Joshua Sloan
University of Wyoming
jsloan3@uwyo.edu

Homework 3
COSC 5010 - Data Mining

1 Problems from the Book

1.1 Chapter 5 - Problem 17

a)

For ROC curve generation, see external ch5p17.py file →

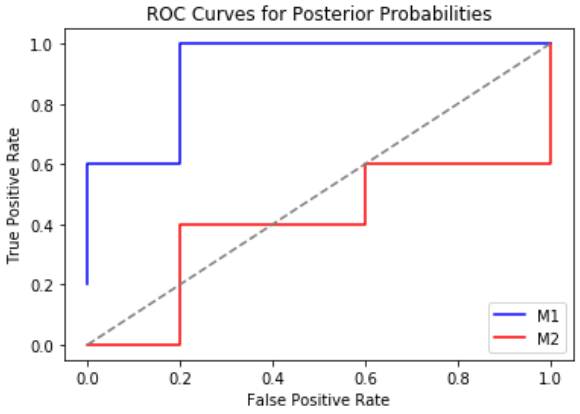

Figure 1: ROC Curves

From the plot, we can see that M_1 is better given that it has a greater area under the curve. Furthermore, M_2 is shown to be accurate less than 50% of the time, making it worse than a random guess.

Precision = $\frac{TP}{TP+FP}$

Recall = $\frac{TP}{TP+FN}$

F-measure = $\frac{2TP}{2TP+FP+FN}$

b)

Precision = 0.75

Recall = 0.6

F-measure = 0.6667

(See python code for implementation)

c)

Precision = 0.5

Recall = 0.2

F-measure = 0.2857

(See python code for implementation)

Based on the resulting F-measure scores, M_1 is still the better model. This is consistent with the ROC curve.

d)

Precision = 0.5556

Recall = 1.0

F-measure = 0.7143

(See python code for implementation)

Based on the resulting F-measure scores, setting the threshold equal to 0.1 provides better results than a threshold of 0.5. However, in regard to the ROC curve if you look at the False Positive Rate compared to the Recall (i.e. True Positive Rate) for both threshold values, you can see that at 0.5: FPR = 0.2 and TPR = 0.6 whereas at 0.1: FPR = 0.8 and TPR = 1.0 (again, see python implementation). With FPR = 0 and TPR = 1 being the optimal scores, you can argue that the threshold value of 0.5 is better, given that the distance to the optimum value is lower than that at 0.1. For this reason, I would prefer to use the threshold value of 0.5 unless I know that False Positive errors are of zero concern for our model and we only need to ensure 100% accuracy of True Positive predictions, in which case I would go with 0.1.

1.2 Chapter 5 - Problem 23

a)

Given the large volume of noise attributes relative to the number of distinguishing attributes, I do not think that kNN will work well. Naive Bayes may work well given that the distinguishing attributes will likely have more predictive weight than the noise attributes. Similarly, Decision Tree should also do well as it is likely that the nodes will be selected from distinguishing attributes (which appear to do a pretty good job of predicting class label).

b)

There appears to be a high attribution correlation for class label. In this regard, we are not able to confidently make the independence assumption for attributes required for Naive Bayes. As a result I do not think that it will perform very well. On the other hand, a Decision Tree should have greater success using these tightly coupled attributes and kNN could also work well given that there is some semblance of clustering respective to the attributes (although I can imagine there would be some inaccurate regions) in conjunction with a small number of noise attributes relative to the number of distinguishing attributes.

c)

We can see that there exists a limited number of noise attributes and distinguishing attributes hint at decently defined clusters, such that kNN should perform alright (I do not think that clusters will be very well defined nor separated, but exist as to allow for some degree of accuracy for decision making). Naive Bayes should perform well given that there is a majority appearance in each distinguishing attribute set relative to the class label. These sort of probabilistic metrics inherent to the data tend to work well for Naive Bayes. Decision tree may also work well, but there are going to require a lot of decision splits in order to ensure that branches are not trying to determine membership exclusive to one distinguishing attribute set given that the errors are not too incredibly distinguishable based on one set alone (depending on how many attributes belong to each set).

d)

Given the distribution of classes, I would say that kNN will work pretty well, given that you can see the data is generally separated into 20 clusters. There might be some predictive issues near the region boundaries, but overall it should do a pretty good job. A decision tree will work, but it is going to require a lot of split points to narrow down a class label prediction. Naive Bayes is not likely to work very well given that there is exists an overlap pattern respective to each attribute dimension and as a result we are not able to make the required assumption for Naive Bayes that attributes are independent of each other.

e)

From the distribution, I would imagine kNN will work quite well since the data already appears to belong to 2 well formed clusters. I decision tree can also work, however I would recommend using an oblique decision split given the degree of value overlap respective to both the X and Y attributes. Given this overlap, Naive Bayes would not perform as well for points belonging to the overlap region for each attribute.

f)

Given the distribution, I would say the kNN would likely work the best. A decision tree could also work pretty well, however I would recommend using an oblique decision split using both attribute X and attribute Y since there is a large overlap of distribution respective to any one dimension. Naive Bayes would not work well given that my previous statement highlights how the attributes are not independent of one another. As a result, there is no way to make a good probabilistic decision based solely on the value of one single attribute and combining poor probabilistic decisions is unlikely to yield good predictive results.

1.3 Chapter 8 - Problem 7

Answer: (c) More centroids should be allocated to the denser region.

Explanation: Given that the points are equally distributed between the 2 regions, we will want to allocate more centroids towards the denser region because more points will affect the squared error due to a lack in an ideal centroid placement. This can be justified because we have well separated regions, meaning that it is unlikely that a point from a different region will be clustered to a centroid of another region (expect the extreme case when a region receives zero of the K centroids) However, the wording indicates that both regions will receive at least 1 centroid, allowing us to fall back on the former statement about how region density affects the squared error.

1.4 Chapter 8 - Problem 11

What does it mean is the SSE for one variable is low for all clusters?
If the SSE for a single variable is low for all clusters than it means that there is not much variation between the range of values the variable can take and as a result, it is not a very good indicator of which cluster a data point should belong to.

Low for just one cluster?
If the SSE of just one cluster is low, then that means that this variable is a good indicator of which value a cluster should belong to, as it helps determine which data points should belong to that cluster.

High for all clusters?
If the SSE for a single variable is high for all clusters, this is a good indication that this variable is not a useful indicator of cluster membership, as this variable might have a sporadic or noisy distribution.

High for just one cluster?
If the SSE of just one cluster is high, then this means that this variable is not a very good indicator of the membership to the specific cluster.

How could you use the per variable SSE information to improve your clusters?
By looking at variables with both low or high SSE across all clusters, we know that these are not good indicators of cluster membership and can therefore be ignored (or at least given reduced weight) when determining which cluster a data point should belong to. Additionally, we can check the variables with high SSE for just one cluster in relation to other variable specific SSE for that cluster. An overabundance of one or the other can help determine whether or not that variable is a good indicator for that specific cluster.

Once we have determined which variables are the good indicators of cluster membership and which to ignore (or weight-reduce), we can cluster based on the total SSE after responding to the bad variables, such that we allow the good variables to work together to determine cluster membership.

2 Clustering

2.1 Given k clusters and their respective cluster sizes $s_1, s_2 ... s_k$, what is the probability that two random (with replacement) data vectors (from the clustered dataset) belong to the same cluster?

Given that we know samples are taken with replacement, that means that the two selected data vectors are independent of one another, such that:

P(2v$\in C_i$) = $\frac{1}{s_T} * \frac{1}{s_T} + \frac{2}{s_T} * \frac{2}{s_T} + ... + \frac{k}{s_T} * \frac{k}{s_T}$, where s_T = total number of data vectors

= $\frac{1^2}{s_T^2} + \frac{2^2}{s_T^2} + ... + \frac{k^2}{s_T^2}$

= $\frac{1^2+2^2+...+k^2}{s_T^2}$

= $\frac{1^2+2^2+...+k^2}{(1+2+...+k)^2}$

= $\frac{k(k+1)(2k+1)/6}{(k(k+1)/2)^2}$ (by the sum of squares and triangular number series)

= $\frac{2(2k+1)}{3k(k+1)}$

$\therefore$ given the value of k there is a $\frac{2(2k+1)}{3k(k+1)}$ chance that two randomly selected data vector (with replacement) will belong to the same cluster. □

2.2 Now assume you are given this probability (you don't have s_i's and k), and the fact that clusters are equally sized, can you find k? This gives you an idea for predicting the number of clusters in a dataset.

Yes, you can find value of k directly from the probability itself.

Given that we know that all clusters are the same size, we have:

P(2v$\in C_i$) = $\frac{s_1^2+s_2^2+...+s_n^2}{s_T^2}$, where $s_1 = s_2 = ... = s_n$ (only in size, they remain indicative of different clusters) by following a similar reduction to that of the previous question, given that the samples are independent.

We can then utilize the fact that we know that the total number of data vectors is equal to $n * k$, where k is the number of clusters and n is the number of data vectors belonging to each cluster, such that:

P(2v$\in C_i$) = $\frac{k_1^2+k_2^2+...+k_n^2}{(n*k)^2} = \frac{n(k^2)}{(n*k)^2}$.

Because each of the k clusters has the same number of n data vectors, we have $n = k$, such that:

for k = 1: P(2v$\in C_i$) = $\frac{1*(1^2)}{(1*1)^2} = \frac{1}{1} = 1$
for k = 2: P(2v$\in C_i$) = $\frac{2*(2^2)}{(2*2)^2} = \frac{2*4}{4^2} = \frac{8}{16} = \frac{1}{2}$
for k = 3: P(2v$\in C_i$) = $\frac{3*(3^2)}{(3*3)^2} = \frac{3*9}{9^2} = \frac{27}{81} = \frac{1}{3}$
for k = 4: P(2v$\in C_i$) = $\frac{4*(4^2)}{(4*4)^2} = \frac{4*16}{16^2} = \frac{64}{256} = \frac{1}{4}$
for k = 5: P(2v$\in C_i$) = $\frac{5*(5^2)}{(5*5)^2} = \frac{5*25}{25^2} = \frac{125}{625} = \frac{1}{5}$
$\vdots$
etc.

Therefore, to find our value of k, we simply need to set $\frac{k(k^2)}{(k*k)^2}$ equal to the probability provided and solve for it.

Another method we can extract from the aforementioned pattern is to simply find k directly from the given probability. In the case that the probability is represented as a fraction, we simply need to reduce it to the lowest value it can take. The value in the denominator is our corresponding value for k. In the case that the probability is represented as a decimal, we simply to find its equivalent fraction and reduce it. Again, the value in the denominator is our value of k. Given the properties above (i.e. sampling with replacement and equally sized clusters), our probability will always reduce to the form $\frac{1}{k}$. □

2.3 Give an example of a dataset consisting of 4 data vectors where there exist two different optimal (minimum SSE) 2-means (k-means, k=2) clustering of the dataset.

Consider the following dataset:

X-value	Y-value
-1	1
1	1
-1	-1
1	-1

such that we can denote the four 2-D data vectors as:

A: (-1, 1)
B: (1,1)
C: (-1,-1)
D: (1,-1)

(labeled A,B,C and D for convenience of explaining).

Plotting the resulting points will create the shape of a square with vertices at the points denoted by the corresponding x and y values (see Figure 2 below).

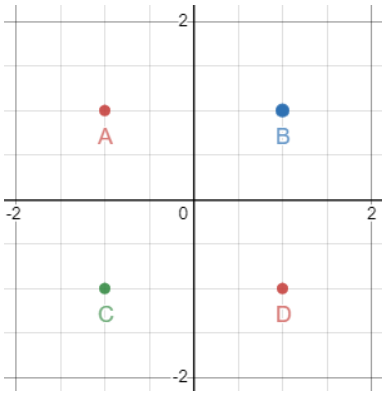

Figure 2: Plotted Dataset

- **Calculate the optimal SSE value for your example.**

  The optimal SSE value is the minimum SSE obtainable, which for our example can be obtained by either C_1 = {A,B} and C_2 = {C,D}, such that:

  SSE = $\sum_{i=1}^{2} \sum_{x \in C_i}$ dist^2(m_i,x),

  where x is a data point in cluster C_i, m_i is the representative point for cluster C_i and dist() is Euclidean distance.

  Thus, given m_1 = (0,1) and m_2 = (0,-1),

  SSE = $\sqrt{(-1-0)^2 + (1-1)^2}^2 + \sqrt{(1-0)^2 + (1-1)^2}^2 + \sqrt{(-1-0)^2 + (-1-(-1))^2}^2 + \sqrt{(1-0)^2 + (-1-(-1))^2}^2$

  $= \sqrt{1}^2 + \sqrt{1}^2 + \sqrt{1}^2 + \sqrt{1}^2$

  $= 1 + 1 + 1 + 1 = 4$

or from C_1 = {A,C} and C_2 = {B,D}, such that:

Given m_1 = (-1,0) and m_2 = (1,0),

SSE = $\sqrt{(-1-(-1))^2+(1-0)^2}^2+\sqrt{(-1-(-1))^2+(-1-0)^2}^2+\sqrt{(1-1)^2+(1-0)^2}^2+\sqrt{(1-1)^2+(-1-0)^2}^2$

$= \sqrt{1}^2+\sqrt{1}^2+\sqrt{1}^2+\sqrt{1}^2$

= 1 + 1 + 1 + 1 = 4

where any other clustering assignment will give a greater SSE value (clustering with 3 points in one cluster and the remaining in the other yields an SSE ≈ 5.334).□

- **In general, how should datasets geometrically look like so that we have more than one optimal solution?**

  To have more than one optimal solution, datasets should be symmetric in their respective dimensional space.

- **What defines the number of optimal solutions?**

  The number of optimal solutions is determined by the number of clusters which are able to have symmetric points to all other clusters. In this manner, this is determined to be N/k, where N is the total number of data vectors and k is the number of clusters (only when the dataset is symmetric).

  In our case, the number of optimal solutions = 4/2 = 2, which checks out (as all other possible cluster variations give greater SSE).

- **This problem provides an example of situations where k-means does not necessarily converge to the same optimal all the time.**

  True. The optimal SSE that is selected is determined from which initialized points are selected as the starting cluster centroids in the k-means clustering algorithm (under the aforementioned conditions), as well as which cluster a data vector is arbitrarily assigned to the case of a tie in distance to the centroid.

3 KDD Cup 2009

The kddcup99.arff file was loaded into Weka and the following two commands were ran on the data:

18:35:55: Command: weka.filters.unsupervised.instance.Randomize -S 42
(42 is random seed default value)

18:38:32: Command: weka.filters.unsupervised.instance.RemovePercentage -P 90.0
(Remove 90% of the data such that 10% is left)

The result was saved as altered_kddcup99.arff (attached in the question 3 zip file).

Next, a Naive Bayes, Decision Tree (J48) and 10-NN (IB10) classifier were trained on the new data using 10-fold cross validation and result buffers were stored in NaivesBayesResults, J48Results and IB10Results respectively (again, see question 3 zip file).

4 Text Clustering

See external question 4 zip file.