|  |
| --- |
| INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY |
| DATA MINING |
| ASSIGNMENT 5 |
|  |
| **Vaisakh** |
| **11/20/2014** |

|  |
| --- |
| Codes Are Attached as m files |

1)K-MEANS

WITH EUCLEDIAN DISTANCE (code attached in files kmeans.m findClosestCentroids.m computeCentroids.m findDB.m findSC.m

kMeansInitCentroids.m distortion.m)

|  |  |
| --- | --- |
| **K VALUE** | **DB INDEX** |
| 2 | 12.1945 |
| 3 | 2.3951 |
| 4 | 9.6809 |

BEST K VALUE : **3**

**Cluster plot**



****



WITH MANHATTAN((code attached in files kmeans1.m findClosestCentroids1.m computeCentroids.m findDB1.m findSC1.m

kMeansInitCentroids.m distortion1.m)

|  |  |
| --- | --- |
| **K VALUE** | **DB INDEX** |
| 2 | 3.6883 |
| 3 | 2.7310 |
| 4 | 2.1071 |

BEST K VALUE : **4**

****

**Cluster plot**

**2.1 Agglomerative Clustering**

The merging sequence is

• (*A*3),(*A*5)

• (*A*4),(*A*8)

• (*A*3,*A*5),(*A*6)

• (*A*1),(*A*4,*A*8)

• (*A*2),(*A*7)

• (*A*3,*A*5,*A*6),(*A*1, *A*4, *A*8)

• (*A*3, *A*5, *A*6, *A*1, *A*4, *A*8),(*A*2, *A*7)

****

**Divisive Clustering**

Splitting sequence is

• L1 (*A*1, *A*4, *A*8),(*A*2, *A*3, *A*5, *A*6, *A*7)

• L2 (*A*1),(*A*4, *A*8),(*A*2, *A*7),(*A*3, *A*5, *A*6)

• L3 (*A*4),(*A*8),(*A*2),(*A*7),(*A*3, *A*5),(*A*6)

• L4 (*A*3),(*A*5)

The splitting at a level is done by performing k-means clustering at that level with k=2.

****

**SELF ORGANIZING MAP(K-SOM)**

Using KSOM algorithm trained the neurons and 3 clusters were visualized

Codes Attached : ksom.m , findbmu.m, g.m



HIT HISTOGRAM(with 5X5 grids)

J

HIT HISTOGRAM(with 10x10 grid)

4) **VISUALIZING TECHNIQUES IN SOM**

The U-matrix stands for unified distance and contains in each cell the euclidean distance (in the input space) between neighboring cells. Small values in this matrix mean that SOM nodes are close together in the input space, whereas larger values mean that SOM nodes are far apart, even if they are close in the output space. As such, the U-matrix can be seen as summary of the probability density function of the input matrix in a 2D space. Usually, those distance values are discretized, color-coded based on intensity and displayed as a kind of heatmap. U-matrix (unified distancematrix) representation of the Self-OrganizingMap visualizes the distances between the

neurons. The distance between the adjacent neuons is calculated and presented with different colorings between the adjacent nodes. A dark coloring between the neurons corresponds to a large distance . A light coloring between the neurons signifies that the vectors are close to each other in the input space. Light areas can be thought as

clusters and dark areas as cluster separators. This can be a helpful presentation when one tries to find clusters in the input data without having any a priori information about the clusters. The U-Matrix value of a particular node is the average distance between the node’s weight vector and that of its closest neighbors. In a square grid, for instance, we might consider the closest 4 or 8 nodes (the Von Neumann and Moore neighborhoods, respectively), or six nodes in a hexagonal grid .



In the Figure above we can see the neurons of the network marked as black dots. The representation reveals that these are a separate cluster in the upper right corner of this representation. The clusters are separated by a dark gap.

There are many other methods proposed by researchers which uses visualization techniques that take the distribution of the data set in input space and its density into account. Most commonly, this is visualized as hit histograms, which display the number of data points projected to each map node. A more advanced method is the P-Matrix

that visualizes howdensely populated each unit is by counting the number of data points within the sphere of a certain radius around the model vector in question. Another recently proposed technique that aims at depicting both density and cluster structures is the Smoothed Data Histogram, which relies on a parameter that determines how

blurred the visualization will be. There are also techniques that depict the contribution of the individual variable.

**Cluster Evaluation Techniques**

1. Assessing Clustering Tendency

Clustering tendency assessment determines whether a given data set has a non-random structure, which may lead to meaningful clusters. Consider a data set that does not have any non-random structure, such as a set of uniformly distributed points in a data space. Even though a clustering algorithm may return clusters for the data, those clusters are random and are not meaningful.

The **Hopkins Statistic** is a spatial statistic that tests the spatial randomness of a variable as distributed in a space. Given a data set, D, which is regarded as a sample of a random variable, o, we want to determine how far away o is from being uniformly distributed in the data space. We calculate the Hopkins Statistic as follows:

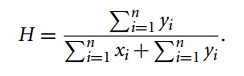
* Sample n points, p1, . . . , pn, uniformly from D. That is, each point in D has the same probability of being included in this sample. For each point, pi, we find the nearest neighbor of pi (1 ≤ i ≤ n) in D, and let xi be the distance between pi and its nearest neighbor in D. That is,



* Sample n points, q1, . . . , qn, uniformly from D. For each qi (1 ≤ i ≤ n), we find the nearest neighbor of qi in D − {qi }, and let yi be the distance between qi and its nearest neighbor in D −{qi }. That is,



* Calculate the Hopkins Statistic, H , as



If D were uniformly distributed, then and xi would be close to each other, and thus H would be about 0.5. However, if D were highly skewed, then would be substantially smaller than in expectation, and thus H would be close to 0.

Our null hypothesis is the homogeneous hypothesis—that D is uniformly distributed and thus contains no meaningful clusters. The nonhomogeneous hypothesis (i.e., that D is not uniformly distributed and thus contains clusters) is the alternative hypothesis. We can conduct the Hopkins Statistic test iteratively, using 0.5 as the threshold to reject the alternative hypothesis. That is, if H > 0.5, then it is unlikely that D has statistically significant clusters.

1. Determining the Number of Clusters

Determining the “right” number of clusters in a data set is important, not only because some clustering algorithms like k-means require such a parameter, but also because the appropriate number of clusters controls the proper granularity of cluster analysis. It can be regarded as finding a good balance between compressibility and accuracy in cluster analysis. If we treat the entire data set as a cluster, this would maximize the compression of the data, but such a cluster analysis has no value. On the other hand, treating each object in a data set as a cluster gives the finest clustering resolution (i.e., most accurate due to the zero distance between an object and the corresponding cluster center). In some methods like k-means, this even achieves the best cost. However, having one object per cluster does not enable any data summarization.

Determining the number of clusters is far from easy, often because the “right” number is ambiguous. Figuring out what the right number of clusters should be often depends on the distribution’s shape and scale in the data set, as well as the clustering resolution required by the user. There are many possible ways to estimate the number of clusters. Here, we briefly introduce a few simple yet popular and effective methods.

A simple method is to set the number of clusters to about for a data set of n points. In expectation, each cluster has points.

**Elbow Method**

The elbow method is based on the observation that increasing the number of clusters can help to reduce the sum of within-cluster variance of each cluster. This is because having more clusters allows one to capture finer groups of data objects that are more similar to each other. However, the marginal effect of reducing the sum of within-cluster variances may drop if too many clusters are formed, because splitting a cohesive cluster into two gives only a small reduction. Consequently, a heuristic for selecting the right number of clusters is to use the turning point in the curve of the sum of within-cluster variances with respect to the number of clusters.

Technically, given a number, k > 0, we can form k clusters on the data set in question using a clustering algorithm like k-means, and calculate the sum of within-cluster variances, var (k). We can then plot the curve of var with respect to k. The first (or most significant) turning point of the curve suggests the “right” number.

**Cross Validation Method**

The “right” number of clusters in a data set can also be determined by cross validation, a technique often used in classification. First, divide the given data set, D, into m parts. Next, use m − 1 parts to build a clustering model, and use the remaining part to test the quality of the clustering. For example, for each point in the test set, we can find the closest centroid. Consequently, we can use the sum of the squared distances between all points in the test set and the closest centroids to measure how well the clustering model fits the test set. For any integer k > 0, we repeat this process m times to derive clusterings of k clusters by using each part in turn as the test set. The average of the quality measure is taken as the overall quality measure. We can then compare the overall quality measure with respect to different values of k, and find the number of clusters that best fits the data.

1. Measuring Clustering Quality

We have a few methods to choose from for measuring the quality of a clustering. In general, these methods can be categorized into two groups according to whether ground truth is available. Here, ground truth is the ideal clustering that is often built using human experts.

If ground truth is available, it can be used by extrinsic methods, which compare the clustering against the group truth and measure. If the ground truth is unavailable, we can use intrinsic methods, which evaluate the goodness of a clustering by considering how well the clusters are separated. Ground truth can be considered as supervision in the form of “cluster labels.” Hence, extrinsic methods are also known as supervised methods, while intrinsic methods are unsupervised methods.

**Extrinsic Methods**

When the ground truth is available, we can compare it with a clustering to assess the clustering. Thus, the core task in extrinsic methods is to assign a score, Q(C , C g ), to a clustering, C , given the ground truth, Cg. Whether an extrinsic method is effective largely depends on the measure, Q, it uses.

In general, a measure Q on clustering quality is effective if it satisfies the following four essential criteria:

*Cluster homogeneity.*

This requires that the more pure the clusters in a clustering are, the better the clustering. Suppose that ground truth says that the objects in a data set, D, can belong to categories L1, ... , Ln. Consider clustering, C1, wherein a cluster C ∈ C1 contains objects from two categories Li, Lj (1 ≤ i < j ≤ n). Also consider clustering C2, which is identical to C1 except that C2 is split into two clusters containing the objects in Li and Lj, respectively. A clustering quality measure, Q, respecting cluster homogeneity should give a higher score to C2than C1, that is,Q(C2, Cg ) > Q(C1, Cg ).

*Cluster completeness*

This is the counterpart of cluster homogeneity. Cluster completeness requires that for a clustering, if any two objects belong to the same category according to ground truth, then they should be assigned to the same cluster. Cluster completeness requires that a clustering should assign objects belonging to the same category (according to ground truth) to the same cluster. Consider clustering C1, which contains clusters C1 and C2, of which the members belong to the same category according to ground truth. Let clustering C2 be identical to C1 except that C1 and C2 are merged into one cluster in C2. Then, a clustering quality measure, Q, respecting cluster completeness should give a higher score to C2, that is, Q(C2, Cg ) > Q(C1, Cg ).

*Rag bag*

In many practical scenarios, there is often a “rag bag” category containing objects that cannot be merged with other objects. Such a category is often called “miscellaneous,” “other,” and so on. The rag bag criterion states that putting a heterogeneous object into a pure cluster should be penalized more than putting it into a rag bag. Consider a clustering C1 and a cluster C ∈ C1 such that all objects in C except for one, denoted by o, belong to the same category according to ground truth. Consider a clustering C2 identical to C1 except that o is assigned to a cluster C’ not = C in C2 such that C0 contains objects from various categories according to ground truth, and thus is noisy. In other words, C0 in C2is a rag bag. Then, a clustering quality measure Q respecting the rag bag criterion should give a higher score to C2, that is,Q(C2, Cg ) > Q(C1, Cg ).

*Small cluster preservation*.

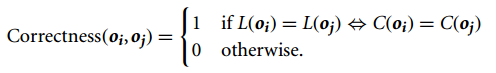
If a small category is split into small pieces in a clustering, those small pieces may likely become noise and thus the small category cannot be discovered from the clustering. The small cluster preservation criterion states that splitting a small category into pieces is more harmful than splitting a large category into pieces. Consider an extreme case. Let D be a data set of n + 2 objects such that, according to ground truth, n objects, denoted by o1, . . . , on, belong to one category and the other two objects, denoted by on+1,on+2, belong to another category. Suppose clustering C1 has three clusters, C1 = {o1, . . . , on}, C2 = {on+1}, andC3 = {on+2}. Let clustering C2have three clusters, too, namely C1 = {o1, . . . , on−1},C2 = {on}, and C3 = {on+1, on+2}. In other words, C1splits the small category andC2splits the big category. A clustering quality measure Q preserving small clustersshould give a higher score to C2, that is, Q(C2, Cg ) > Q(C1, Cg ).

Many clustering quality measures satisfy some of these four criteria. Here, we introduce the BCubed precision and recall metrics, which satisfy all four criteria.

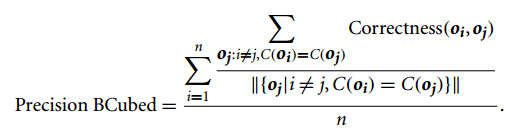
**BCubed precision**

BCubed evaluates the precision and recall for every object in a clustering on a given data set according to ground truth. The precision of an object indicates how many other objects in the same cluster belong to the same category as the object. The recall of an object reflects how many objects of the same category are assigned to the same cluster.

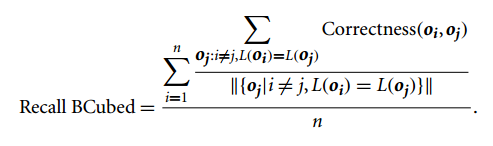
Formally, let D ={o1, . . . , on} be a set of objects, and C be a clustering on D. Let L(oi )(1 ≤ i ≤ n) be the category of oigiven by ground truth, and C(oi ) be the cluster ID of oi in C . Then, for two objects, oi and oj\, (1 ≤ i, j , ≤ n, i 6= j ), the correctness of the relation between oi and oj in clustering C is given by



BCubed precision is defined as



BCubed recall is defined as

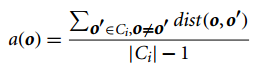


**Intrinsic Methods**

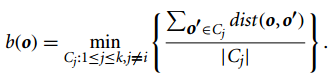
When the ground truth of a data set is not available, we have to use an intrinsic method to assess the clustering quality. In general, intrinsic methods evaluate a clustering by examining how well the clusters are separated and how compact the clusters are. Many intrinsic methods have the advantage of a similarity metric between objects in the data set.

*The silhouette coefficient*

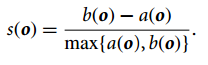
The silhouette coefficient is such a measure. For a data set, D, of n objects, suppose D is partitioned into k clusters, C1, ... , Ck. For each object o ∈ D, we calculate a(o) as the average distance between o and all other objects in the cluster to which o belongs. Similarly, b(o) is the minimum average distance from o to all clusters to which o does not belong. Formally, suppose o ∈ Ci (1 ≤ i ≤ k); then



and



The **silhouette coefficient** of o is then defined as



The value of the silhouette coefficient is between −1 and 1. The value of a(o) reflects the compactness of the cluster to which o belongs. The smaller the value, the more compact the cluster. The value of b(o) captures the degree to which o is separated from other clusters. The larger b(o) is, the more separated o is from other clusters. Therefore, when the silhouette coefficient value of o approaches 1, the cluster containing o is compact and o is far away from other clusters, which is the preferable case. However, when the silhouette coefficient value is negative (i.e., b(o) < a(o)), this means that, in expectation, o is closer to the objects in another cluster than to the objects in the same cluster as o. In many cases, this is a bad situation and should be avoided.

To measure a cluster’s fitness within a clustering, we can compute the average silhouette coefficient value of all objects in the cluster. To measure the quality of a clustering, we can use the average silhouette coefficient value of all objects in the data set. The silhouette coefficient and other intrinsic measures can also be used in the elbow method to heuristically derive the number of clusters in a data set by replacing the sum of within-cluster variances.

**Mahalanobis distance**

The **Mahalanobis distance** is a measure of the distance between a point P and a  distribution D, introduced by P. C. Mahalanobis in 1936. It is a multi-dimensional generalization of the idea of measuring how many standard deviations away P is from the mean of D. This distance is zero if P is at the mean of D, and grows as P moves away from the mean: Along each principal component axis, it measures the number of standard deviations from P to the mean of D. If each of these axes is rescaled to have unit variance, then Mahalanobis distance corresponds to standard Euclidean distance in the transformed space. Mahalanobis distance is thus [unitless](http://en.wikipedia.org/wiki/Unitless" \o "Unitless) and scale-invariant, and takes into account the correlations of the data set.

Definition and properties

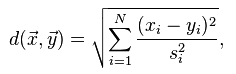
The Mahalanobis distance of an observation Description: x = ( x_1, x_2, x_3, \dots, x_N )^T from a group of observations with mean Description: \mu = ( \mu_1, \mu_2, \mu_3, \dots , \mu_N )^T and covariance matrix *S* is defined as:



Mahalanobis distance (or "generalized squared interpoint distance" for its squared value[[3]](http://en.wikipedia.org/wiki/Mahalanobis_distance" \l "cite_note-3)) can also be defined as a dissimilarity measure between two random vectors Description: \vec{x} and Description: \vec{y}of the same distribution with the covariance matrix *S*:



If the covariance matrix is the identity matrix, the Mahalanobis distance reduces to the Euclidean distance. If the covariance matrix is diagonal, then the resulting distance measure is called a *normalized Euclidean distance*:



where *si* is the standard deviation of the *xi* and *yi* over the sample set.

Mahalanobis distance is preserved under full-rank linear transformations of the space spanned by the data. This means that if the data has a nontrivial nullspace, Mahalanobis distance can be computed after projecting the data (non-degenerately) down onto any space of the appropriate dimension for the data.

Intuitive explanation

Consider the problem of estimating the probability that a test point in *N*-dimensional Euclidean space belongs to a set, where we are given sample points that definitely belong to that set. Our first step would be to find the average or center of mass of the sample points. Intuitively, the closer the point in question is to this center of mass, the more likely it is to belong to the set.

However, we also need to know if the set is spread out over a large range or a small range, so that we can decide whether a given distance from the center is noteworthy or not. The simplistic approach is to estimate the standard deviation of the distances of the sample points from the center of mass. If the distance between the test point and the center of mass is less than one standard deviation, then we might conclude that it is highly probable that the test point belongs to the set. The further away it is, the more likely that the test point should not be classified as belonging to the set.

This intuitive approach can be made quantitative by defining the normalized distance between the test point and the set to be Description:  {x - \mu} \over \sigma . By plugging this into the normal distribution we can derive the probability of the test point belonging to the set.

The drawback of the above approach was that we assumed that the sample points are distributed about the center of mass in a spherical manner. Were the distribution to be decidedly non-spherical, for instance ellipsoidal, then we would expect the probability of the test point belonging to the set to depend not only on the distance from the center of mass, but also on the direction. In those directions where the ellipsoid has a short axis the test point must be closer, while in those where the axis is long the test point can be further away from the center.

Putting this on a mathematical basis, the ellipsoid that best represents the set's probability distribution can be estimated by building the covariance matrix of the samples. The Mahalanobis distance is simply the distance of the test point from the center of mass divided by the width of the ellipsoid in the direction of the test point.

Relationship to normal random variables

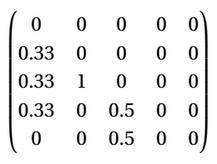
In general, given a normal (Gaussian) random variable Description: X with variance Description: S=1 and mean Description: \mu = 0, any other normal random variable Description: R (with mean Description: \mu_1 and variance Description: S_1) can be defined in terms of Description: X by the equation Description: R = \mu_1 + \sqrt{S_1}X. Conversely, to recover a normalized random variable from any normal random variable, one can typically solve for Description: X = (R - \mu_1)/\sqrt{S_1} . If we square both sides, and take the square-root, we will get an equation for a metric that looks a lot like the Mahalanobis distance:

Description: D = \sqrt{X^2} = \sqrt{(R - \mu_1)^2/S_1} = \sqrt{(R - \mu_1) S_1^{-1} (R - \mu_1) }.

The resulting magnitude is always non-negative and varies with the distance of the data from the mean, attributes that are convenient when trying to define a model for the data.

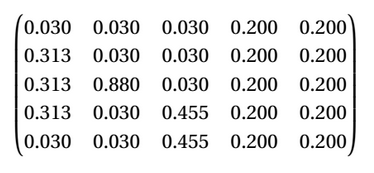
GOOGLE PAGE RANKING ALGORITHM

In these nodes 4 and 5 are dangling nodes. The link structure is



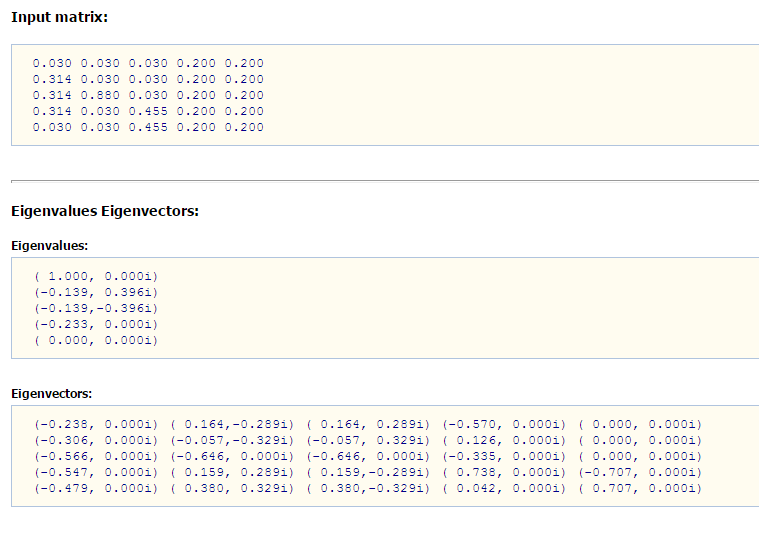
M=(1-p)\*A+p\*S where S is a 5\*5 matrix with 1/n as the entries where n is the number of pages in the web structure. P = 0.15

M matrix is given below



PTO

Calculated Eigen Values And Vectors By Online ToolBox are Reported Below



Taking Eigen Vector Corresponding To Eigen Value 1

The rank Obtained is **-0.238**

**-0.306**

**-0.566**

**-0.547**

**-0.479**