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

Ricky Ma, x2m2b
Kaiwen Hu, e3z1b

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.

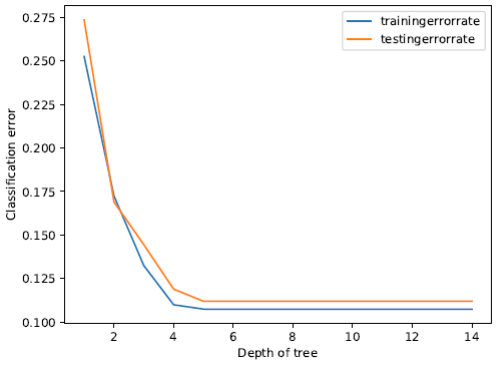
1 Training and Testing

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?
The difference between training and testing error remains relatively small throughout all 15 depths. The classification error decreases until reaching a minimum at a depth of 5 for both training and testing datasets.



2020-01-20 194427.png

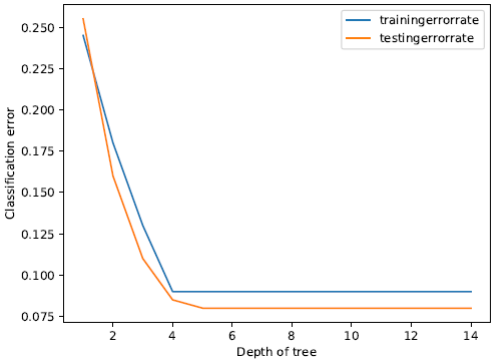
Note: it's OK to reuse code from Assignment 1.

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?

To minimize validation set error, a decision tree of depth 5 should be chosen. If the training and validation set were switched, then only a depth of 4 is necessary.



2020-01-20 194454.png

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):

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

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):

- $p(\text{<your name>} = 1 \mid \text{spam})$. 1/6
- $p(\text{pharmaceutical} = 1 \mid \text{spam})$. 5/6
- $p(\text{PayPal} = 0 \mid \text{spam})$. 2/6
- $p(\text{<your name>} = 1 \mid \text{not spam})$. 1
- $p(\text{pharmaceutical} = 1 \mid \text{not spam})$. 1/4
- $p(\text{PayPal} = 0 \mid \text{not spam})$. 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.)

$p(\text{<your name>} = 1, \text{pharmaceutical} = 1, \text{PayPal} = 0 \mid \text{spam})$
$\approx p(\text{<your name>} = 1 \mid \text{spam}) * p(\text{pharmaceutical} = 1 \mid \text{spam}) * p(\text{PayPal} = 0 \mid \text{spam})$
$= (1/6) * (5/6) * (2/6) \approx 0.0463$

$p(\text{<your name>} = 1, \text{pharmaceutical} = 1, \text{PayPal} = 0 \mid \text{not spam})$
$\approx p(\text{<your name>} = 1 \mid \text{not spam}) * p(\text{pharmaceutical} = 1 \mid \text{not spam}) * p(\text{PayPal} = 0 \mid \text{not spam})$
$= (1) * (1/4) * (3/4) = 0.1875$

$\hat{x}$ is likely 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.

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

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.) lunar
2. Which words are present in training example 501? car, fact, gun, video
3. Which newsgroup name does training example 501 come from? 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`.
Naive Bayes (ours) validation error: 0.188
Naive Bayes (sklearn) validation error: 0.187

```
def fit(self, X, y):
N, D = X.shape
# Compute the number of class labels
C = self.num_classes
# Compute the probability of each class i.e p(y==c)
counts = np.bincount(y)
p_y = counts/N
# Compute the conditional probabilities i.e.
# p(x(i,j)=1 | y(i)==c) as p_xy
# p(x(i,j)=0 | y(i)==c) as 1-p_xy
p_xy = 0.5 * np.ones((D, C))
for i in range(C):
    for j in range(D):
        x = 0
        for k in range(N):
            if X[k, j] == 1 and y[k] == i:
                x = x + 1
        p_xy[j, i] = x/counts[i]
# print(p_xy)
self.p_y = p_y
self.p_xy = p_xy
```

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 $\log(ab) = \log(a) + \log(b)$,

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

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?
Cost: O(dtk)

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).

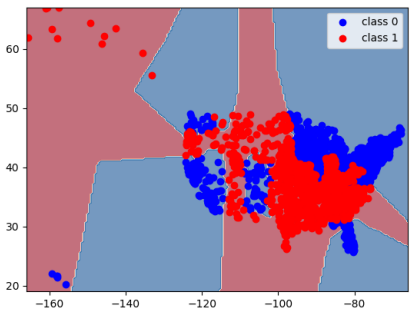
Fill in the `predict` function in `knn.py` so that the model file implements the k-nearest neighbour prediction rule. You should Euclidean distance, and may numpy's `sort` and/or `argsort` functions useful. You can also use `utils.euclidean_dist_squared`, which computes the squared Euclidean distances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

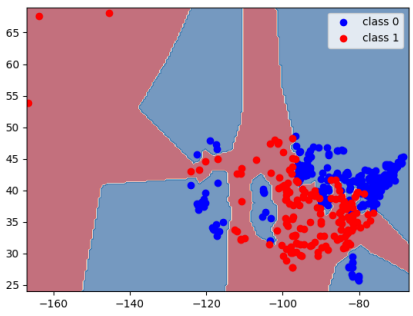
1. Write the `predict` function. Include this code in your GradeScope submission.

```
def predict(self, Xtest):
    T, D = Xtest.shape
    N, D = self.X.shape
    dists = utils.euclidean_dist_squared(self.X, Xtest)
    sortedDists = np.argsort(dists, axis=0)
    y_pred = np.empty(T)
    for ti in range(T):
        y_pred[ti] = stats.mode(self.y[sortedDists[:self.k, ti]])[0][0]
    print(y_pred)
    return y_pred
```

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?
   k=1 training error: 0.000, k=1 testing error: 0.065
   k=3 training error: 0.028, k=3 testing error: 0.066
   k=10 training error: 0.072, k=10 testing error: 0.097
   All of these numbers are much smaller compared to what I 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.
   my implementation: scikit-learn implementation:





4. Why is the training error 0 for $k = 1$? When $k = 1$, the model is likely overfitting and overly complex. This happens because since we are only finding the nearest neighbour for each training example, each example is labelled correctly so that the error is 0.

5. If you didn't have an explicit test set, how would you choose k? You can choose k with cross-validation. The model can be trained on 80% of the data and tested on the remaining 20%. This can be repeated to compute a mean error and finally decide on what k should be.

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.

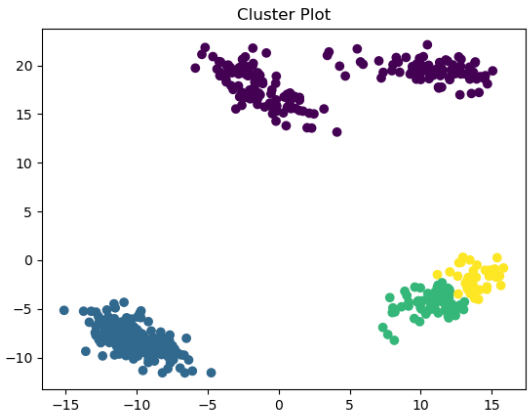
If you run `python main.py -q 4` it will fit a deep `DecisionTree` using the information gain splitting criterion. You will notice that the model overfits badly.

1. Why doesn't the random tree model have a training error of 0?
   The random tree is fit to a bootstrap sample of data. Hence, some examples are duplicated and some are missing, causing a non-zero training error.
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.
   Decision tree info gain: Training error-0.000, Testing error-0.367
   Random tree: Training error-0.193, Testing error-0.561
   Random forest: Training error-0.000, Testing error-0.201
   Yes, the results are what I expected. Random forests average the results of multiple (in this case, 50) random trees. Therefore, the testing error must be lower than a single random tree.
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.
   scikit-learn's implementation runs much faster. When it's nestimators (number of trees) is set to 50, it has a slightly better test error at 0.159. My implementation had a test error of 0.163. Another trial resulted in 0.174 (my implementation) and 0.163 (scikit-learn).

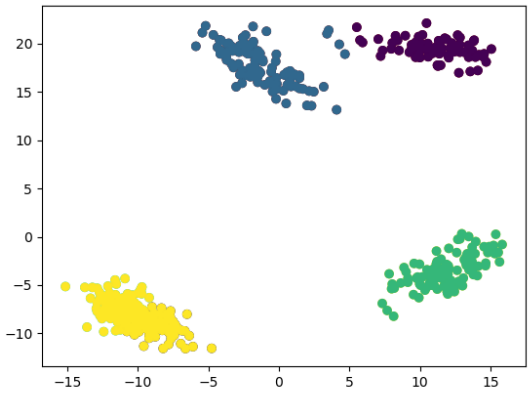
[1] The notation $\lfloor x \rfloor$ means the "floor" of x, or "x rounded down". You can compute this with `np.floor(x)` or `math.floor(x)`.

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},

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

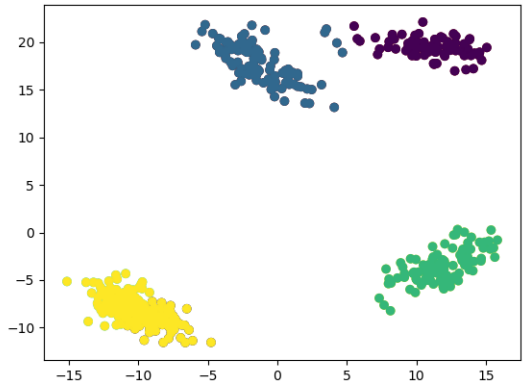
where y_i is the index of the closest mean to x_i. This is a natural criterion because the steps of k-means alternately optimize this objective function in terms of the w_c and the y_i values.

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.

```
def error(self, X):
    N, D = X.shape
    prediction = self.predict(X)
    error = 0
    for i in range(0, N):
        mean = self.means[prediction[i],]
        xi = X[i,]
        error += (xi[0]-mean[0])**2 + (xi[1]-mean[1])**2
    print(error)
    return error
```

2. What trend do you observe if you print the value of this error after each iteration of the k-means algorithm?
   The error may or may not change after each iteration, staying the same or fluctuating both higher and lower.

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.
   `n_clusters` is the number of clusters that the model should generate.
   `init` defines how the model should choose the initial means (e.g. randomly, smartly, or based on the shape of the ndarray).
   `n_init` is the number of times the algorithm will run with different means.
   `max_iter` is the max number of iterations of the algorithm for a single run.

5.2 Selecting k in k-means

Rubric: {reasoning:5}

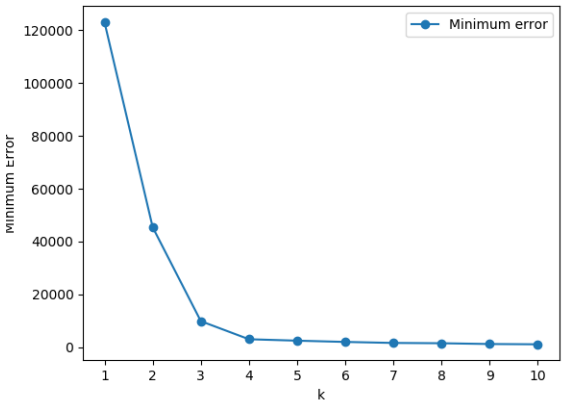
We now turn to the task of choosing the number of clusters k.

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

The k that would minimize the error function would always be $k = 1$, where all data points are clustered into one group, resulting in no classification error. Hence, this method would not work.

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

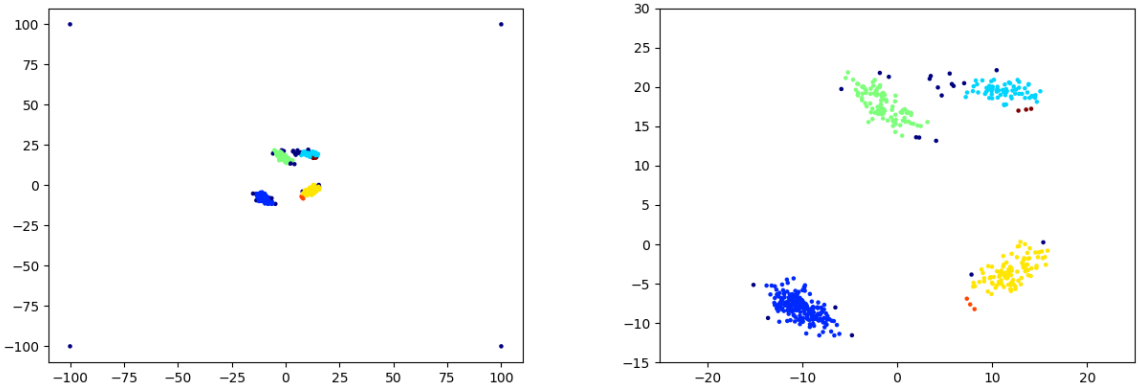


4. The *elbow method* for choosing k consists of looking at the above plot and visually trying to choose the k that makes the sharpest "elbow" (the biggest change in slope). What values of k might be reasonable according to this method? Note: there is not a single correct answer here; it is somewhat open to interpretation and there is a range of reasonable answers.
Either $k = 3$ or $k = 4$. The angles between the line segments seem to be about the same.

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.
   `eps=3`, `minPts=3`
2. 3 clusters (merging the top two, which also seems like a reasonable interpretation).
   `eps=5`, `minPts=3`
3. 2 clusters.
   `eps=15`, `minPts=3`
4. 1 cluster (consisting of the non-outlier points).
   `eps=20`, `minPts=3`

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?
   A boxplot is better for discrete data. Visualization also helps readers to easily and quickly have a overall understanding of the data. Boxplots also show outliers.
2. What is a reason that the the data may not be IID in the email spam filtering example from lecture?
   It is possible that the emails are not truly independent. Due to subscriptions, the types of emails you are sent may depend on the emails you have already received.
3. What is the difference between a validation set and a test set?
   The validation set is a subset of the training set. The test set is not part of the training set.
4. Why can't we (typically) use the training error to select a hyper-parameter?
   This would result in biases when testing the test set.
5. What is the effect of n on the optimization bias (assuming we use a parametric model).
   As n increases, optimization bias increases.
6. What is an advantage and a disadvantage of using a large k value in k-fold cross-validation.
   As k increases, the model becomes more accurate, but more expensive.
7. Why can we ignore $p(x_i)$ when we use naive Bayes?
   Naive Bayes assumes that all features x_i are conditionally independent given a label y_i.
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. parameter
   (b) Our estimate of $p(x_{ij} \mid y_i)$ for some x_{ij} and y_i. parameter
   (c) The value β in Laplace smoothing. 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.
   Given a small k, approximation error is high and training error is low, meaning the model is likely overfitting. Given a large k, approximation error is lower and training error is higher.

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?
    We can add transformed audio data during the training phase. Some examples: adjusting pitch up and down, slowing down or speeding up the audio, or varying the volume.

11. Both supervised learning and clustering models take in an input x_i and produce a label y_i. What is the key difference?
    Clustering models are unsupervised learning techniques, meaning there are no explicit target labels y_i. In supervised learning, the labels are predefined.

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)?
    Yes, using a validation set would reduce the chance of overfitting.

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?
    No, KNN may produce non-convex regions.