B656 DATA MINING

Homework 2

25 February 2018

Chuck Jia

Problem 1

In this problem, we assume that we follow the same procedures to draw ROC curves as on page 298 in *Introduction to Data Mining* by Tan, Steinbach, and Kumar.

Since there are more than one way to construct a ROC curve and there are different ways in counting thresholds, to aid our argument, we set up the following rules

(1) For a given threshold, we assign all data points **greater than or equal to** the threshold to the positive class. All other data points are assigned to the negative class.

(2) We assume that $(0,0)$ and $(1,1)$ are also drawn by two thresholds. And these two thresholds are **always included** in our counts of necessary points.[1]

Also, we here calculate the maximum and minimum thresholds needed to actually construct the correct ROC curve. We do not consider cases where with certain number of points we can calculate the correct area but cannot fully draw the exact ROC curve.

The general process in constructing a ROC curve is as follows. For more details, see the descriptions in the textbook.

- We sort the p_i in increasing order.
- We start with the smallest p_i as threshold and assign all data points to their classes by rule (1) above. We draw that corresponding point in the ROC curve.
- Then we use the next smallest p_i as threshold and assign data points to classes. We draw that corresponding point in the ROC curve.
- We repeat this process using p_i's as thresholds, until all p_i's have been used.
- Then we use $\max\{p_i\}+1$ as threshold, and draw the point (0,0)

(a) We claim that the minimum number of thresholds is 2.

Proof. First we show that

$$\text{for any predictions } \{p_i\}\text{, we need to use at least 2 thresholds} \qquad \text{(Statement 1)}$$

- If we use $\min\{p_i\}-1$ as threshold, then by rule (1), all data points will be classified as positive. Thus, there are no false negatives and no true negatives. Therefore,

$$\text{true positive rate} = \frac{TP}{TP+FN} = \frac{TP}{TP+0} = 1, \qquad \text{false positive rate} = \frac{FP}{TN+FP} = \frac{FP}{0+FP} = 1$$

That is, the point (1,1) should be on the ROC curve.

[1] Alternatively, we can see in the proof below that a ROC curve always reaches these curves and thus we can simply ignore these two points. We will not use this alternative counting rule.

- If we use $\max\{p_i\}+1$ as threshold, then all data points will be classified as negative. Thus, there are no false positives and no true positives. Therefore,

$$\text{true positive rate} = \frac{TP}{TP+FN} = \frac{0}{0+FN} = 0, \qquad \text{false positive rate} = \frac{FP}{TN+FP} = \frac{0}{TN+0} = 0$$

  That is, the point (0,0) should be on the ROC curve.

Therefore, we showed that for any set of predictions, the ROC curve has at least two distinct points. Thus, we need to use at least 2 thresholds.

We now show that

there is a set of predictions $\{p_i\}$, for which we only need 2 thresholds to draw the ROC curve (Statement 2)

Let $p_i = 1$ for all i. Then we have the following observations.

- Any threshold less than or equal to 1 will classify all data points as positive. Therefore, using the same argument as in the proof for Statement 1

$$\text{true positive rate} = \frac{TP}{TP+FN} = \frac{TP}{TP+0} = 1, \qquad \text{false positive rate} = \frac{FP}{TN+FP} = \frac{FP}{0+FP} = 1$$

- Any threshold greater than 1 will classify all data points as negative. Therefore, using the same argument as before,

$$\text{true positive rate} = \frac{TP}{TP+FN} = \frac{0}{0+FN} = 0, \qquad \text{false positive rate} = \frac{FP}{TN+FP} = \frac{0}{TN+0} = 0$$

Therefore, there are only 2 possible pairs of (FPR, TPR), for which we only need 2 thresholds to calculate.

Now combining Statement 1, and Statement 2, we have shown that the minimum number of thresholds we need is 2.

(b) We claim that the maximum number of thresholds we need is

$$\text{maximum number of thresholds} = \begin{cases} 2\min(n_+, n_-) + 2, & \text{if } n_+ \neq n_- \\ n+1 = 2n_+ + 1 = 2n_- + 1, & \text{if } n_+ = n_- \end{cases}$$

Proof. Note that if we use all the p_i's as thresholds, we would be able to construct the correct ROC curve, as argued in the textbook. Therefore, to find the maximum number of thresholds, we only need to choose the maximum number of p_i's to be used as thresholds.

Let $\{\tilde{p}_i\}$ be the sorted $\{p_i\}$ in increasing order. That is, $\{\tilde{p}_i\} = \{p_i\}$ with $\tilde{p}_i \leq \tilde{p}_{i+1}$. Let $\tilde{x}_i$ and $\tilde{y}_i$ be the corresponding data points and labels.

Now we use $\tilde{p}_1, \tilde{p}_2, \cdots, \tilde{p}_n$ in order as thresholds.

For a threshold $\tilde{p}_i$, we consider the following 4 cases on the classes of $\tilde{x}_i$ and $\tilde{x}_{i-1}$, i.e. the values of $\tilde{y}_i$ and $\tilde{y}_{i-1}$. For convenience, we use $FPR_i, TPR_i, FP_i, TP_i, FN_i, TN_i$ to denote the rates and counts calculated using $\tilde{p}_i$ as threshold.

- Case 1: $\tilde{y}_{i-1} = \tilde{y}_i = -1$.

  Using $\tilde{p}_{i-1}$ as threshold gives

$$FPR_{i-1} = \frac{FP_{i-1}}{n_-}, \quad TPR_{i-1} = \frac{TP_{i-1}}{n_+} \tag{E1}$$

  Now we use $\tilde{p}_i$ as threshold. Using the new threshold only changes the classification for $\tilde{x}_{i-1}$ from positive to negative. Note that since $\tilde{y}_{i-1} = -1$ is negative to begin with, this change in classification does not change the true positive count TP, but decreases the false positive count FP by 1. Therefore,

$$FPR_i = \frac{FP_i}{n_-} = \frac{FP_{i-1} - 1}{n_-}, \quad TPR_i = \frac{TP_i}{n_+} = \frac{TP_{i-1}}{n_+} = TPR_{i-1} \tag{E2}$$

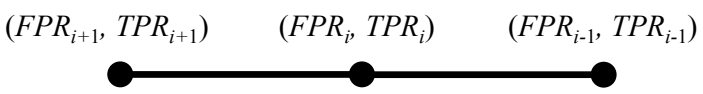
  Next, we use $\tilde{p}_{i+1}$ as threshold. Similar as before, the new threshold only changes the classification for $\tilde{x}_i$ from positive to negative. Since $\tilde{y}_i = -1$ is negative to begin with, this change in classification does not change the TP count, but decreases the FP count by 1. Thus again,

$$FPR_{i+1} = \frac{FP_{i+1}}{n_-} = \frac{FP_{i-1} - 2}{n_-}, \quad TPR_{i+1} = \frac{TP_{i+1}}{n_+} = \frac{TP_i}{n_+} = TPR_i = TPR_{i-1} \tag{E3}$$

  Note that (E1), (E1), and (E1) gave 3 points on the ROC curve:

$$(FPR_{i-1}, TPR_{i-1}), \quad (FPR_i, TPR_i) = (FPR_i, TPR_{i-1}) \quad (FPR_{i+1}, TPR_{i+1}) = (FPR_{i+1}, TPR_{i-1}),$$

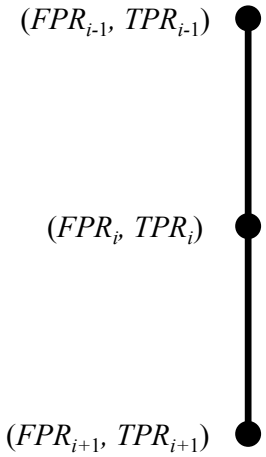
  They all have the same vertical coordinate and therefore form a horizontal line.



  In this case, the middle point is not necessary as points will be connected by straight lines. That is, we do not need to use $\tilde{p}_i$ as a threshold.

- Case 2: $\tilde{y}_{i-1} = \tilde{y}_i = 1$.

  A similar analysis with case 1 shows that the 3 ROC points derived using the thresholds $\tilde{p}_{i-1}$, $\tilde{p}_i$, and $\tilde{p}_{i+1}$ form a horizontal line and therefore the middle point is not necessary.



- Case 3: $\tilde{y}_{i-1} = -1$ and $\tilde{y}_i = 1$. Using $\tilde{p}_{i-1}$ as threshold gives

$$FPR_{i-1} = \frac{FP_{i-1}}{n_-}, \quad TPR_{i-1} = \frac{TP_{i-1}}{n_+} \tag{E4}$$

Now we use $\tilde{p}_i$ as threshold. Using the new threshold only changes the classification for $\tilde{x}_{i-1}$ from positive to negative. Note that since $\tilde{y}_{i-1} = -1$ is negative to begin with, this change in classification does not change the true positive count TP, but decreases the false positive count FP by 1. Therefore,

$$FPR_i = \frac{FP_i}{n_-} = \frac{FP_{i-1} - 1}{n_-} = FPR_{i-1} - \frac{1}{n_-}, \quad TPR_i = \frac{TP_i}{n_+} = \frac{TP_{i-1}}{n_+} = TPR_{i-1} \tag{E5}$$

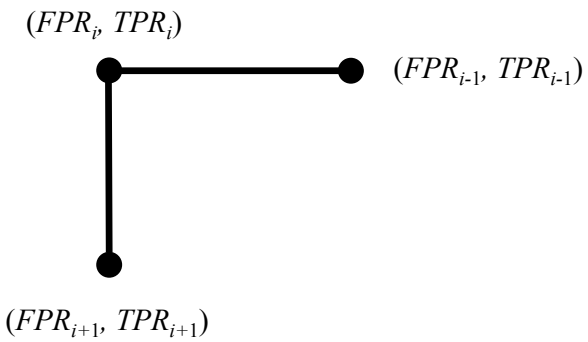
Next, we use $\tilde{p}_{i+1}$ as threshold. Again new threshold only changes the classification for $\tilde{x}_i$ from positive to negative. Since $\tilde{y}_i = 1$ is positive to begin with, this change in classification decreases the TP count by 1, but does not change the FP count.

$$FPR_{i+1} = \frac{FP_{i+1}}{n_-} = \frac{FP_i}{n_-} = FPR_i, \quad TPR_{i+1} = \frac{TP_{i+1}}{n_+} = \frac{TP_i - 1}{n_+} = TPR_i - \frac{1}{n_+} \tag{E6}$$

Note that (E4), (E5), and (E6) gave 3 points on the ROC curve:

$$(FPR_{i-1}, TPR_{i-1}), \quad (FPR_i, TPR_i) = (FPR_{i-1} - \tfrac{1}{n-}, TPR_{i-1}), \quad (FPR_{i+1}, TPR_{i+1}) = (FPR_i, TPR_i - \tfrac{1}{n+}),$$

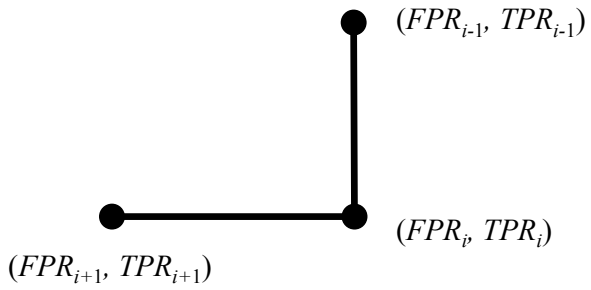
These 3 points will form the following structure:



In this case, $\tilde{p}_i$ is a necessary threshold to construct the correct ROC curve.

- Case 4: $\tilde{y}_{i-1} = 1$ and $\tilde{y}_i = -1$.

  With similar analysis as in Case 3, we can see the 3 points calculated by using thresholds $\tilde{p}_{i-1}$, $\tilde{p}_i$, and $\tilde{p}_{i+1}$ form the following structure.



  Therefore, in this case, $\tilde{p}_i$ is again a necessary threshold to construct the correct ROC curve.

In summary, we saw from the above analysis that if $\tilde{y}_i = \tilde{y}_{i-1}$, then we can skip $\tilde{p}_i$ as threshold. But if $\tilde{y}_i \neq \tilde{y}_{i-1}$, then $\tilde{p}_i$ is a necessary threshold in order to correctly construct the ROC curve.

Then to achieve the maximum number of necessary threshold, we need the maximum number of $\tilde{y}_i$ with $\tilde{y}_i \neq \tilde{y}_{i-1}$.

(1) **Case of $n_- \neq n_+$.**

If $n_- < n_+$, then the number of pairs $(\tilde{y}_i, \tilde{y}_{i+1})$ with $\tilde{y}_i < 0$ and $\tilde{y}_{i+1} > 0$ is less than n_-, as each pair would require one data point in the negative class. Similarly, the number of pairs $(\tilde{y}_i, \tilde{y}_{i+1})$ with $\tilde{y}_i > 0$ and $\tilde{y}_{i+1} < 0$ is less than n_-. Therefore, there are at most $2n_-$ such pairs. This implies that we only need at most $2n_-$ thresholds from $\{\tilde{p}_i,\ 2 \leq i \leq n\}$. And clearly, we need $\tilde{p}_1$ and $\tilde{p}_n + 1$ as thresholds to calculate the two end points (1,1) and (0,0). That is,

$$\text{we only need at most } 2n_- + 2 \text{ thresholds in total}$$

Now we show that there is a $\{p_i\}$ that requires at least $2n_- + 2$ thresholds. If we have the following arrangement of classes, then we would need $2n_- + 2$ thresholds.

$$(\tilde{y}_1, \tilde{y}_2, \cdots, \tilde{y}_n) = (+1, -1, +1, -1, \cdots, +1, -1, +1, +1, +1, \cdots, +1)$$

That is, we start with $+1$ and keep alternating between $+$ and $-$ until we run out of -1. By the analysis above, we can see that

$$\text{this above instance requires at least } 2n_- + 2 \text{ thresholds}$$

Thus we proved that if $n_- < n_+$, then the maximum number of threshold required is $2n_- + 2$.

Using the same argument, we can show that if $n_+ < n_-$, then the maximum number of threshold required is $2n_+ + 2$. Therefore, we have shown

$$\text{if } n_+ \neq n_-, \text{ then then the maximum number of thresholds required is } 2\min(n_+, n_-) + 2$$

(2) **Case of $n_- = n_+$.**

If $n_- = n_+$, without loss of generality, assume $\tilde{y} = -1$. Then all negative $\tilde{y}_i$ but $\tilde{y}_1$ would be in at most two pairs of $(+1, 1)$, and $(-1, +1)$, and therefore account for $2(n_- - 1)$ necessary thresholds. $\tilde{y}_1$ accounts at most for two necessary thresholds too, as an end point and as one in the pair $(\tilde{y}_1, \tilde{y}_2)$. Adding the threshold $\tilde{y}_n + 1$, we have shown that at most we need $2(n_- - 1) + 2 + 1 = 2n_- + 1 = n + 1$ different thresholds.

Now if we have the following alternating arrangement of classes,

$$(\tilde{y}_1, \tilde{y}_2, \cdots, \tilde{y}_n) = (-1, +1, -1, +1, \cdots, -1, +1)$$

then by a similar analysis as above, we can see that we need at least $2n_- + 1 = n + 1$ thresholds. Therefore, we have shown that if $n_- = n_+$, then the maximum number of thresholds we need is $2n_+ + 1 = 2n_- + 1 = n + 1$.

In summary, we have shown that the maximum number of thresholds we need is

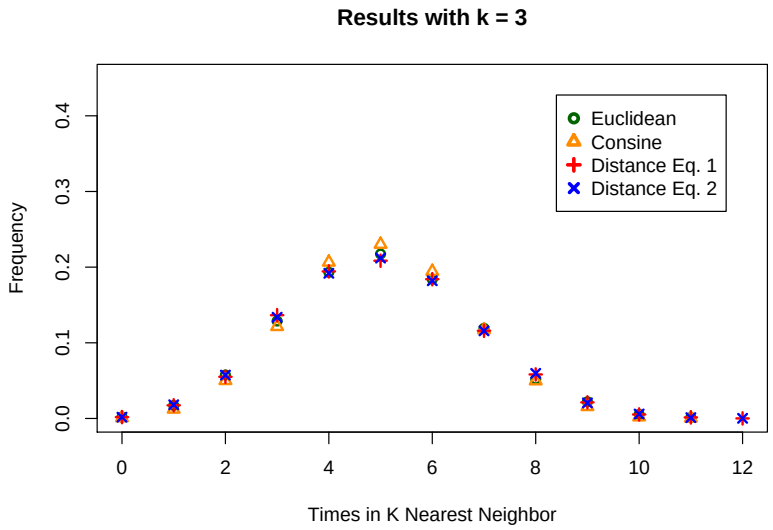
$$\text{maximum number of thresholds} = \begin{cases} 2\min(n_+, n_-) + 2, & \text{if } n_+ \neq n_- \\ n + 1 = 2n_+ + 1 = 2n_- + 1, & \text{if } n_+ = n_- \end{cases}$$

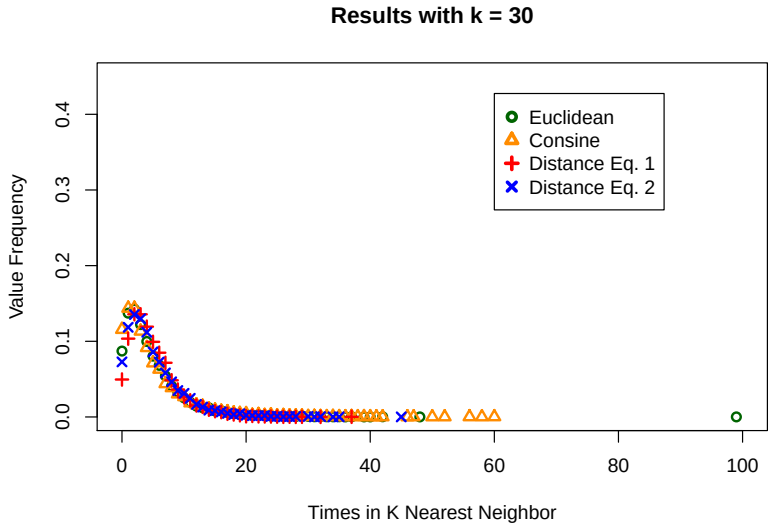
Problem 2

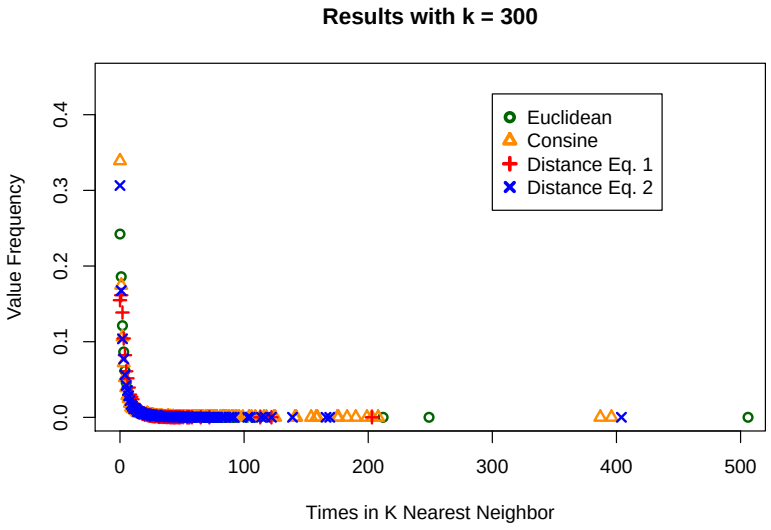
(a) First, we show the result with the uniform generators.

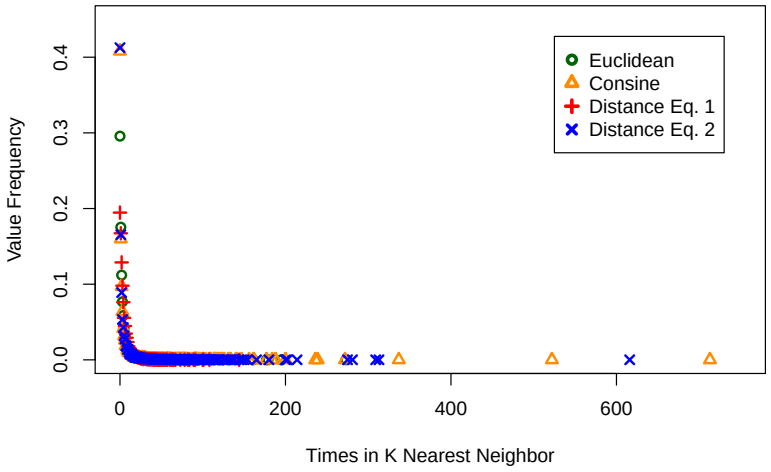
Here we provide two types of plots: dot plots and line plots on the same results. The common practice is to combine the two plots, but in our case there are advantages not to do so. It is easy to observe the differences from the line plots, while adding dots will make the task harder, as the density of dots is high at many interesting areas in the plot. However, there are some locations in the plot where the data point density is extremely low. For observations in those areas, it is convenient to use the dot plots.

Dot Plot: Uniform Generator

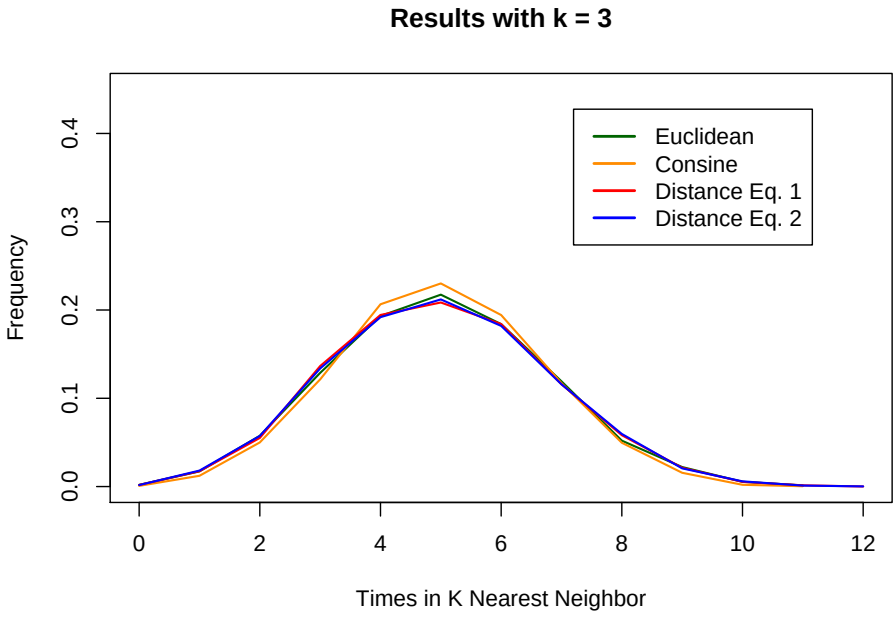


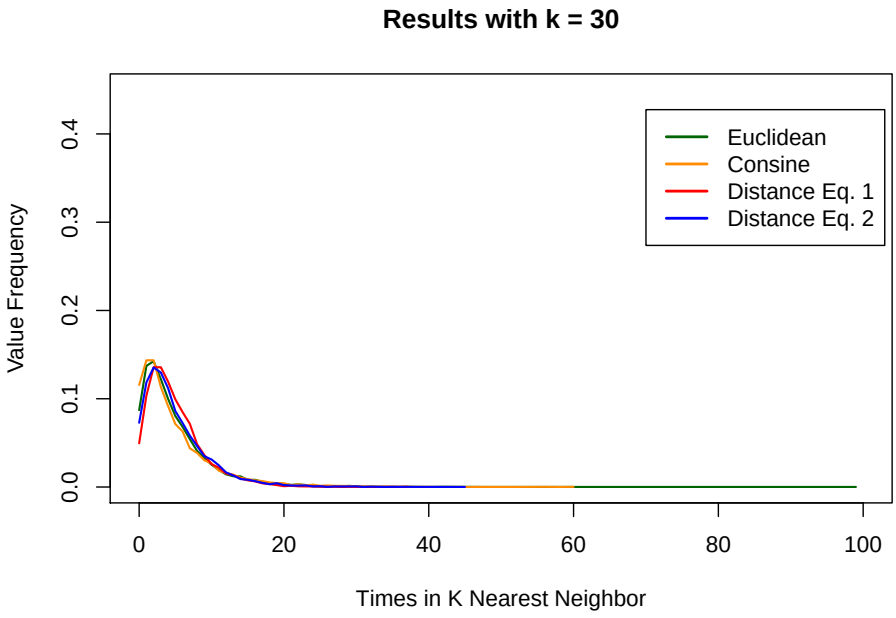


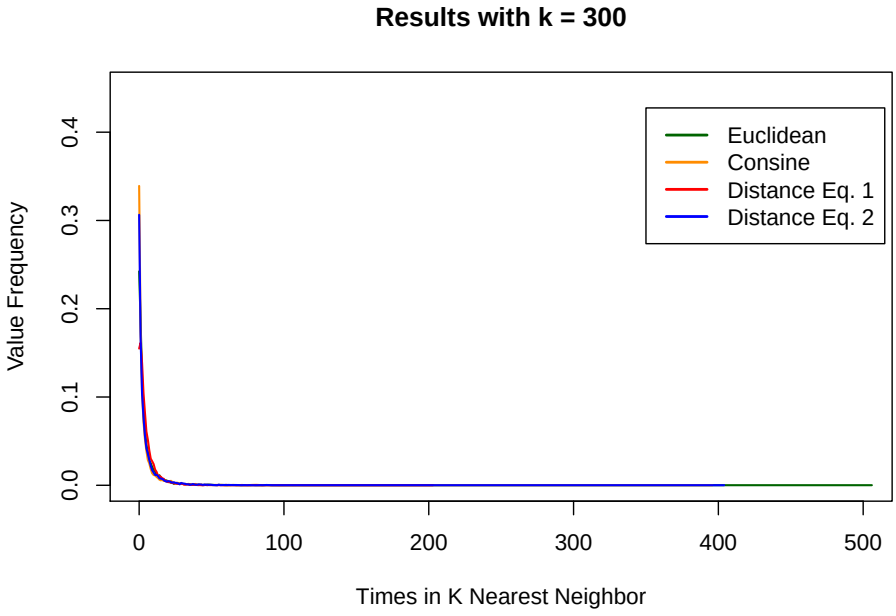

Results with k = 300
Euclidean
Consine
Distance Eq. 1
Distance Eq. 2
Value Frequency
0.0
0.1
0.2
0.3
0.4
0
100
200
300
400
500
Times in K Nearest Neighbor

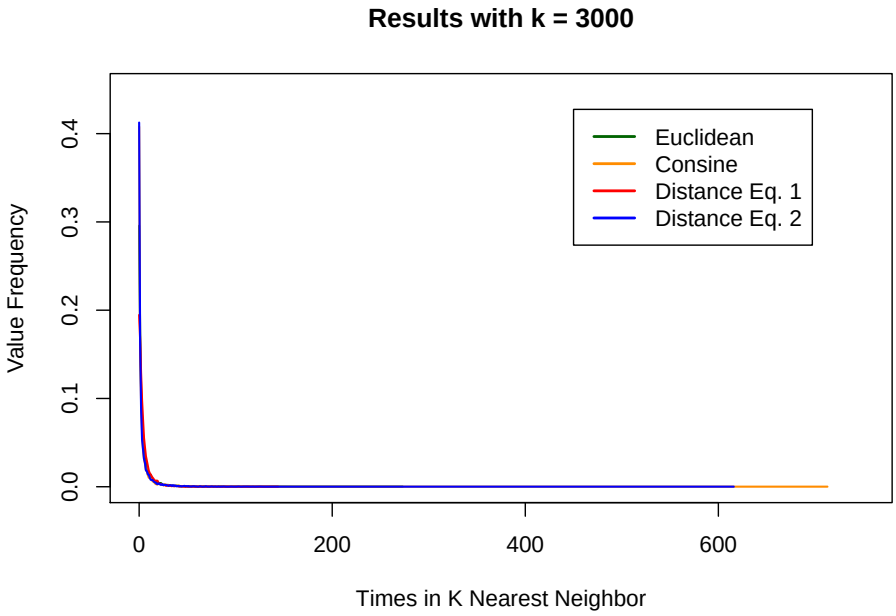

Results with k = 3000
Euclidean
Consine
Distance Eq. 1
Distance Eq. 2
Value Frequency
0.0
0.1
0.2
0.3
0.4
0
200
400
600
Times in K Nearest Neighbor

Line Plots: Uniform Generator





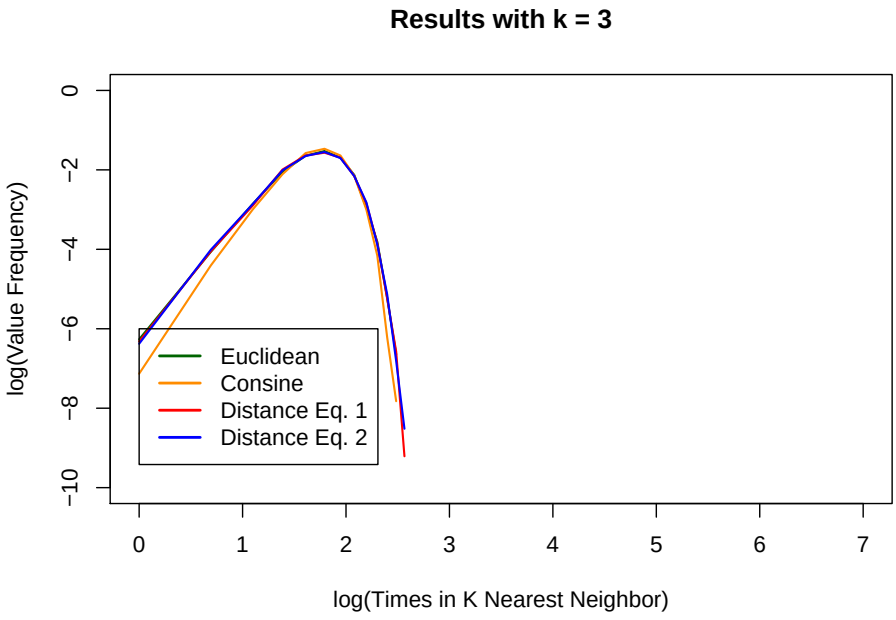


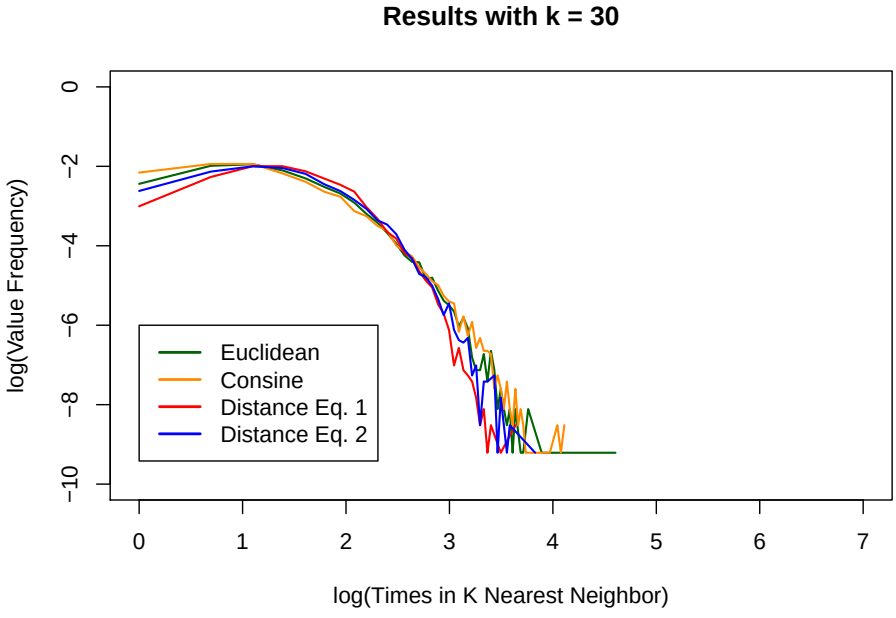


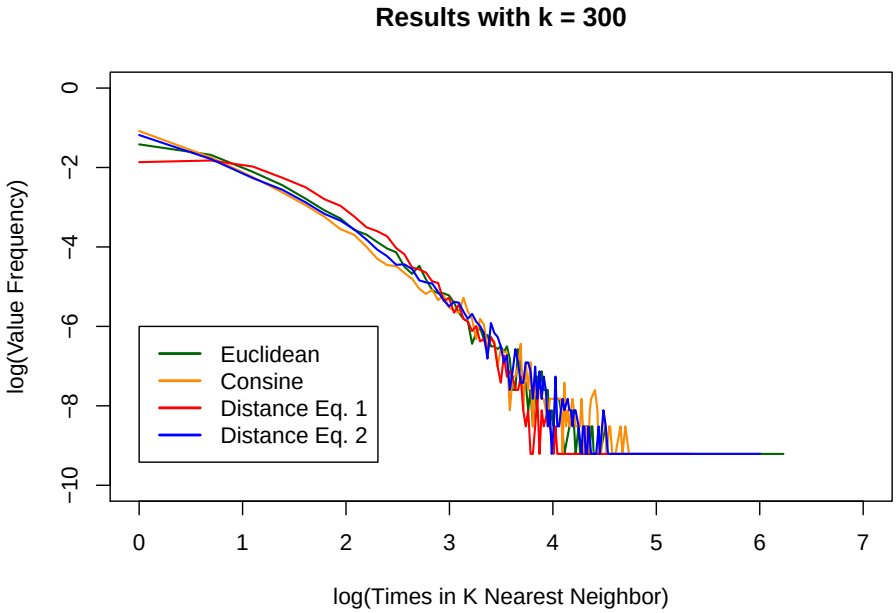
Now in order to observe potential patterns or properties, we also provide the log-log plots of the results.[2]

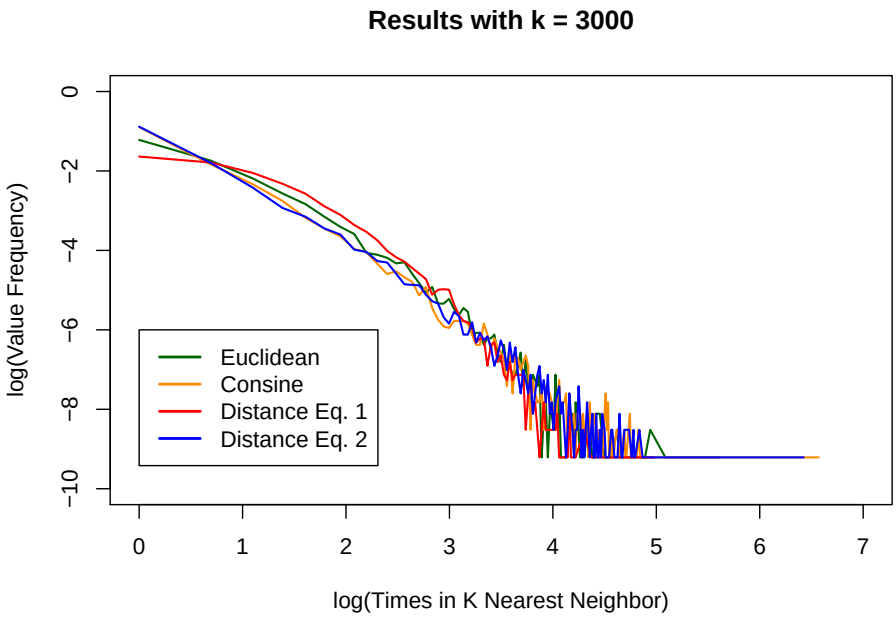
[2]Note that the x-axis is in fact $\log(n_5(x)+1)$ instead of $\log(n_5(x))$, in order to eliminate the $-\infty$ points.

Log-log Plots: Uniform Generator



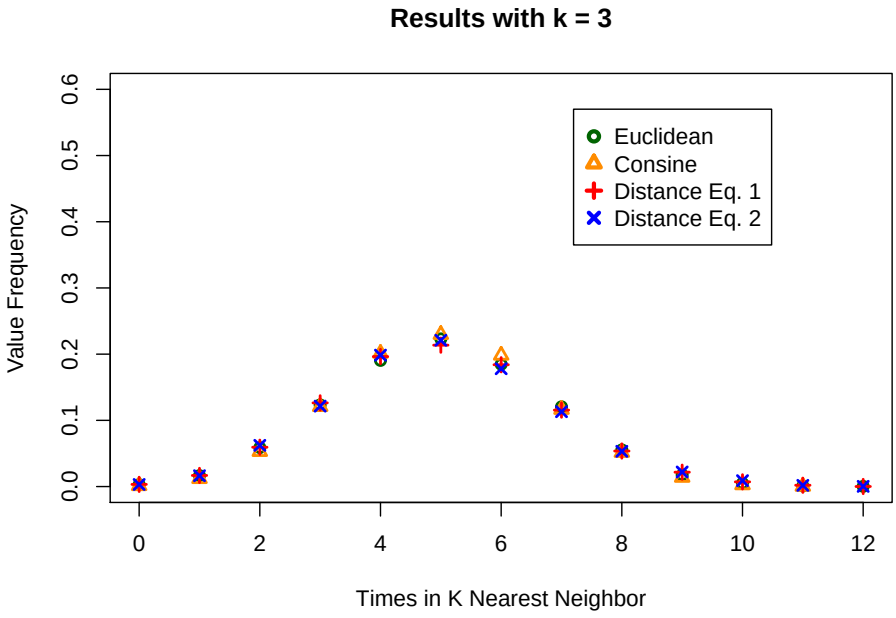


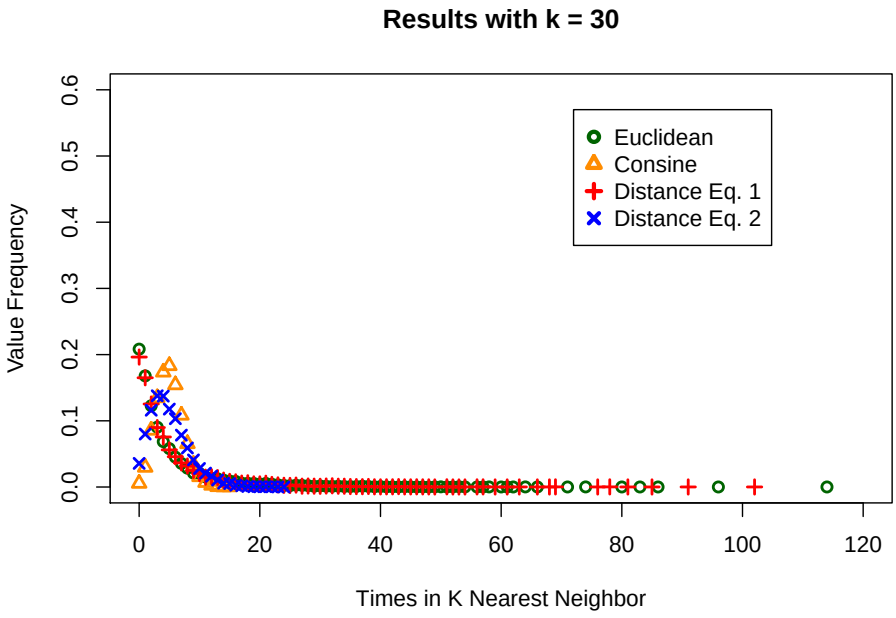


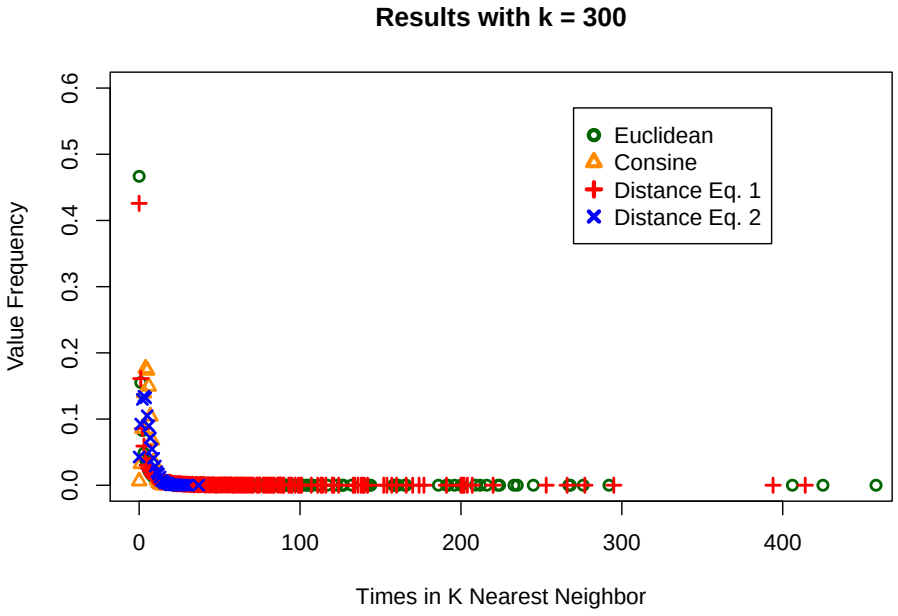


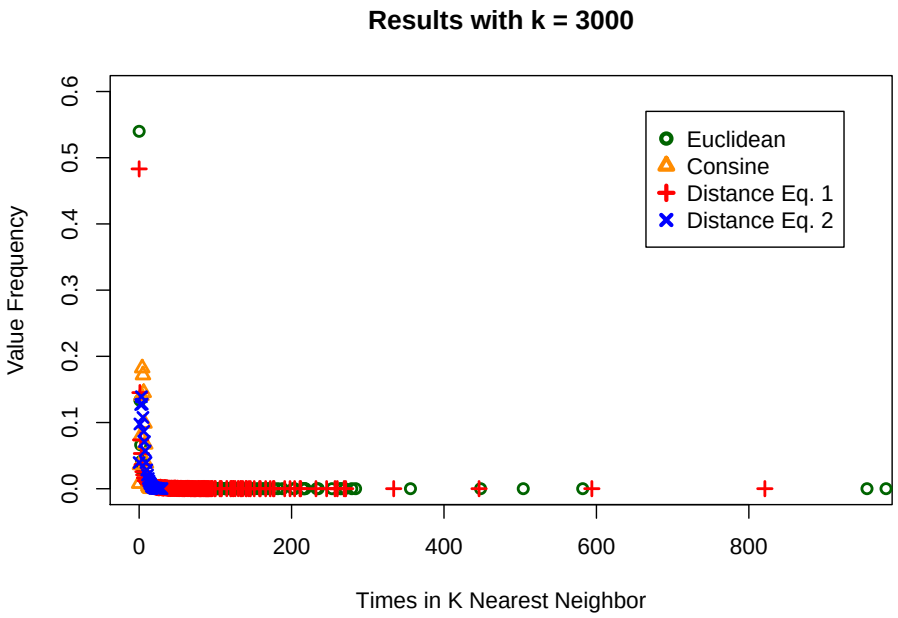
(b) Now we provide the same 3 types of plots for results using a normal random number generator.

Dot Plots: Normal Generator

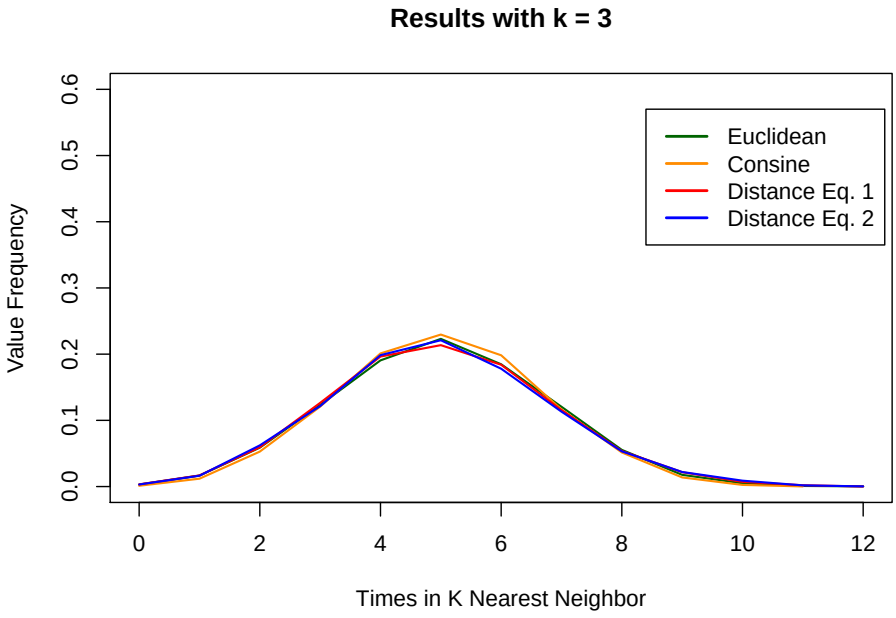


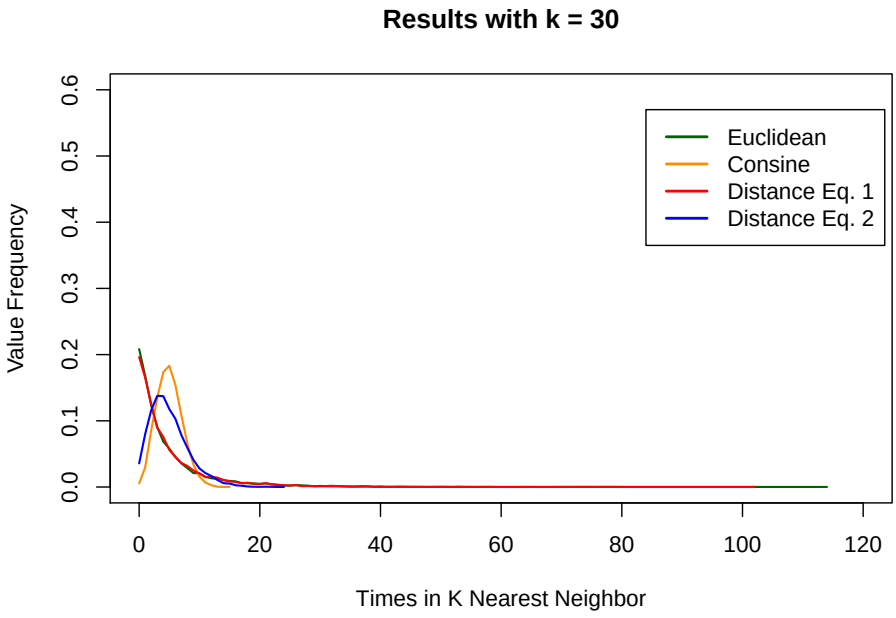


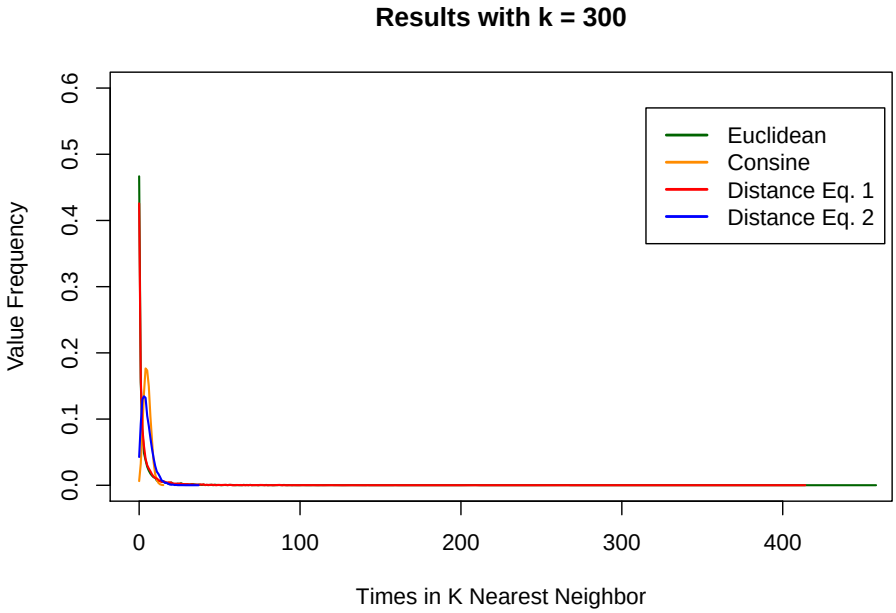

Results with k = 300
Value Frequency
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
100
200
300
400
Times in K Nearest Neighbor
Euclidean
Consine
Distance Eq. 1
Distance Eq. 2

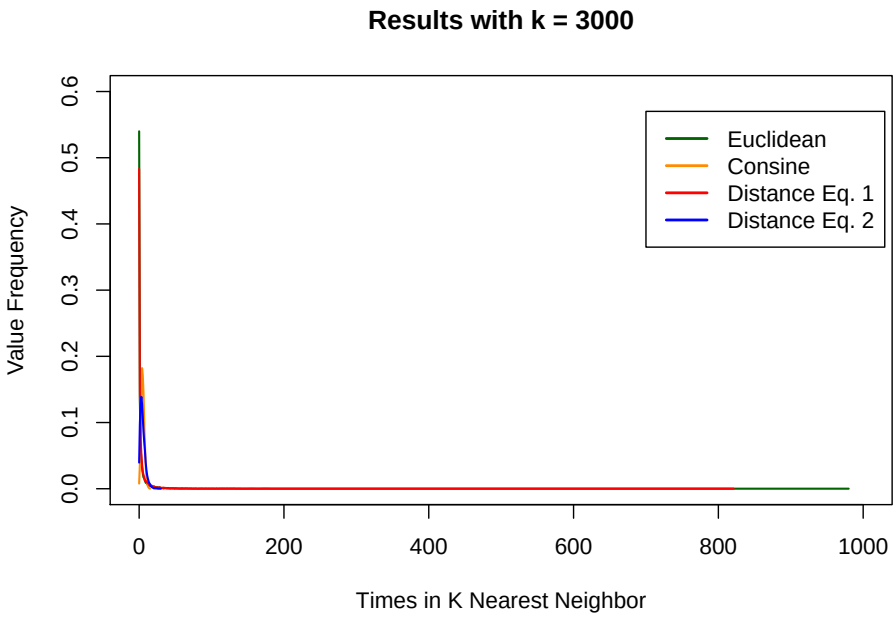

Results with k = 3000
Value Frequency
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
200
400
600
800
Times in K Nearest Neighbor
Euclidean
Consine
Distance Eq. 1
Distance Eq. 2

Line Plots: Normal Generator

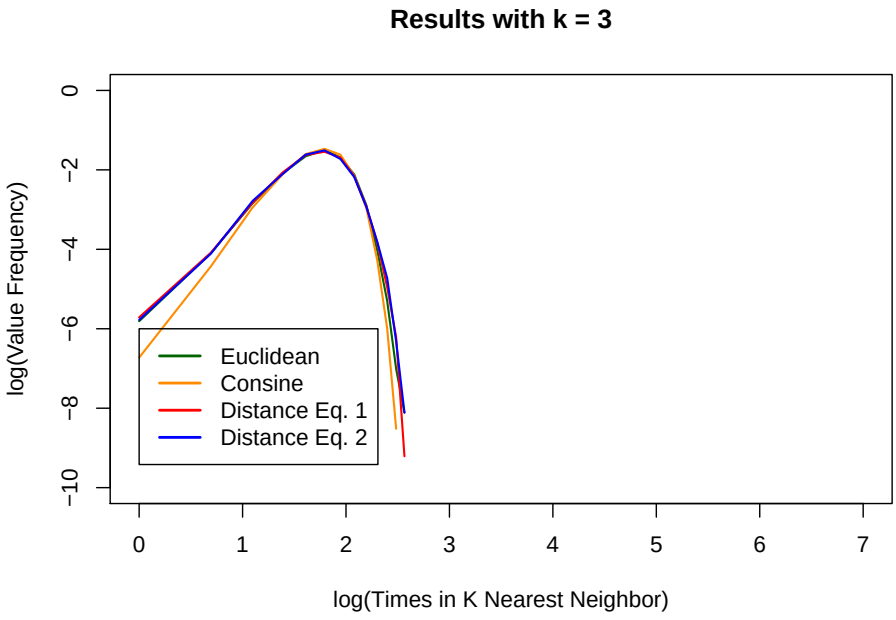


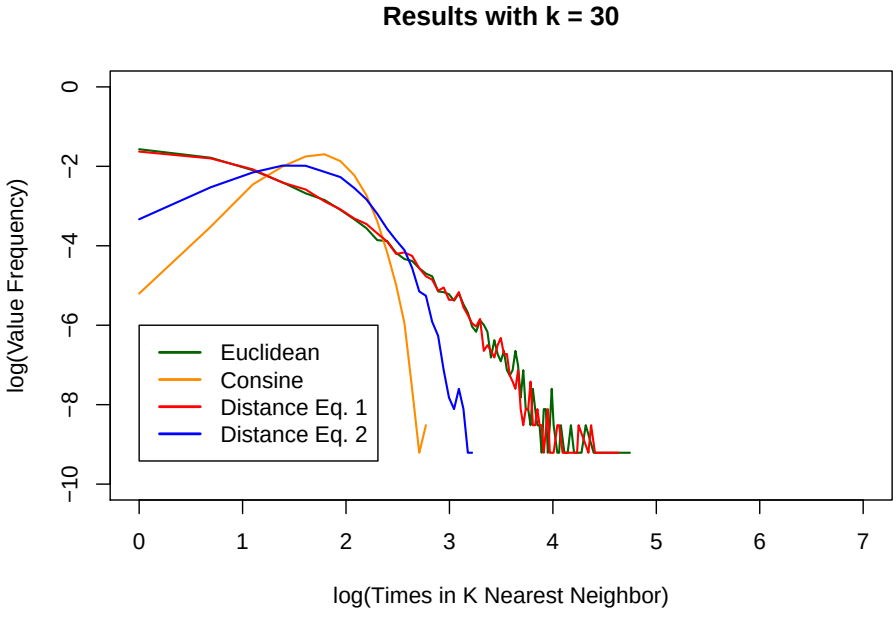


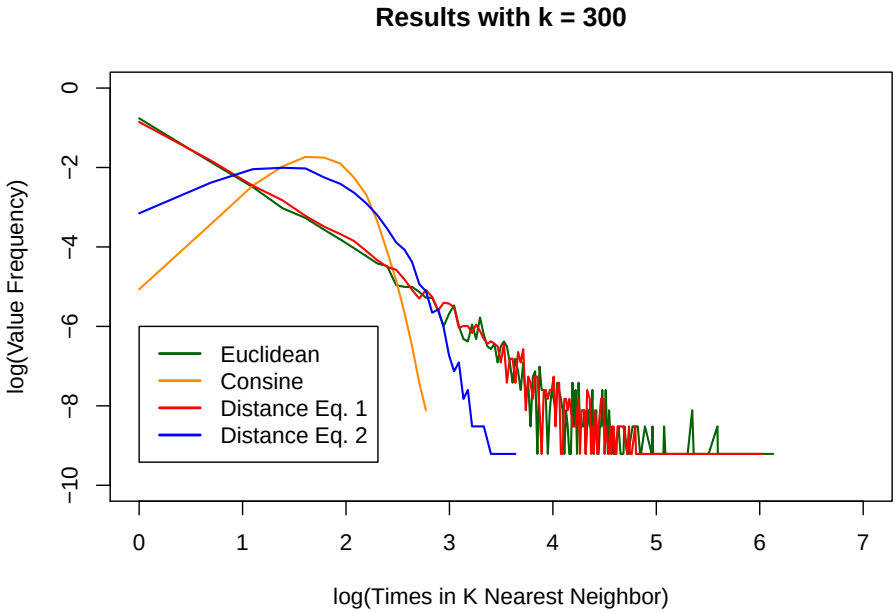

Results with k = 300
Value Frequency
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
100
200
300
400
Times in K Nearest Neighbor
Euclidean
Consine
Distance Eq. 1
Distance Eq. 2

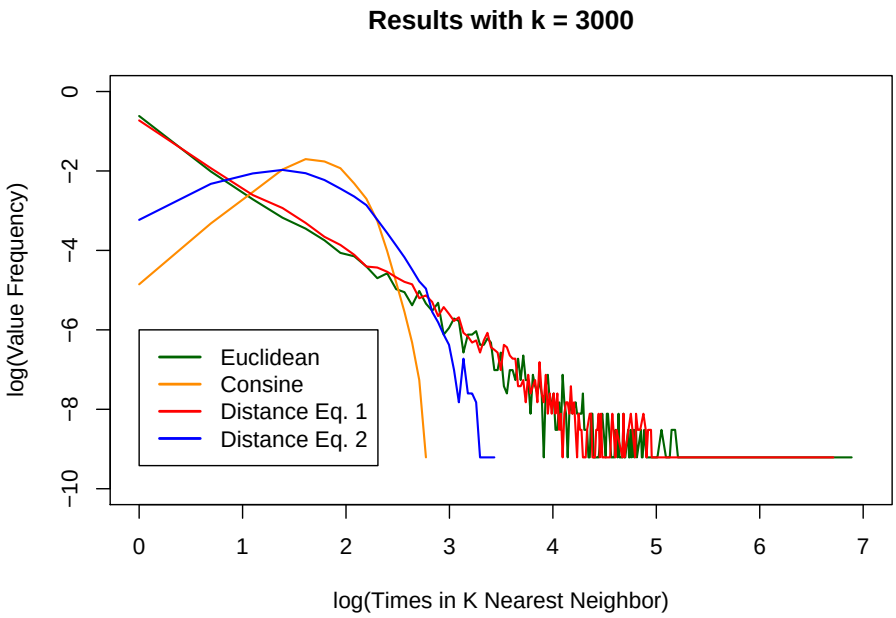

Results with k = 3000
Value Frequency
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
200
400
600
800
1000
Times in K Nearest Neighbor
Euclidean
Consine
Distance Eq. 1
Distance Eq. 2

Log-log Plots: Normal Generator









(c) Observations

Uniformly generated data:

(1) With all 4 distance functions, as k grows, the number of outliers grows. Here an outlier means a data point x with very low $n_5(x)$ values (e.g. 1 or 2). This is natural, as the dimension number grows, the space becomes larger. Therefore in general the 10000 data points will be more sparsely distributed, which results in more outliers.

(2) With all 4 distance functions, as k grows, the number of large hubs grows. Same with the uniform case, in low dimensions, there are no large hubs (say, points that are in the 5-neighborhood of another

100 points). As the number of dimensions grow, the large hubs start to show up. However, the hubs still only constitute a small portion of all data.

Normally generated data:

(1) With Euclidean and Eq. 1 distance functions, as k grows, the number of outliers grows.

(2) With Euclidean and Eq. 1 distance functions, as k grows, the number of large hubs grows. As we can see, in low dimensions, there are no large hubs (say, points that are in the 5-neighborhood of another 100 points). As the number of dimensions grow, the large hubs start to show up. However, the hubs still only constitute a small portion of all data.

(3) With Cosine and Eq.2 distance functions, as k grows, the number of outliers and hubs do not vary much. The distribution of n_5 values stay relatively the same.

(d) Now we propose the following measures to quantify our comparisons on outliers and hubs.

(1) For outliers, we propose 2 measures

- Measure 1 is naturally derived directly from the definition of outliers. It is defined as

$$m_1(N_5) = \text{card}(\{x \in \mathcal{D} : n_5(x) \leq \texttt{threshold}\})$$

  It counts the number of outliers in our results, where outliers are determined by the value `threshold`. Here, $\text{card}(\cdot)$ is a set function which outputs the cardinality of a set. When a set S is finite, $\text{card}(S)$ is the number of elements in S. One problem of this approach is that we need to manually decide the value of the `threshold`.

- Measure 2 is defined as

$$m_2(N_5) = \sum_{i=1}^{n} \exp\left(-n_5(x_i)\right)$$

  Note that it penalizes data points with large n_5 values exponentially. Therefore, a N_5 result with many outliers will have high measure under m_2, while a N_5 result with few outliers will have low measure.

(2) For hubs, we similarly propose 2 measures

- Measure 1 is naturally derived directly from the definition of hubs. It is defined as

$$m_1(N_5) = \text{card}(\{x \in \mathcal{D} : n_5(x) \geq \texttt{threshold}\})$$

  It counts the number of hubs in our results, where hubs are determined by the value `threshold`.

- Measure 2 is defined as

$$m_2(N_5) = \sum_{i=1}^{n} \exp\left(n_5(x_i)\right)$$

  Note that it gives exponentially high weights to data points with large n_5 values. Therefore, a N_5 result with many hubs will have high measure under m_2, while a N_5 result with few hubs will have low measure.

Since m_2 might involve exponential growth when we have a large data set, we can normalize it by the following way

$$m_2^{\text{normalized}}(N_5) = \frac{\sum_{i=1}^{n} \exp\left(n_5(x_i)\right)}{n \exp(\max_i\{n_5(x_i)\})}$$

Problem 3

How to run our program

This code for this problem is written in MATLAB. *k*-means and Elkan's *k*-means are written as two functions with the same 3 inputs

```
kmeans(data_set, num_cluster, dist_fcn)
kmeans_elkan(data_set, num_cluster, dist_fcn)
```

The first input `data_set` is a string, which is the file name of the data set file. The data set file is required to be written in `.csv` format, with no missing values, and with only numerical data.

The second input `num_cluster` is an integer, representing the number of clusters.

The last input `dist_fcn` is an integer for the selection of distance functions. Setting value of `dist_fcn` to 1, 2, 3, and 4 will result in using the Eclidean, Consine, Eq. 1, and Eq. 2 as distance function, respectively.

Both functions will output a vector `[labels, tot_err, itr_no]` denoting the predicted labels, the total sum of squares errors, and the total number of iterations.

We also provided an evaluation script to help with the evaluations. It is also written as a function

```
evaluation(data_set, num_cluster, method_no, dist_fcn)
```

The input `data_set`, `num_cluster`, and `dist_fcn` are same as before. The input `method_no` allows selection between standard and Elkan's *k*-means. A value of 1 would result in the standard *k*-means, while a value of 2 would result in Elkans *k*-means.

The evaluation function prints the total error rates and total within cluster sum of squares errors. Note that the evaluation function assumes that the data set has the last column as true labels.

(b) We repeat the experiment carried out by Elkan in 2003, with the Birch data set 1. The results are shown below.

Note that the speedups are no longer as significant as in the original paper, which goes as high as 351. This is because compared to 2003, today's MATLAB works much faster on matrix or vector computations. However, loops in MATLAB code is still not efficient. The speed difference between MATLAB loops and vector computations is very significant. Our standard *k*-means is highly vectorized. For example, the distance function can take n pairs of vectors at the same time and returns n distance calculations. In this way, we do not have any loops in MATLAB code over dimensions. In contrast, in our implementation of Elkan's algorithm, because of the comparisons on the lower and upper bounds, we were not able to vectorize the code as much. Therefore, although the standard *k*-means calculates the distance functions more times, the highly vectorized code boosted its efficiency, whereas the Elkan's algorithm spends more time on MATLAB loops. This results in our observation that the speedups are not impressive. This also caused the speedup decrease as k grows, since some of the k loops in the standand *k*-means are implemented by vectorized computations but less so in the Elkan's *k*-means. We would expect a different speedup result if we use more efficient, lower-level language like C.

Experiments on the Birch 1 Set

	$k = 3$	$k = 20$	$k = 100$
Standard	27.097662	64.243590	72.775367
Fast	3.804287	10.495013	18.878180
Speedup	7.12	6.12	3.85

Time is measured in seconds

(d) To assess the quality of our results, we used 3 data sets from the UCI machine learning repository: Glass, Iris, and Frogs. We ran the k-means algorithm on those sets. We use the the **total error rates**, defined below, to evaluate our implementation of the k-means algorithm.

In our evaluation method, the predicted label of a cluster is calculated as the most common label in that cluster. All data points in that cluster will be assigned the same cluster label. For example, if x is an element in the cluster C, and the most common label in C is Class 1, then x will be labeled as Class 1. The number of errors is calculated as the number of data points whose predicted label is different from its true label. The total error rate is then calculated as

$$\text{total error rate} = \frac{\text{number of data points whose predicted labels are different from their true labels}}{\text{total number of data points}}$$

Also, for comparison, we also performed experiments using randomly generated data. More specifically, in the "random guess" experiment, we generate predicted labels for all data points using the sample distribution from the true labels. For example, if the true labels are $[1,2,1,2,1]$, then we will generate 5 labels, each being 1 or 2 with probability of $\frac{3}{5}$ and $\frac{2}{5}$, respectively. Comparison with the result from this random guess experiment will tell us if our algorithm actually learns any information.

Below we list in a table the total error rates for all pairs of distance function and data set.

Total Error Rates

	Glass	**Iris**	**Frogs**
Euclidean	33.1776%	10.6667%	14.1765%
Cosine	64.4860%	66.6667%	35.9416%
Eq. 1	33.6449%	15.3333%	16.2057%
Eq. 2	33.6449%	64.6667%	15.9416%
Random Guess	72.8972%	71.3333%	89.7012%

Observation

- From the error rates above, we can see that the algorithm with all 4 distance functions performs better than random guesses. Therefore, our program finds meaningful clusters for all 3 data sets.
- For the Glass set, the Euclidean, Eq. 1, and Eq. 2 distances perform relatively well while the cosine distance produces only slightly better result than random guess.
- For the Iris set, the Euclidean and Eq. 1 distances perform very well, while the other two do not produce satisfactory results.
- For the Frogs set, all 4 distances produce satisfactory results. The Euclidean, Eq.1, and Eq.2 gave really low error rates.
- Here our results are the best results among 200 independent runs. Since k-means algorithm only guarantees local optimum, bad initial conditions will lead to bad results. For example, here is a sample of 20 runs on the Iris set with the Euclidean distance:

  [10.6667, 11.3333, 33.3333, 10.6667, 11.3333, 11.3333, 11.3333, 11.3333, 11.3333, 11.3333, 33.3333, 33.3333, 11.3333, 10.6667, 11.3333, 11.3333, 33.3333, 33.3333, 10.6667, 11.3333]

  Note that there are 3 possible results and the worst, 33.3333 is quite far away from the better result 10.6667 we found. Therefore, multiple restarts greatly improved our results.

Problem 4

(a) An Impossibility Theorem for Clustering

In this article, the author Jon Kleinberg discussed how far we can achieve in developing clustering methods. By using an axiomatic framework, he showed that there does not exist any clustering method that can satisfy the 3 axioms he proposed. Theses 3 axioms, stated in formal logical statements, are abstracted from properties we would like to see in a good clustering method. In other words, no matter how hard we try, we will not be able to design a clustering scheme that has all the 3 desirable properties at the same time. This discussion is interesting as it is independent from any particular clustering method and thus the result can be applied to every clustering method satisfying his very general definition of clustering methods.

In his discussion, a clustering function is defined as a function f who maps a finite set S and a pairwise distance function d on S to a partition of S. Note that this definition is very general. All deteriministic clustering methods, which are functions who take a set and a distance function on the set and output a partition of the same set, fall into this category. Thus we can see that Kleinberg's discussion applies to many of the usual clustering methods such as k-means and k-median.

Then he rigorously defined 3 properties as 3 axioms that we usually desire in clustering methods.

- Scale-invariance is the property that if we rescale a distance function d, using the same clustering f, we would get the same result as before rescaling. In other words, $f(d(x)) = f(c \cdot d(x))$ where $c > 0$. This is desirable because we want the clustering to be independent from the measuring units. Clustering data with inch as unit should give us the same result if we cluster the same data set with values converted to centimeters.
- Richness is the property that by changing the distance function only, we can achieve all possible clustering results, i.e. all possible set partitions. Formally, it is saying that the codomain of f is the collection of all partitions of S. This is reasonable as it assures the power of our clustering method. We do not want to miss the best partition when clustering and we can be sure of this if f is capable of producing all possible results.
- Consistency is the property that if we have a cluster result and we only increase the distance between points from different clusters and decrease distance between same-cluster points, then our clustering method would give us the same result.

Then the impossibility result can be state as: for a multi-element data set S, there is no clustering method f that satisfies all three of the properties above.

In the following section, the author proved some stronger results regarding the scale-invariance and consistency property. More specifically, he proved that if a clustering method f satisfies the two properties then its range has to be an antichain. He also showed that the reverse is true, i.e. for every antichain of partitions of a set, there is a clustering function whose range is that antichain.

As an application in the next section, the author showed that a large family of clustering methods, (k,g)-centroid clustering methods, do not satisfy the consistency property. This result applies to many centroid based clustering methods we commonly use, as the (k,g)-centroid family includes popular cluster methods

like k-means and k-median. This is a very interesting but disappointing result, as it indicates that no matter how good our distance function and tie-breaking method are, k-means and k-median methods can never achieve the consistency property.

The impossibility result is very disappointing, as it points out that we will never be able to design a "perfect" clustering method in the sense of having all 3 nice properties. To explore how much we can actually achieve, the author in the last section of the article, discussed results on relaxing the properties. Along this direction, Kleinberg first gave two examples, one derived from Theorem 3.2 which satisfies scale-invariance, consistency and a relaxed richness and one of the single-linkage clustering with distance-r stopping condition which satisfies consistency, richness and a relaxed scale-invariance.

In remaining part of the section Kleinberg focused on the relaxation of the consistency property. He proposed the refinement-consistency, which requires if we decrease within cluster distances and increase between cluster distances, the clustering function would give a refinement of the previous result. He claimed that there are still no clustering functions that would satisfy all three of scale-invariance, richness and the relaxed consistency properties. However, he points out that with this relaxed consistency, one can find some clustering functions that can satisfy almost all three. In particular Kleinberg claims that there exists a function that achieves scale-invariance and refinement-consistency, with its range include all partitions of the set except the trivial partition, i.e. a partition with each block having only 1 element. Thus this function almost satisfies the richness property, except that it is unable to produce one uninteresting partition. Other possibilities on relaxations are also proposed at the end of the article. Kleinberg claimed without proof, that clustering functions exist that satisfy those relaxed conditions.

Comments

The observations and results from the article is very general and deep. The three axioms Kleinberg proposed look like natural requirements for a good clustering method. Kleinberg proved from a logical point of view that it is pointless to search for clustering results that satisfy all those requirements.

However, although the three axioms are plausible requirements, some of them, or all of the three are overly strict, even with the relaxations.

First, the consistency property is not totally natural. If we decrease within cluster distances and increase between cluster distances, many times, it would be reasonable to form a totally different clustering partition. For example, if there is 5 clusters to begin with and then after the distance transformation, 3 of them become so faraway from the rest that it makes sense if we end up with only 2 clusters.

Secondly, the richness property is also too strict. For a good clustering scheme, we often do not really need the clustering function to be able to reach all possible partitions. There are always some uninteresting partitions that we do not want to end up as the clustering result, like the example of the trivial partition Kleinberg gave in the last section. Moreover, sometimes, we only want to find the best clustering results that satisfy certain conditions. For example, if we know the actual number of clusters, we only need the clustering function to be able to output all partitions with a certain number of blocks. The clustering function can be still be very good clustering methods, even though it does not satisfy the richness property.

(b) Measures of Clustering Quality: A Working Set of Axioms for Clustering

In this article, the authors first discussed Kleinberg's impossibility results. By analyzing Kleinberg's framework, they criticized the impossibility theorem's approach and proposed a new, more flexiable and richer approach: measures of clustering quality.

By discussing Kleinberg's results, Ackerman and Ben-David criticized the axioms Kleinberg used, especially the consistency property, which requires a clustering function to be able to output the same result if we decrease within-cluster distances and increase between-cluster distances. They claimed, by a simple example, that the consistency property is not always reasonable, as sometimes when the relative distances change, it is might be "wise" for the clustering method to output a different, better result. Deriving the same clustering result, in those cases, would be undesirable. Thus, the consistency axiom does not always guarantee a good clustering method, and a reasonable clustering method does not necessarily need to satisfy the consistent property.

To address problems in Kleinberg's framework, Ackerman and Ben-David proposed a new framework, which uses a clustering-quality measure. They defined the measure as a function that takes in a data set and a clustering method and outputs a score, or more generally, an element in an ordered set. Instead of determining whether a clustering method is good or not by checking how many axioms it satisfies, as did in Kleinberg's work, the clustering-quality measure provides a more flexible solution.

In the following section, Ackerman and Ben-David worked on axiomitizing properties that the clustering-quality measures should have. They transformed the 3 desired properties from Kleinberg's work into 3 axioms on the clustering-quality measures.

- The scale invariance property requires a quality measure m to output the same score if we rescale the distance funciton, i.e. requiring that $m(C,X,d) = m(C,X,\lambda d)$ for any clustering method C, data set X, and distance function d.
- The consistency property requires that if we decrease within-cluster distances and increase between-cluster distances, the measure $m(C,X,d)$ would not decrease.
- The richness property requires that for every nontrivial clustering method, there exists a distance with which the clustering-quality measure of the clustering is the maximum score.

To show that these are reasonable requirements, or form a consistent set of requirements, the authors then provided an example of the clustering-quality measure: the relative margin measure, which satisfies all 3 of the properties above.

In the next section, Ackerman and Ben-David argued that the set of axioms they proposed achieved soundness. However, it fails to have completeness, as some functions that obviously should not be considered to be a quality measure would somehow satisfy all 3 axioms they proposed. They showed that this is possible by giving the simple example of a identity-based clustering-quality measure that gives one score if two specific points belong to the same cluster and a different score otherwise. This leads to the proposal a fourth axiom, which was used to address this problem.

The fourth axiom is about permutation invariance. It states that if two clusterings are isomorphic, then their quality measures should be the same. Here two clusterings C and C' are isomorphic means that the

two clusterings have the same domains, and there is a distance-preserving bijection $\phi : X \to X$ such that if x and y are within one cluster under C, then $\phi(x)$ and $\phi(y)$ should be within one cluster under C'. In simpler words, the permutation invariance property requires that if we only change the name of the data points, the clustering-quality measure function shoud give out the same score. This axiom eliminates the problem of the identity-based clustering-quality measures mentioned above. The author also claimed that the set of all 4 axioms are consistent, as the relative margin measure satisfies all of them.

Ackerman and Ben-David then provided two examples of clustering-quality measures that satisfy all 4 axioms: weakest link measure and additive margin measure. They claimed that the complexity of computing the measures are all at most polynomial, at least under some restrictions. They also proposed several ways to combine clustering-quality measures and create new clustering-quality measures.

In the last section of the article Ackerman and Ben-David discussed about clustering quality evaluations that are dependent on the number of clusters. The evaluations studied in other parts of the article are all independent from the number of clusters. However, when evaluating clustering methods like the k-means method, it is useful to use evaluation methods that depends on the number of clusters. The authors proposed a new clustering-quality measure $\mathcal{L}$-normalization. They also discussed the phenomenon that many evaluation methods or loss functions bias towards more refined cluster. They defined these properties rigorously as refinement preference or coarsening preference. They proposed that to evaluate clustering with fixed number of clusters, the refinement preferred measures are useful. But if the correct number of clusters are unknown, then those meausures should be avoided.

Problem 5

(a) In the case of 1-dimensional $\mathbb{R}$, we can rewrite the distance function as

$$d(x,y) = (x-y)^2$$

Assume that one cluster has points $x_1, x_2, \cdots, x_n$. Then the centroid is the solution to the following problem

$$\arg\min_{c\in\mathbb{R}} \sum_{i=1}^{n} (x_i - c)^2$$

Note that when the cluster is fixed, the function

$$f(c) = \sum_{i=1}^{n} (x_i - c)^2$$

is a C^∞ function. Taking the first derivative, we have

$$f'(c) = 2\sum_{i=1}^{n} (c - x_i) = 2(nc - \sum_{i=1}^{n} x_i)$$

Letting the derivative equal to 0, we have

$$2(nc - \sum_{i=1}^{n} x_i) = 0$$

Solving for c, we have the one critical point of f

$$c = \frac{\sum_{i=1}^{n} x_i}{n}$$

Now take the second derivative of f, and we have

$$f''(c) = 2n > 0, \quad \forall c \in \mathbb{R}$$

Therefore, f is globally convex, and hence at the critical point we found above, f achieves its minimum. That is, the centroid of the cluster should be

$$c = \frac{\sum_{i=1}^{n} x_i}{n}$$

(b) In the case of 1-dimensional $\mathbb{R}$, we can rewrite the distance function as

$$d(x,c) = \frac{|x-c|}{\max\{|x|, |c|, |x-c|\}}$$

Assume that one cluster has points $x_1, x_2, \cdots, x_n$, in increasing order. Then the centroid is the solution to the following problem

$$\arg\min_{c\in\mathbb{R}} \sum_{i=1}^{n} \frac{(x-c)^2}{(\max\{|x|, |c|, |x-c|\})^2} \tag{E7}$$

For convenience, we denote

$$f(c) = \sum_{i=1}^{n} \frac{(x-c)^2}{(\max\{|x|,|c|,|x-c|\})^2}$$

Lemma 1. The minimum of f does not occur in $(x_n,\infty)\setminus\{0\}$ or $(-\infty,x_1)\setminus\{0\}$.

Proof. For $c > x_n$ and $c > 0$, we can simplify the expression of f to

$$f(c) = \sum_{\substack{i=1\\x_i<0}}^{n} \frac{(x_i-c)^2}{(x_i-c)^2} + \sum_{\substack{i=1\\x_i\geq 0}}^{n} \frac{(x_i-c)^2}{c^2} = (\text{number of negative } x_i) + \sum_{\substack{i=1\\x_i\geq 0}}^{n} \left(1-\frac{x_i}{c}\right)^2$$

which is an increasing function of c. Therefore, the minimum cannot occur in the open interval (x_n,∞). At worst, it would occur at the point x_n or 0, if $x_n < 0$.

For $c > x_n$ and $c < 0$, $x_i < 0$ for all i. We can simplify the expression of f to

$$f(c) = \sum_{i=1}^{n} \left(\frac{x_i-c}{x_i}\right)^2 = \sum_{i=1}^{n} \left(1-\frac{c}{x_i}\right)^2$$

which is a decreasing function. Thus, $c = 0$ would give smaller value of f than any point in $(x_n,0)$. The minimum cannot occur in the open interval $(x_n,0)$.

Using a similar argument, we can show that the minimum of f does not occur in $(-\infty,x_1)\setminus\{0\}$. □

Now assume that the data set involves both positive and negative data points. We still use $x_1,\cdots,x_k$ to denote a list of increasing data points. Assume that x_k is a negative point and x_{k+1} is a positive point.

For $c < 0$,

$$\begin{aligned} f(c) &= \sum_{x_i<c<0} \frac{(x_i-c)^2}{x_i^2} + \sum_{c<x_i<0} \frac{(x_i-c)^2}{c^2} + \sum_{x_i>0} \frac{(x_i-c)^2}{(x_i-c)^2} \\ &= \sum_{x_i<c<0} \frac{(x_i-c)^2}{x_i^2} + \sum_{c<x_i<0} \frac{(x_i-c)^2}{c^2} + (\text{number of positive } x_i) \end{aligned}$$

For $c > 0$,

$$f(c) = \sum_{0<c<x_i} \frac{(x_i-c)^2}{x_i^2} + \sum_{0<x_i<c} \frac{(x_i-c)^2}{c^2} + (\text{number of negative } x_i)$$

Note that the expression of the summands in $f(c)$ only changes at points $\{x_i\}$ and 0. Therefore, within the intervals

$$[x_1,x_2],\ [x_2,x_3],\ \cdots,\ [x_{k-1},x_k],\ [x_k,0],\ [0,x_{k+1}],\ [x_{k+1},x_{k+2}],\ \cdots,\ [x_{n-1},x_n] \tag{E8}$$

f is a C^∞ function. Also notice that the lowest power of c within f is -2 and the highest power is d. Therefore, the lowest power of c in $f'(c)$ is -3 and highest power is 1. Thus, solving $f'(c) = 0$ involves finding roots for at worst a 4th order polynomial, which can be done **analytically**. By the lemma we proved above, we only need to check all the intervals in (E8). Therefore, we can determine all the critical points of f in finite time.

We propose the following algorithm to find the centroids:

- For each interval of (E8),
  - Calculate f as a fixed expression. This involves only calculation of the coefficients.

  - Calculate the derivative of f. This also involves only calculation of the coefficients.
  - Analytically solve for the roots of f' within the interval. Each of the roots is recorded as a critical point.

- Evaluate f at all the critical points, at 0, and at all x_i.
- The lowest f value from the last step is the minimum of f. And we also know from the last step where the minimum occurs. Therefore, we have our centroid value.

The time complexity of this algorithm is $O(n)$.

The cases of all positive or negative data points are simplified cases and can be solved by a slightly modified algorithm.

Note that this algorithm can be generated to n-dimensional. Instead of having intervals, we solve in n-dimensional boxes. The time complexity are $O(nkf)$, where f is the number of fractions of the space.