Also, briefly comment on the accuracy as compared to the random forest and scikit-learn's naive Bayes implementation.

- the implementation is available at avaialble in *naive_bayes.py* file. the validation errors are as follows
  Random Forest (sklearn) validation error : 0.198.
  Naive Bayes (ours) validation error: 0.315.
  Naive Bayes (sklearn) validation error: 0.187.
  Surely our implementation of naive bayes is not so accurate as scikit learns implentation because of underflow.the accuracy decreases in our implementation because of floting point multiplication.

1.4 Laplace smoothing

Rubric: {code:1}

Do the following:

1. Modify your code so that it uses Laplace smoothing, with β as a parameter taken in by the constructor.
   - the code has been implemented in *naive_bayes.py* file.
2. Did you need to modify your fit function, predict function, or both?
   - No, we do not have to modiy the fit and predict function.
3. Take a look at the documentation for the scikit-learn version of naive Bayes the code is using. How much Laplace smoothing does it use by default? Using the same amount of smoothing with your code, do you get the same results?
   - the value of beta used was 1. when implemented in our algorithm,we do not get the same result. the error was reduced by 0.04.

1.5 Runtime of Naive Bayes for Discrete Data

Rubric: {reasoning:3}

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: **the cost of classifying t examples is** $O(dt)$

2 Random Forests

2.1 Implementation

Rubric: {reasoning:7}

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 `DecisionStump` in two ways: it uses the information gain splitting criterion (instead of classification accuracy), and 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

[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)`.

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 2` it will fit a deep `DecisionTree` using the information gain splitting criterion. You will notice that the model overfits badly.

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

   - the random tree model does not have a training error of zero because of its infinite depth. the error gets propogated in the tree due to its infinite depth and therefore is not zero.

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.

   - **the implemetation is available in** $random\_forests.py$**.**

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.

   - the training and testing errors are as follows.
     Decision tree info gain Training error: 0.000, Testing error: 0.367 .
     Random tree info gain Training error: 1.000, Testing error: 1.000 .
     Random forest info gain Training error: 0.532, Testing error: 0.682 .
     the results are not as expected. the error value obtained by our implementation is higher than other classifier menthods.

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.

   - the training and test error with scikit-learns implemetnation is as follows.
     Random forest info classifier gain, more trees Training error: 0.000, Testing error: 0.159 .

2.2 Very-Short Answer Questions

Rubric: {reasoning:3}

1. What is a a disadvantage of using a very-large number of trees in a random forest classifier?

   - the runtime increases as the number of trees increases.

2. Your random forest classifier has a training error of 0 and a very high test error. Which ones of the following could help performance?

   (a) Increase the maximum depth of the trees in your forest.

   (b) Decrease the maximum depth of the trees in your forest.

   (c) Increase the amout of data you consider for each tree (Collect more data and use $2n$ objects instead of n).

   (d) Decrease the amount of data you consider for each tree (Use $0.8n$ objects instead of n).

   (e) Increase the number of features you consider for each tree.

   (f) Decrease the number of features you consider for each tree.

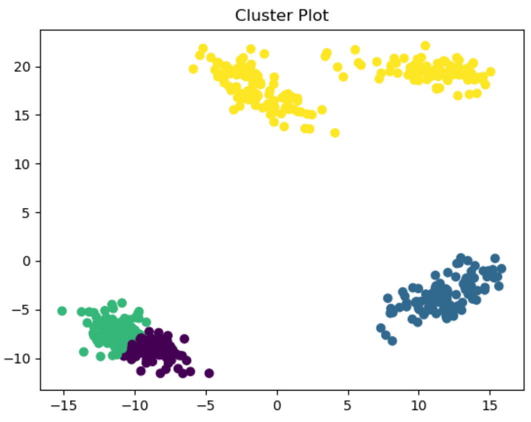
   - **(c) increase the amount of data you consider for each tree.**

- **(e)Increase the number of features you consider for each tree.**

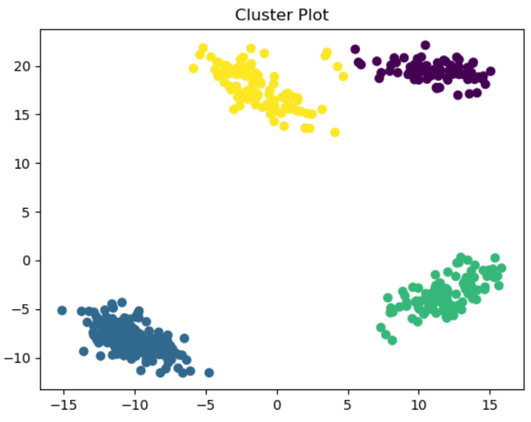
3. Suppose that you were training on raw audio segments and trying to recognize vowel sounds. What could you do to encourage the final classifier to be invariant to translation?
   - we can train the classifier on data that has been obtianed by transalating the original data to a certain extent. this can make the classifier translation invariant.

3 Clustering

If you run `python main.py -q 3`, 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:



3.1 Selecting among k-means Initializations

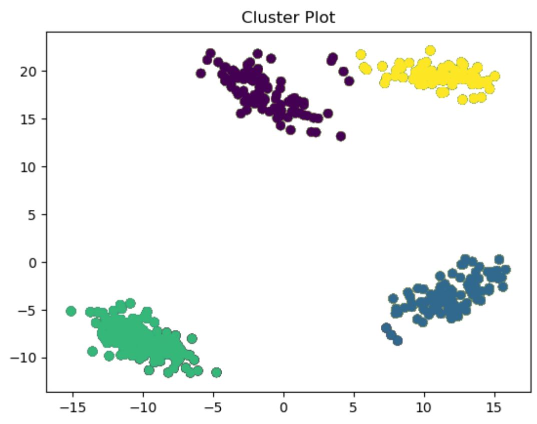
Rubric: {reasoning:5}

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},

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

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. Hand in your code.
   - the error fucntion has been implemented in *kmeans.py* file.
2. What trend do you observe if you print the value of *error* after each iteration of the k-means algorithm?
   - the error decreases with each iteration.
3. Using `plot_2dclustering`, output the clustering obtained by running k-means 50 times (with $k = 4$) and taking the one with the lowest error.
   - Below is the plot with the lowest error.



4. Looking at the hyperparameters of scikit-learn's `KMeans`, explain the first four (`n_clusters`, `init`, `n_init`, `max_iter`) very briefly.

- `n_clusters:`It tells the number of centroids to generate and number of clusters to form.
- `init:` It is the method uded for initialization, can be either Kmeans++, ndarray or random
- `n_init:` it is a representation of the times the kmeans algrithm will run with different centroid seeds.
- `max_iter:` It represents the maximum number of iteration of Kmeans algorithm in a single run.

3.2 Selecting k in k-means

Rubric: {reasoning:5}

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

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

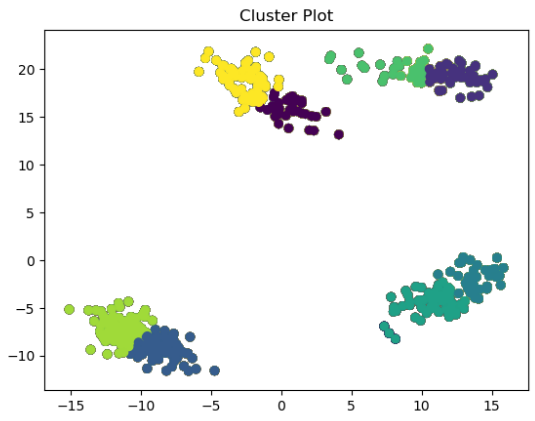
- we cannot select k on the basis of the error because error is based on distance. it is possible to reach a value of K where the distance becomes zero, which is not useful.

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

    - Even on test data, the algorithm will choose the value of K so as to minimize the value or error(distance). we get the same result as above.

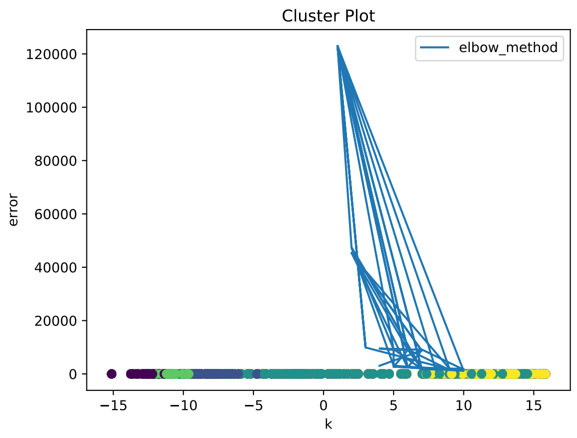
3. Hand in a plot of the minimum error found across 50 random initializations, as a function of k, taking k from 1 to 10.

    - Below is the plot with lowest error with k = 8. It was obtained by running the algorithm 50 times with random initialization and k varying 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.

- The value of K I beleive is resonable 3.



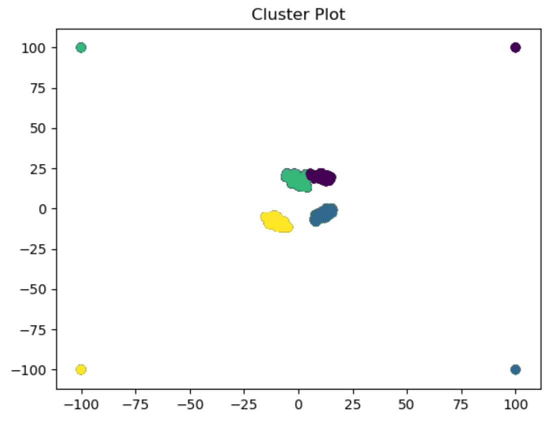
3.3 k-medians

Rubric: {reasoning:5}

The data in *clusterData2.pkl* is the exact same as the above data, except it has 4 outliers that are very far away from the data.

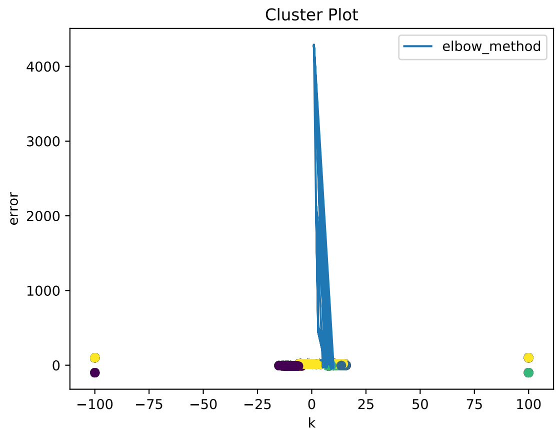
1. Using the `plot_2dclustering` function, output the clustering obtained by running k-means 50 times (with $k = 4$) and taking the one with the lowest error. Are you satisfied with the result?

   - Below is the plot with the lowest error obtained by running kmeans 50 times and $k = 4$.



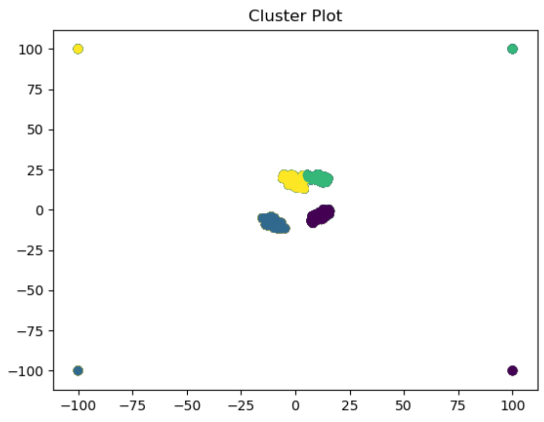
2. What values of k might be chosen by the elbow method for this dataset?

   - I beleive that K should be 3.



3. Implement the k-*medians* algorithm, which assigns examples to the nearest w_c in the L1-norm and to updates the w_c by setting them to the "median" of the points assigned to the cluster (we define the d-dimensional median as the concatenation of the median of the points along each dimension). Hand in your code and plot obtained with 50 random initializations for $k = 4$.

   - the code was implemented in kmeadians.py file and below is the plot with the lowest error obtained with 50 random initializations and $k = 4$.

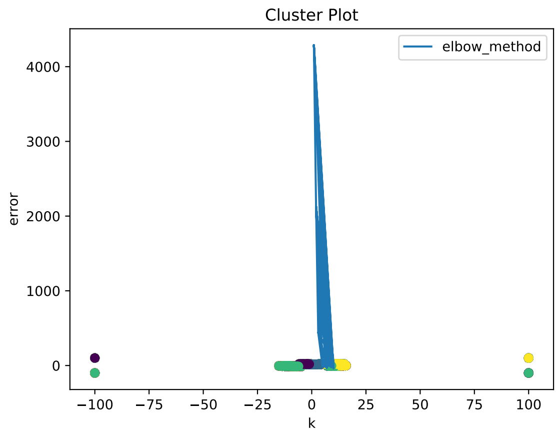


4. Using the L1-norm version of the error (where y_i now represents the closest median in the L1-norm),

$$f(w_1, w_2, \dots, w_k, y_1, y_2, \dots, y_n) = \sum_{i=1}^{n} \|x_i - w_{y_i}\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{d} |x_{ij} - w_{y_i j}|,$$

what value of k would be chosen by the elbow method under this strategy? Are you satisfied with this result?

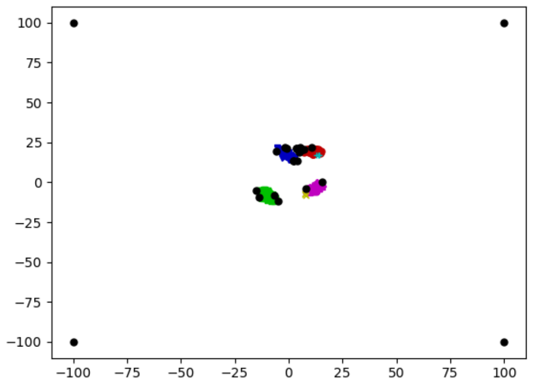
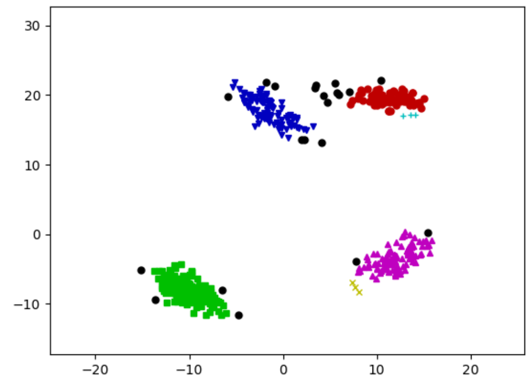
- the value of k should be 3.



3.4 Density-Based Clustering

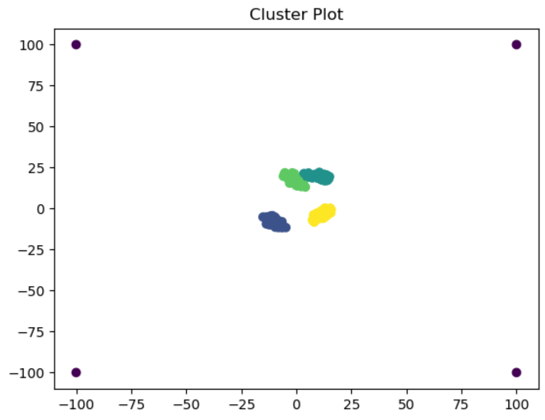
Rubric: {reasoning:2}

If you run `python main.py -q 3.4`, it will apply the basic density-based clustering algorithm to the dataset from the previous part. The final output should look somewhat like this:

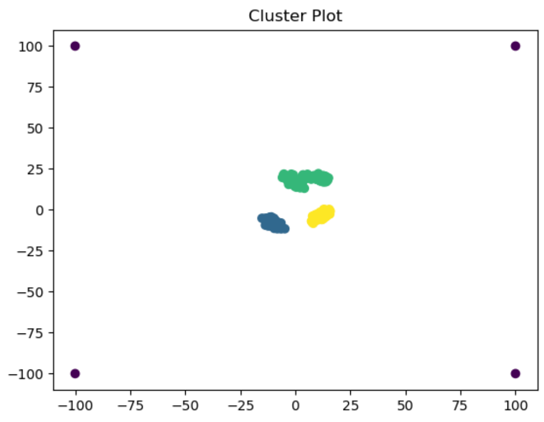



(The right plot is zoomed in to show the non-outlier part of the data.) Even though we know that each object was generated from one of four clusters (and we have 4 outliers), the algorithm finds 6 clusters and does not assign some of the original non-outlier objects to any cluster. However, the clusters 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:

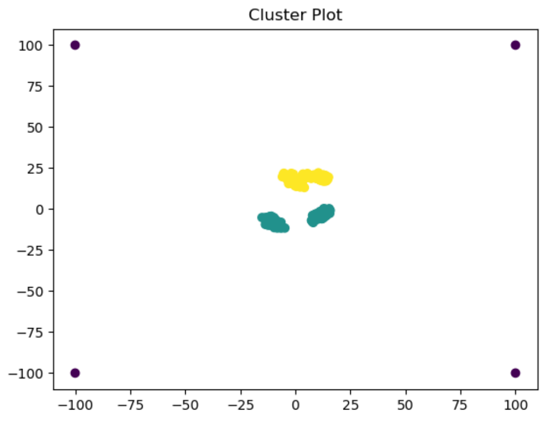
1. The 4 "true" clusters.
   - eps = 2.5 and minpts = 6



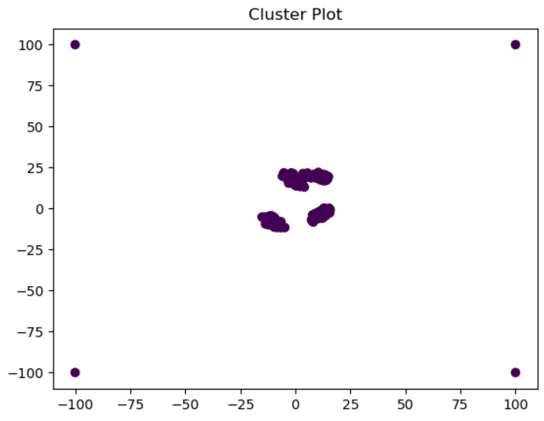
2. 3 clusters (merging the top two, which also seems like a reasonable interpretaition).
   - eps = 6 and minpts = 3



3. 2 clusters.
   - eps = 13 and minpts = 3



4. 1 cluster (consisting of the non-outlier points).
   - eps = 0.1 and minpts = 5



3.5 Very-Short Answer Questions

Rubric: {reasoning:3}

1. Does the standard k-means clustering algorithm always yield the optimal clustering solution for a given k?
   - No, the optimal clustering solution is not quaranteed always.
2. If your set out to minimize the distance between each point and its mean in a k-means clustering, what value of k minimizes this cost? Is this value useful?
   - Minimizing the distance is not a useful approach as when k is same as the number of samples, the value of distance reaches zero. Instead,the plot of the mean distance to the centroid as a function of k can be used as a rough estimate to determine k.
3. Describe a dataset with k clusters where k-means would not be able to find the true clusters.
   - Kmeans fails when there are concentric clusters of data and it sometimes creates clusters even in uniformly distributed data.
4. Suppose that you had only two features and that they have very-different scales (like kilograms vs. milligrams). How would this affect the result of density-based clustering?
   - It becomes difficult to choose an appropriate e if features have very-different scales. this would distort the clustering to an extent.
5. Name a key advantage and drawback of using a supervised outlier detection method rather than an unsupervised method?
   - the advantage is that we can detect complicated outlier patterns, but the disadvantage is that it can only detect known outlier patterns.

4 Vector Quantization

Rubric: {reasoning:6}

Discovering object groups is one motivation for clustering. Another motivation is *vector quantization*, where we find a prototype point for each cluster and replace points in the cluster by their prototype. If our inputs are images, vector quantization gives us a rudimentary image compression algorithm.

Your task is to implement image quantization in *quantize_image.py* with `quantize` and `dequantize` functions. The `quantize` function should take in an image and, using the pixels as examples and the 3 colour channels as features, run k-means clustering on the data with 2^b clusters for some hyperparameter b. The code should store the cluster means and return the cluster assignments. The `dequantize` function should return a version of the image (the same size as the original) where each pixel's orignal colour is replaced with the nearest prototype colour.

To understand why this is compression, consider the original image space. Say the image can take on the values $0, 1, \ldots, 254, 255$ in each colour channel. Since $2^8 = 256$ this means we need 8 bits to represent each colour channel, for a total of 24 bits per pixel. Using our method, we are restricting each pixel to only take on one of 2^b colour values. In other words, we are compressing each pixel from a 24-bit colour representation to a b-bit colour representation by picking the 2^b prototype colours that are "most representative" given the content of the image. So, for example, if $b = 6$ then we have 4x compression.

The loaded image contains a 3D-array representing the RGB values of a picture. Implement the *quantize* and *dequantize* functions and show the image obtained if you encode the colours using 1, 2, 4, and 6 bits with the provided image. You are welcome to use the provided implementation of k-means or the scikit-learn version.

1. Hand in your *quantizeImage* and *deQuantizeImage* functions.
    - the code has been implemented in quantize_image.py file.
2. Show the image obtained if you encode the colours using 1, 2, 4, and 6 bits per pixel (instead of the original 24-bits).
3. Briefly comment on the prototype colours learned in case each, which are saved by the code.