B565 Fall 2018: Homework #2

Due on Thursday, September 14, 10:00P

Dr. Radivojac

Arnav Arnav (aarnav@iu.edu)

February 26, 2018

Contents

Problem 1: ROC curves

To create an Receiver Operating Characteristic curve, for a given classification task, we need the actual labels of the classes in the dataset and the scores given to each of the data points by the classification algorithm. We then sort the actual class labels based on the classification score for each of the data points.
Suppose we are starting from the positive class, we see that when the number of points in the positive class increases, the true positive rate increases (by $\frac{1}{n_+}$) and when the number of points in the negative classes increases, the false positive rate increases. Thus, we only need to check the false positive rate and the true positive rates for the classification threshold that lie between the scores for points where the class labels change. The rest of the points can be plotted with the knowledge that one one of the true positive rate or the false positive rate increase, when when checking thresholds where class labels do not change.
Now consider that some classification algorithm scores the points in such a manner that on sorting the labels with respect to the scores, all the points in the positive class are together and all those in the negative class are together. That is, after sorting, there is only one place where the class labels change.
Therefore, we only need to check one threshold here (between scores for points where the class changes), and we can draw the ROC curve successfully.
Therefore, We can say that the lower bound of the number of threshold comparisons is 1. It is difficult to reason how the number of points in the dataset are related to the number of changes in class labels (when going in sorted order of classification scores), because these scores are provided by the classification algorithm to a particular data set and can be different for different data sets, and different classification algorithms.
Therefore, in the best case, we need to do only one comparison.

Problem 2: Hubs and outliers in high dimensional space

a. The following plots were obtained by running the code for problem 2, available in the directory code/q2/knn.py

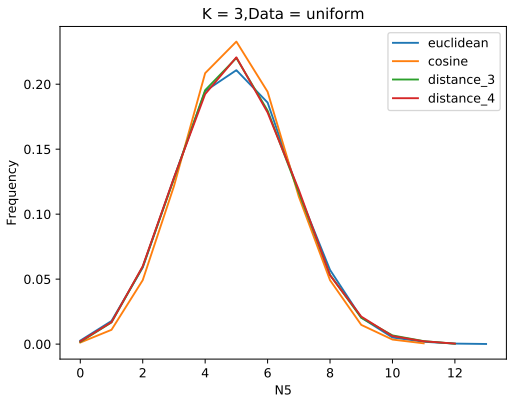

Figure 1: Plot of n5 vs proportion for uniform data points and k = 3

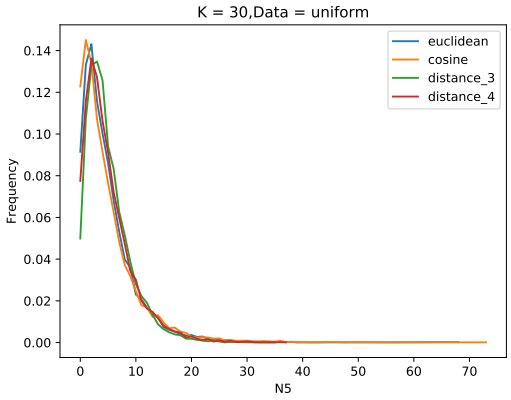

Figure 2: Plot of n5 vs proportion for uniform data points and k = 30

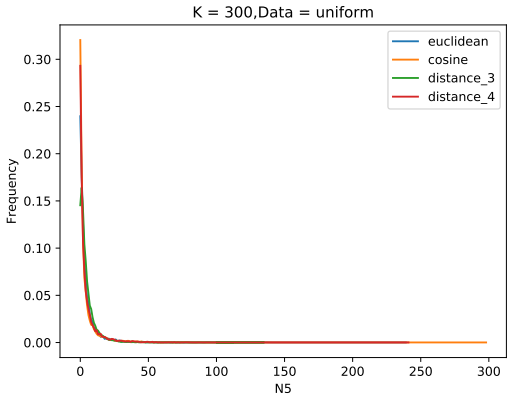

Figure 3: Plot of n5 vs proportion for uniform data points and k = 300

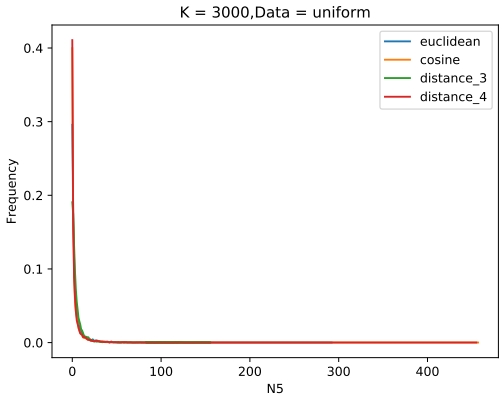

Figure 4: Plot of n5 vs proportion for uniform data points and k = 3000

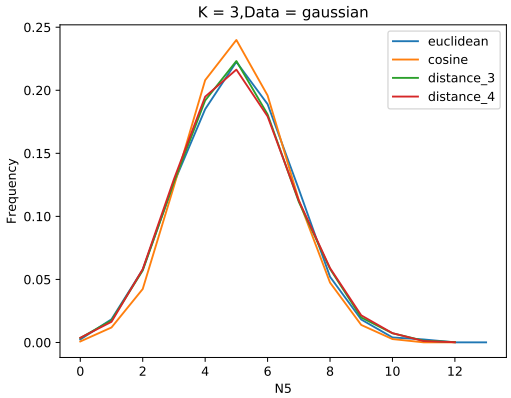

Figure 5: Plot of n5 vs proportion for gaussian data points and k = 3

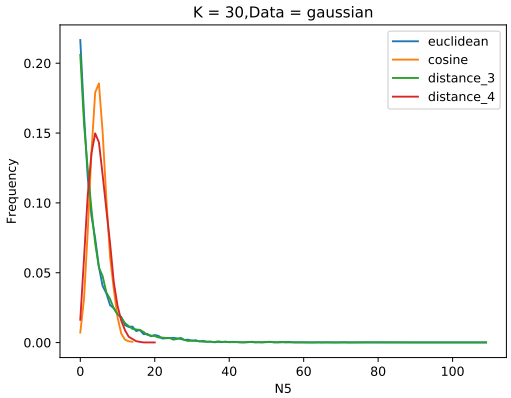

Figure 6: Plot of n5 vs proportion for gaussian data points and k = 30

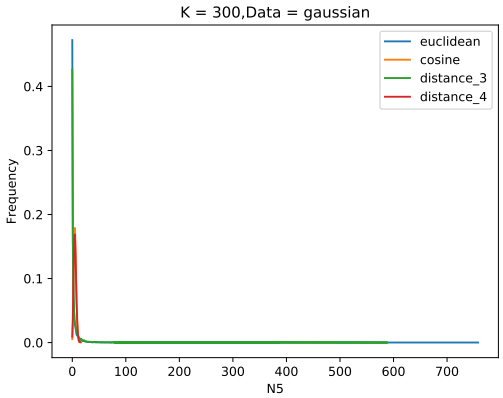

Figure 7: Plot of n5 vs proportion for gaussian data points and k = 300

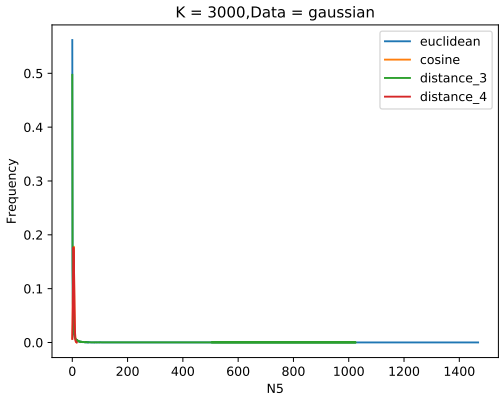

Figure 8: Plot of n5 vs proportion for gaussian data points and k = 3000

b. We see from the graphs that with the increase in dimensionality, the randomly generated points are far apart (curse of dimensionality) and thus most points are not neighbors of fewer points as dimensionality increases.

From figure 5 and figure 1, we can say that points that are in the 5 nearest neighbors of more than 8 points are hubs and those that are in 5 nearest neighbors of less than 2 points are outliers. Most other points lie within a few standard deviations and are more likely to be generated due to nature of the distributions.

Therefore, as the dimensionality increases, the thresholds for hubs need to be lowered, and the thresholds for outliers need to be lowered as well. In other words, as dimensionality increases, for fixed threshold for hubs and outliers, the number of hub points decreases, and the number of outliers increase, since randomly generated points are further apart in a high dimensional space.

Problem 3: K-means Clustering

a. The code for Kmeans algorithm is provided in the code/q3/kmeans.py directory The program is written in python and takes 4 arguments from the user as command line arguments.

- Filename: name of the file containing the data to be clustered
- Method: The method to use for updating centroids (kmeans or elkans)
- distance: The distance function to be used
- K : the number of clusters to be used for clustering.

  to run kmeans algorithm with euclidean distance use method as kmeans, and distance function as euclidean when running the code. The code returns the number of iterations, distance computations, sum squared error and a list of cluster assignments for each point

  The detailed instructions for running the code are provided in the README file in the directory.

b. The Elkans clustering algorithm is included in the code/q3/kmeans.py file

To run the code for Elkan's k-means algorithm, use method as elkans when running the code. The code returns the number of iterations, distance computations, sum squared error and a list of cluster assignments for each point

The following observations were observed when running Elkan's Kmeans clustering on Birch dataset:

Table 1: Performance evaluation of decision tree

K	**Method**	**Distance clacs**	**speedup**
3	$standardKmeans$	15600000	11.02
3	$elkan'sKmeans$	1414959	
20	$standardKmeans$	194000000	59.6
20	$elkan'sKmeans$	3256200	
100	$standardKmeans$	820000000	75.2
100	$elkan'sKmeans$	10904255	

c. The code for K-means algorithm in code/q3/kmeans.py file can be run with the following different distance functions, by using the following values for distance when running the code

- euclidean: Euclidean distance
- cosine: Cosine distance

- cityblock: City block distance
- dist_3 : Distance function specified by Equation 1 in the assignment when p=2
- dist_4 : Distance function specified by Equation 2 in the assignment when p=2

d. We run the K-means algorithm on 3 different data sets from UCI Machine learning repository. These datasets are Wine Dataset [1], Iris Dataset [2] and Glass Identification Dataset [3]

We evaluate the performance of clustering based on the Rand Index [4] for different distance functions and the following results are observed.

Table 2: Performance evaluation of decision tree

Dataset	**Distance**	**Rand Index (Range)**
Iris	*euclidean*	$0.78 - 0.84$
	cosine	$0.87 - 0.93$
	cityblock	$0.82 - 0.87$
	*distance*3	$0.83 - 0.85$
	*distance*4	$0.85 - 0.89$
GlassId.	*euclidean*	$0.68 - 0.72$
	cosine	$0.69 - 0.73$
	cityblock	$0.67 - 0.72$
	*distance*3	$0.68 - 0.73$
	*distance*4	$0.68 - 0.74$
Wine	*euclidean*	$0.73 - 0.76$
	cosine	$0.71 - 0.72$
	cityblock	$0.75 - 0.78$
	*distance*3	$0.77 - 0.78$
	*distance*4	$0.75 - 0.78$

Problem 4: Formalizing Clustering

a. Kleinberg J. The impossibility theorem for clustering. Advances in Neural Information Processing Systems, NIPS 2002.

Although there exists an intuitive definition that comes to mind when we talk about clustering similar data together, there is no formal definition of what a clustering function is and what functions should be considered clustering functions and which ones should not. The paper takes an axiomatic approach and defines three axioms that naturally follow from the intuitive definition of clustering and then goes on to prove that any clustering function can satisfy at most two of the three properties.

The paper defines clustering function as any function f that takes a set S of n points and a distance function d and returns a partitioning of S. The Three axioms that have been defined are as follows:

- Richness: The richness property states that given a set of points S and a clustering function, all the possible partitions of S are possible, each for some distance function d. In other words, any of possible clusterings are achievable with a clustering function f for some distance function d.
- Scale invariance: The scale invariance property accounts for the fact that there should be no inbuilt scale in the clustering function. That is if the distance function is scaled by some positive quantity, the new distance function so obtained should also result in the same clustering as the old distance function.

- Consistency: The Consistency property is defined on a special transform of a distance function that reduces intra-cluster distances and increases inter-cluster distances. The property states that a new distance function obtained by applying such a transform of on the old distance function should return in the same clustering.

The impossibility theorem in clustering states that there exists no such clustering function that can satisfy all of the above properties at the same time, however, most clustering algorithms that work well satisfy some relaxations of these properties.

Most existing clustering functions such as single linkage clustering or centroid based clustering have some stopping conditions that should be looked into, to prove the impossibility theorem. The paper considers the following three:

- k-cluster stopping condition: The clustering algorithm stops when k clusters are obtained.
- r-distance stopping condition: The clustering algorithm stops when there are no clusters with distance less than r between them
- scale-α stopping condition: the clustering algorithm stops when there are no clusters that have a less than $\alpha.\rho$ distance between them, where ρ is the maximum distance between two points in the dataset

The author proves that a clustering function that satisfies the properties of scale invariance and consistency, lacks the property of richness by using the concept of antichains. Given two partitions of a set, one is a refinement of the other if for every set in the first partition there exists a set in the second partition which is a superset of this set. The paper defines a set of partitions of a set to be an antichain if there exist no two subsets that satisfy this property

The author also proves that given an antichain that is a subset of partitions of a set of points, there exists a clustering clustering function that satisfies scale invariance and consistency, which has a range same as the antichainm thus proving that any clustering function that is scale invariant and consistent can only generate a subset of all possible partitions of a set of points, and hence does not satisfy richness.

The author also proves that a centroid based clustering function that uses that uses k-cluster stopping condition and tries to minimize a non decreasing objective function is not consistent.

The author provides examples such as the single linkage clustering function with r-distance stopping condition that satisfies the properties with a natural relaxation of the scale invariance property. The author concludes by saying that to allow clustering functions to satisfy the set of properties the consistency property can be relaxed to refinement consistency, that says that increasing between cluster distances and decreasing within cluster distances results in a clustering that is a refinement of the original clustering, or a coarsening of the original clustering.

b. Ackerman M, Ben-David S. Measures of clustering quality: a working set of axioms for clustering. Advances in Neural Information Processing Systems, NIPS 2008

The paper builds on the ideas of paper on impossibility theorem in clustering by Kleinberg which sets up three natural axioms for clustering and proves that these can not be satisfied at the same time, and thus proving it is difficult to formalize clustering.

The authors disagree with the notion and state that the impossibility arises due to the formalism used in the previous paper and is not an inherent property of clustering. The authors define a set of axioms on clustering quality measures that take into account the axioms set by the previous paper.

The authors define a clustering similarity measure as a function that maps a pair of dataset and the clustering to some value from an ordered set. These clustering quality measures are useful because users need to make a decision whether the clustering describes the pattern in the data well enough or further

data mining is required. They can be used for model selection for clustering to determine , for example, the number of clusters to be used.

The authors describe the axioms of function richness, function consistency and function scale invariance, set by Kleinberg in the previous paper and show how this consistency requirement implies that on increasing between cluster distances and decreasing within cluster distances, the best clustering must not change, which can happen as shown in the paper.

The authors define axioms on clustering quality measures that are analogous to those described earlier, thee are:

- Scale invariance: A clustering quality measure is scale invariant if its value does not change when the distance function used for clustering is scaled by a positive quantity.
- Consistency: A clustering quality measure is consistent if a transform that increases the between cluster distances and reduces the within cluster distances, results in a better value of the quality measure.
- Richness: A quality measure satisfies richness if for any clustering of the data there exists a distance function that results in the best value of the quality measure for the given clustering.
- Isomorphism Invariance: Two clusterings are defined to be isomorphic if for some isomorphism of the dataset, the cluster memberships in the two clusterings before and after isomorphism do not change. A quality measure is isomorphism invariant if its value is same across all isomorphic clusterings of the dataset. This is necessary to prevent functions that use cluster memberships to identify the quality of clustering from being considered as clustering quality measures.

  The authors provide examples of different clustering quality measures that satisfy all the axioms mentioned above, such as relative margin, which gives a low value for a better measure of clustering, and additive margin and weak linkage which give higher value for a better clustering.

  The authors show that these clustering quality measures can be computed in a polynomial time in the number of points in the dataset, which helps in quickly measuring the quality of a clustering. Further, these clustering quality measures can be generated from these quality measures that evaluate a subsets of clusters.

  The paper shows how, quality measures that arise from clustering techniques that try to minimize a loss function, depend on the number of clusters, and do not satisfy scale invariance and consistency axioms. Using such clustering quality measures the number of clusters should be fixed, and they are not helpful in measuring quality of clusterings with different number of clusters.

  In conclusion, the paper successfully explains how the impossibility theorem in clustering is a result of the formalism used to express the axioms and the authors define a new set of axioms on cluster quality measures that help in formalizing clustering to some extent.

Problem 5: Deriving centroids

a. Distance function from equation (1) in the assignment, when p=2 becomes,

$$d^2 = \left(\left(\sum_{i=1}^{k} (x_i - y_i)^+ \right)^2 \left(\sum_{i=1}^{k} (y_i - x_i)^- \right)^2 \right)^{1/2}$$

Case1: x and y are one dimensional

then

$$d^2 = \left(\left(\sum_{i=1}^{k} (x_i - y_i)^+ \right)^2 \right.$$

$SSE = \sum_{i=1}^{k} \sum_{x \epsilon c_i} (d^2(x,y))^2$ $\frac{\partial SSE}{\partial c_k} = \sum_{x \epsilon c_k} 2.(c_k - x_k) = 0$ (for minimizing the distance)

on simplifying, we get

$c_k = \frac{1}{m} \sum_{x \epsilon c_k} x_k$

Thus the best centroid that minimizes the SSE of the cluster is the mean of the points in the cluster.

References

[1] M. Lichman. UCI machine learning repository, 2013.

[2] M. Lichman. UCI machine learning repository, 2013.

[3] M. Lichman. UCI machine learning repository, 2013.

[4] Cambridge university press. evaluation of clustering. stanford nlp website, April 2009.