SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE MATERIAL

Subject Name : DATA ANALYTICS **UNIT III** **Subject Code : SIT1303**

3.1Linear Discriminant Analysis

3.2 Discriminant Analysis

3.3 Nearest Neighbour Classifier

3.4 Fuzzy Clustering

3.5 Self Organizing Map

3.6 Learning Vector Quantization

3.7 Relational Clustering

3.8 Partitioning Methods

i)K-Means

ii)K-Medoids

3.1Linear Discriminant Analysis:

Linear Discriminant Analysis (LDA) is most commonly used as dimensionality reduction technique in the pre-processing step for pattern-classification and machine learning applications. The goal is to project a dataset onto a lower-dimensional space with good class-separability in order avoid overfitting (“curse of dimensionality”) and also reduce computational costs.

Ronald A. Fisher formulated the Linear Discriminant in 1936 (The Use of Multiple Measurements in Taxonomic Problems), and it also has some practical uses as classifier. The original Linear discriminant was described for a 2-class problem, and it was then later generalized as “multi-class Linear Discriminant Analysis” or “Multiple Discriminant Analysis” by C. R. Rao in 1948.

The general LDA approach is very similar to a Principal Component Analysis, but in addition to finding the component axes that maximize the variance of our data (PCA), we are additionally interested in the axes that maximize the separation between multiple classes (LDA).

So, in a nutshell, often the goal of an LDA is to project a feature space (a dataset n-dimensional samples) onto a smaller subspace kk (where k≤n−1k≤n−1) while maintaining the class-discriminatory information.
In general, dimensionality reduction does not only help reducing computational costs for a given classification task, but it can also be helpful to avoid overfitting by minimizing the error in parameter estimation (“curse of dimensionality”).

Logistic regression is a classification algorithm traditionally limited to only two-class classification problems.If you have more than two classes then Linear Discriminant Analysis is the preferred linear classification technique.LDA is a simple model in both preparation and application.

Limitations of Logistic Regression

Logistic regression is a simple and powerful linear classification algorithm. It also has limitations that suggest at the need for alternate linear classification algorithms.

- Two-Class Problems. Logistic regression is intended for two-class or binary classification problems. It can be extended for multi-class classification, but is rarely used for this purpose.
- Unstable With Well Separated Classes. Logistic regression can become unstable when the classes are well separated.
- Unstable With Few Examples. Logistic regression can become unstable when there are few examples from which to estimate the parameters.

Linear Discriminant Analysis does address each of these points and is the go-to linear method for multi-class classification problems. Even with binary-classification problems, it is a good idea to try both logistic regression and linear discriminant analysis.

Representation of LDA Models

- The representation of LDA is straight forward.
- It consists of statistical properties of your data, calculated for each class. For a single input variable (x) this is the mean and the variance of the variable for each class. For multiple variables, this is the same properties calculated over the multivariate Gaussian, namely the means and the covariance matrix.
- These statistical properties are estimated from your data and plug into the LDA equation to make predictions. These are the model values that you would save to file for your model.
- Let's look at how these parameters are estimated.

LDA makes some simplifying assumptions about your data:

- That your data is Gaussian, that each variable is is shaped like a bell curve when plotted.

That each attribute has the same variance, that values of each variable vary around the mean by the same amount on average.

With these assumptions, the LDA model estimates the mean and variance from your data for each class. It is easy to think about this in the univariate (single input variable) case with two classes.

The mean (mu) value of each input (x) for each class (k) can be estimated in the normal way by dividing the sum of values by the total number of values.

muk = 1/nk * sum(x)

Where muk is the mean value of x for the class k, nk is the number of instances with class k. The variance is calculated across all classes as the average squared difference of each value from the mean.

sigma^2 = 1 / (n-K) * sum((x – mu)^2)

Where sigma^2 is the variance across all inputs (x), n is the number of instances, K is the number of classes and mu is the mean for input x.

Making Predictions with LDA

LDA makes predictions by estimating the probability that a new set of inputs belongs to each class. The class that gets the highest probability is the output class and a prediction is made.

The model uses Bayes Theorem to estimate the probabilities. Briefly Bayes' Theorem can be used to estimate the probability of the output class (k) given the input (x) using the probability of each class and the probability of the data belonging to each class:

P(Y=x|X=x) = (PIk * fk(x)) / sum(PIl * fl(x))

Where PIk refers to the base probability of each class (k) observed in your training data (e.g. 0.5 for a 50-50 split in a two class problem). In Bayes' Theorem this is called the prior probability.

PIk = nk/n

The f(x) above is the estimated probability of x belonging to the class. A Gaussian distribution function is used for f(x). Plugging the Gaussian into the above equation and simplifying we end up with the equation below. This is called a discriminate function and the class is calculated as having the largest value will be the output classification (y):

Dk(x) = x * (muk/siga^2) – (muk^2/(2*sigma^2)) + ln(PIk)

Dk(x) is the discriminate function for class k given input x, the muk, sigma^2 and PIk are all estimated from your data.

How to Prepare Data for LDA

This section lists some suggestions you may consider when preparing your data for use with LDA.

- Classification Problems. This might go without saying, but LDA is intended for classification problems where the output variable is categorical. LDA supports both binary and multi-class classification.
- Gaussian Distribution. The standard implementation of the model assumes a Gaussian distribution of the input variables. Consider reviewing the univariate distributions of each attribute and using transforms to make them more Gaussian-looking (e.g. log and root for exponential distributions and Box-Cox for skewed distributions).
- Remove Outliers. Consider removing outliers from your data. These can skew the basic statistics used to separate classes in LDA such the mean and the standard deviation.
- Same Variance. LDA assumes that each input variable has the same variance. It is almost always a good idea to standardize your data before using LDA so that it has a mean of 0 and a standard deviation of 1.

Extensions to LDA

Linear Discriminant Analysis is a simple and effective method for classification. Because it is simple and so well understood, there are many extensions and variations to the method. Some popular extensions include:

- Quadratic Discriminant Analysis (QDA): Each class uses its own estimate of variance (or covariance when there are multiple input variables).
- Flexible Discriminant Analysis (FDA): Where non-linear combinations of inputs is used such as splines.
- Regularized Discriminant Analysis (RDA): Introduces regularization into the estimate of the variance (actually covariance), moderating the influence of different variables on LDA.

The original development was called the Linear Discriminant or Fisher's Discriminant Analysis. The multi-class version was referred to Multiple Discriminant Analysis. These are all simply referred to as Linear Discriminant Analysis now.

3.2 Discriminant Analysis

Discriminant analysis is a classification method that uses statistical measures such as covariance and geometrical measurements such as Euclidean distance to determine which group an unknown data point belongs to. Using discriminant analysis requires two main steps: finding the discriminant coefficients from a well-understood set of data and applying the coefficients to unknown data to yield group classifications.

It builds a predictive model for group membership. The model is composed of a discriminant function (or, for more than two groups, a set of discriminant functions) based on linear combinations of the predictor variables that provide the best discrimination between the groups. The functions are generated from a sample of cases for which group membership is known; the functions can then be applied to new cases that have measurements for the predictor variables but have unknown group membership.

Example to show Discriminant analysis can be used:

On average, people in temperate zone countries consume more calories per day than people in the tropics, and a greater proportion of the people in the temperate zones are city dwellers. A researcher wants to combine this information into a function to determine how well an individual can discriminate between the two groups of countries. The researcher thinks that population size and economic information may also be important. Discriminant analysis allows you to estimate coefficients of the linear discriminant function, which looks like the right side of a multiple linear regression equation. That is, using coefficients a, b, c, and d, the function is:

D = a * climate + b * urban + c * population + d * gross domestic product per capita

If these variables are useful for discriminating between the two climate zones, the values of D will differ for the temperate and tropic countries. If you use a stepwise variable selection method, you may find that you do not need to include all four variables in the function.

Instruction for using Discriminant Analysis:

1. Decide on the variables you wish to include in the study. These variables should be characteristics that you believe will help classify data points into specific, mutually exclusive groups. For example, if your groups are to be "men and women," possible variables include number of children, years of schooling and yearly income.

2. Collect a set of data that can be classified into mutually exclusive groups (e.g., men and women, buyers and sellers, or Chinese and Taiwanese). Collect data on the variables that you have previously decided on for each data point.

3. Calculate the centroids for each group. The calculation of the centroids depends on the number of variables you have chosen to include in the analysis. For example, if you have decided to investigate only two variables, then your centroids will exist in Euclidean 2-space.

4. Calculate the distance between the two centroids, and denote this distance as a vector, "d." The vector will be as many dimensions as the number of variables of interest. In the case that you are investigating two variables, your vector, "d," will be two-dimensional.

5. Compute the within-group sum of squares matrices for each group. Call these matrices "W1" and "W2."

6. Pool the within-group sum of square matrices to yield a within-group covariance matrix. Call this matrix "Cw."

7. Compute the inverse of "Cw." Call this inverse matrix "Cw-1."

8. Multiply "Cw-1" and "d." Call this vector "Cw-1d." Its dimension should be equal to the number of variables you have included in the analysis.

9. Calculate the discriminant function coefficients. These coefficients are proportional to "Cw-1d."

10. Collect data of interest (data you wish to classify into groups). To properly apply discriminant analysis, only collect data on the variables of interest; knowing the classifications beforehand defeats the purpose of performing discriminant analysis.

11. Write each data point as a vector. The dimensions of the vectors are the same as the dimensions of the original set of data.

12. Classify each data point. Multiply each data point by the discriminant function coefficients. The output will give you the classification of the data point. For example, if you are using years of schooling and yearly income as variables to predict the gender of the data points, the resulting number will either be closer to "male" or "female." The group the point is closer to is the group it is classified as.

3.3 K-Nearest Neighbour Classifier

k-nearest neighbors algorithm. In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametricmethod used for classification and

regression. In both cases, the input consists of the k closest training examples in the feature space.

K- Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data).

We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

As an **e**xample, consider the following table of data points containing two features:

SATHYABAMA

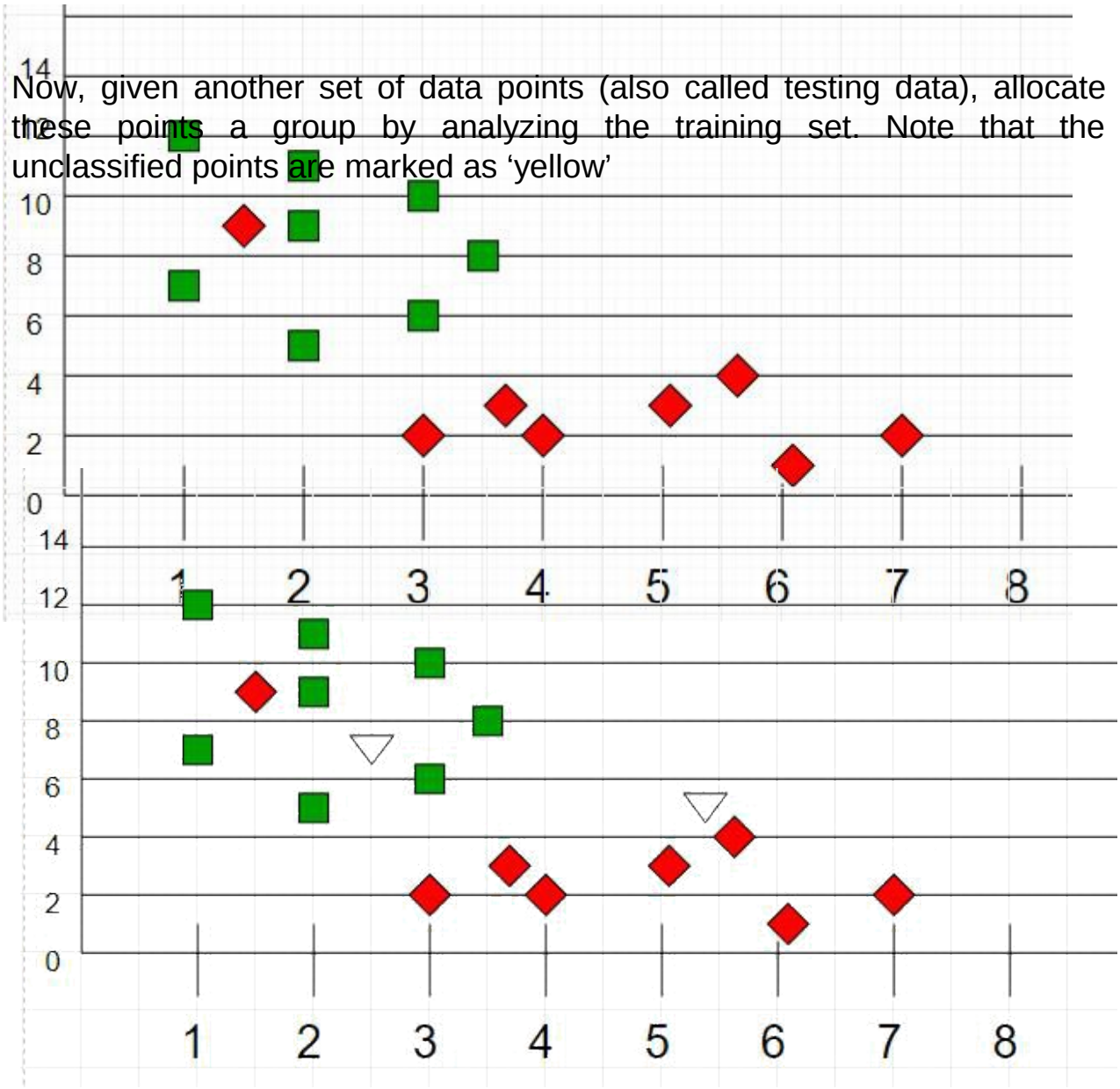
INSTITUTE OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE MATERIAL

Subject Name : DATA ANALYTICS **UNIT III** **Subject Code : SIT1303**

Now, given another set of data points (also called testing data), allocate these points a group by analyzing the training set. Note that the unclassified points are marked as 'yellow'



Intuition

If we plot these points on a graph, we may be able to locate some clusters, or groups. Now, given an unclassified point, we can assign it to a group by observing what group its nearest neighbours belong to. This means, a point close to a cluster of points classified as ‘Red’ has a higher probability of getting classified as ‘Red’.

Intuitively, we can see that the first point (2.5, 7) should be classified as ‘Blue’ and the second point (5.5, 4.5) should be classified as ‘Red’.

Algorithm

Let m be the number of training data samples. Let p be an unknown point.

1. Store the training samples in an array of data points arr[]. This means each element of this array represents a tuple (x, y).

2. for i=0 to m:

3. Calculate Euclidean distance d(arr[i], p).

4. Make set S of K smallest distances obtained. Each of these distances correspond to an already classified data point. Return the majority label among

3.4Fuzzy Clustering

Fuzzy clustering (also referred to as soft **clustering**) is a form of **clustering** in which each data point can belong to more than one **cluster**.

Clustering or Cluster Analysis involves assigning data points to clusters such that items in the same cluster are as similar as possible, while items belonging to different clusters are as dissimilar as possible. Clusters are identified via similarity measures. These similarity measures include distance, connectivity, and intensity. Different similarity measures may be chosen based on the data or the application.

Comparison to hard clustering

In non-fuzzy clustering (also known as hard clustering), data is divided into distinct clusters, where each data point can only belong to exactly one cluster. In fuzzy clustering, data points can potentially belong to multiple clusters.

Membership

Membership grades are assigned to each of the data points(tags). These membership grades indicate the degree to which data points belong to each cluster. Thus, points on the edge of a cluster, with lower membership grades, may be in the cluster to a lesser degree than points in the center of cluster.

Fuzzy C-means clustering

One of the most widely used fuzzy clustering algorithms is the Fuzzy C-means clustering (FCM) Algorithm..

General description

The fuzzy c-means algorithm is very similar to the k-means algorithm:

- Choose a number of clusters.
- Assign coefficients randomly to each data point for being in the clusters.
- Repeat until the algorithm has converged (that is, the coefficients' change between two iterations is no more than , the given sensitivity threshold) :
  - Compute the centroid for each cluster (shown below).
  - For each data point, compute its coefficients of being in the clusters.

Centroid

Any point x has a set of coefficients giving the degree of being in the kth cluster $w_k(x)$. With fuzzy c-means, the centroid of a cluster is the mean of all points, weighted by their degree of belonging to the cluster:

$$c_k = \frac{\sum_x w_k(x)^m x}{\sum_x w_k(x)^m}.$$

Algorithm

The FCM algorithm attempts to partition a finite collection of n elements $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ into a collection of c fuzzy clusters with respect to some given criterion.

Given a finite set of data, the algorithm returns a list of C cluster centres $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_c\}$ and a partition matrix

$$W = w_{i,j} \in [0,1],\ i = 1, \ldots, n,\ j = 1, \ldots, c,$$

, where each element, w_{ij} , tells the degree to which element, x_i, belongs to cluster c_j.

The FCM aims to minimize an objective function:

$$\arg\min_C \sum_{i=1}^{n} \sum_{j=1}^{c} w_{ij}^m \|\mathbf{x}_i - \mathbf{c}_j\|^2,$$

where:

$$w_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|\mathbf{x}_i - \mathbf{c}_j\|}{\|\mathbf{x}_i - \mathbf{c}_k\|} \right)^{\frac{2}{m-1}}}.$$

Comparison to K-means clustering

K-means clustering also attempts to minimize the objective function shown above. This method differs from the k-means objective function by the addition of the membership values w_{ij} and the fuzzifier, , with . The fuzzifier determines the level of cluster fuzziness. A large results in smaller membership values, , and hence, fuzzier clusters. In the limit , the memberships, , converge to 0 or 1, which implies a crisp partitioning. In the absence of experimentation or domain knowledge, is commonly set to 2. The algorithm minimizes intra-cluster variance as well, but has the same problems as k-means; the minimum is a local minimum, and the results depend on the initial choice of weights.

Clustering generalizes partition clustering methods (such as k-means and medoid) by allowing an individual to be partially classified into more than one cluster. In regular clustering, each individual is a member of only one cluster. Suppose we have K clusters and we define a set of variables mi1 mi2 miK , ,, that represent the probability that object i is classified into cluster k. In partition clustering algorithms, one of these values will be one and the rest will be zero. This represents the fact that these algorithms classify an individual into one and only one cluster. In fuzzy clustering, the membership is spread among all clusters. The mik can now be between zero and one, with the stipulation that the sum of their values is one. We call this a fuzzification of the cluster configuration. It has the advantage that it does not force every object into a specific cluster. It has the disadvantage that there is much more information to be interpreted. To understand the reason that fuzzy clustering was developed, consider the following two-variable dataset whose values are plotted below. The data have three obvious clusters and two outlier points (6 and 13). A regular clustering algorithm searching for three clusters will force these two points into specific clusters. This may cause distortion in the final solution. Fuzzy clustering, however, will assign a probability of about 0.33 for each cluster. This equal membership probability signals that these two points are outliers. When you only have two variables, you can plot your data and see what the clusters are. Unfortunately, most clustering projects come with more than two variables, so plotting is not possible. Hence, we must use techniques like fuzzy clustering to deal with the anomalies that can occur

3.5 Self Organizing Map

A self-organizing map (SOM) or self-organizing feature map (SOFM) is a type of artificial neural network (ANN) that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map, and is therefore a Self-Organizing Map.

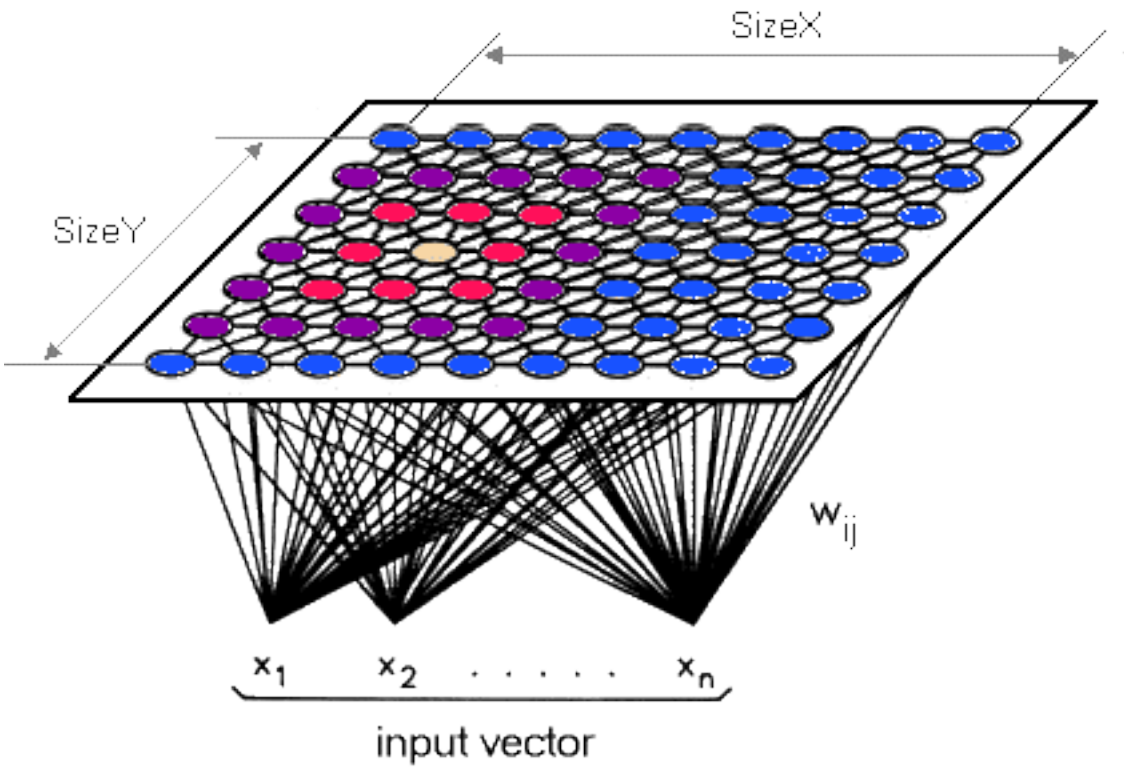
Self Organizing Map(SOM) by TeuvoKohonen provides a data visualization technique which helps to understand high dimensional data by reducing the dimensions of data to a map. SOM also represents clustering concept by

grouping similar data together. Therefore it can be said that SOM reduces data dimensions and displays similarities among data.

With SOM, clustering is performed by having several units compete for the current object. Once the data have been entered into the system, the network of artificial neurons is trained by providing information about inputs. The weight vector of the unit is closest to the current object becomes the winning or active unit. During the training stage, the values for the input variables are gradually adjusted in an attempt to preserve neighborhood relationships that exist within the input data set. As it gets closer to the input object, the weights of the winning unit are adjusted as well as its neighbors.

TeuvoKohonen writes "The SOM is a new, effective software tool for the visualization of high-dimensional data. It converts complex, nonlinear statistical relationsihps between high-dimensional data items into simple geometric relationships on a low-dimensional display. As it thereby compresses information while preserving the most important topological and metric relationships of the primary data items on the display, it may also be thought to produce some kind of abstractions."

Reducing Data DimensionsUnlike other learning technique in neural networks, training a SOM requires no target vector. A SOM learns to classify the training data without any external supervision.



Data Similarity; Getting the Best Matching Unit is done by running through all right vectors and calculating the distance from each weight to the sample vector. The weight with distance is the winner. There are numerous ways to determine the distance, however, the most commonly used method is the Euclidean Distance and/or Cosine Distance.SOM Algorithm. Each data from data set recognizes themselves by competing for representation. SOM mapping steps starts from initializing the weight vectors. From there a sample vector is selected randomly and the map of weight vectors is searched to find which weight best represents that sample. Each weight vector has neighboring weights that are close to it. The weight that is chosen is rewarded by being able to become more like that randomly selected sample vector. The neighbours of that weight are also rewarded by being able to become more like the chosen sample vector. From this step the number of neighbours and how much each weight can learn decreases over time. This whole process is repeated a large number of times, usually more than 1000 times. In sum, learning occurs in several steps and over many iterations.

1. Each node's weights are initialized.
2. A vector is chosen at random from the set of training data.
3. Every node is examined to calculate which one's weights are most like the input vector. The winning node is commonly known as the Best Matching Unit (BMU).
4. Then the neighbourhood of the BMU is calculated. The amount of neighbors decreases over time.
5. The winning weight is rewarded with becoming more like the sample vector. The nighbors also become more like the sample vector. The closer a node is to the BMU, the more its weights get altered and the farther away the neighbor is from the BMU, the less it learns.
6. Repeat step 2 for N iterations.
7. Result Interpretation.An example of the result of a Self Organizing Map is shown below

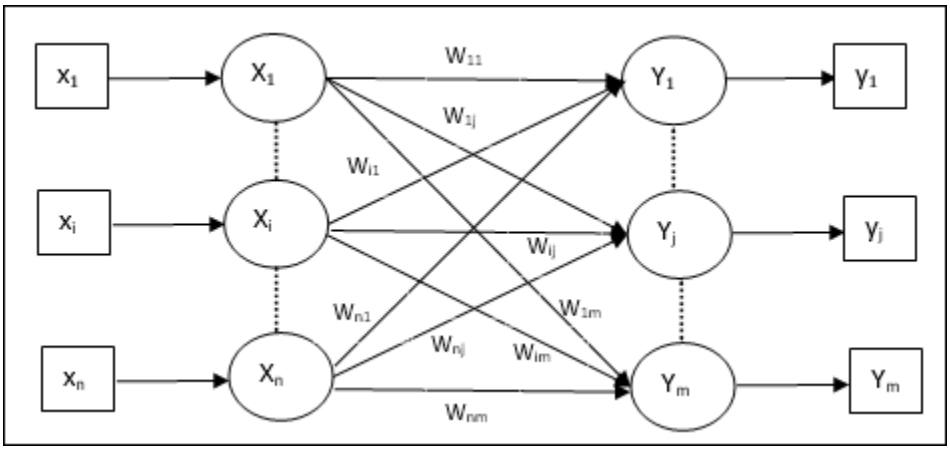
If the average distance is high, then the surrounding weights are very different and a dark color is assigned to the location of the weight. If the average distance is low, a lighter color is assigned. The resulting map shows that black is not similar to the white parts because there are lines of black representing no similarity between white parts. Looking at the map it clearly represents that the two not very similar by having black in between. It can be said that the white parts represent different clusters and the black lines represent the division of the clusters.

3.6 Learning Vector Quantization(LVQ)

LVQ is a prototype-based supervised classification algorithm. LVQ is the supervised counterpart of vector quantization systems.Learning Vector Quantization (LVQ), different from Vector quantization (VQ) and Kohonen Self-Organizing Maps (KSOM), basically is a competitive network which uses supervised learning. We may define it as a process of classifying the patterns where each output unit represents a class. As it uses supervised learning, the network will be given a set of training patterns with known classification along with an initial distribution of the output class. After completing the training process, LVQ will classify an input vector by assigning it to the same class as that of the output unit.

Architecture

Following figure shows the architecture of LVQ which is quite similar to the architecture of KSOM. As we can see, there are “n” number of input units and “m” number of output units. The layers are fully interconnected with having weights on them.



Parameters Used

Following are the parameters used in LVQ training process as well as in the flowchart

- x = training vector (x_1,...,x_i,...,x_n)

- T = class for training vector x
- w_j = weight vector for j^{th} output unit
- C_j = class associated with the j^{th} output unit

Training Algorithm

Step 1 − Initialize reference vectors, which can be done as follows −

- Step 1(a) − From the given set of training vectors, take the first “m” (number of clusters) training vectors and use them as weight vectors. The remaining vectors can be used for training.
- Step 1(b) − Assign the initial weight and classification randomly.
- Step 1(c) − Apply K-means clustering method.

Step 2 − Initialize reference vector αα

Step 3 − Continue with steps 4-9, if the condition for stopping this algorithm is not met.

Step 4 − Follow steps 5-6 for every training input vector x.

Step 5 − Calculate Square of Euclidean Distance for j = 1 to m and i = 1 to n

$$D(j)=\sum i=1n\sum j=1m(xi-wij)2D(j)=\sum i=1n\sum j=1m(xi-wij)2$$

Step 6 − Obtain the winning unit J where D(j) is minimum.

Step 7 − Calculate the new weight of the winning unit by the following relation −

if T = C_j then $wj(new)=wj(old)+\alpha[x-wj(old)]wj(new)=wj(old)+\alpha[x-wj(old)]$

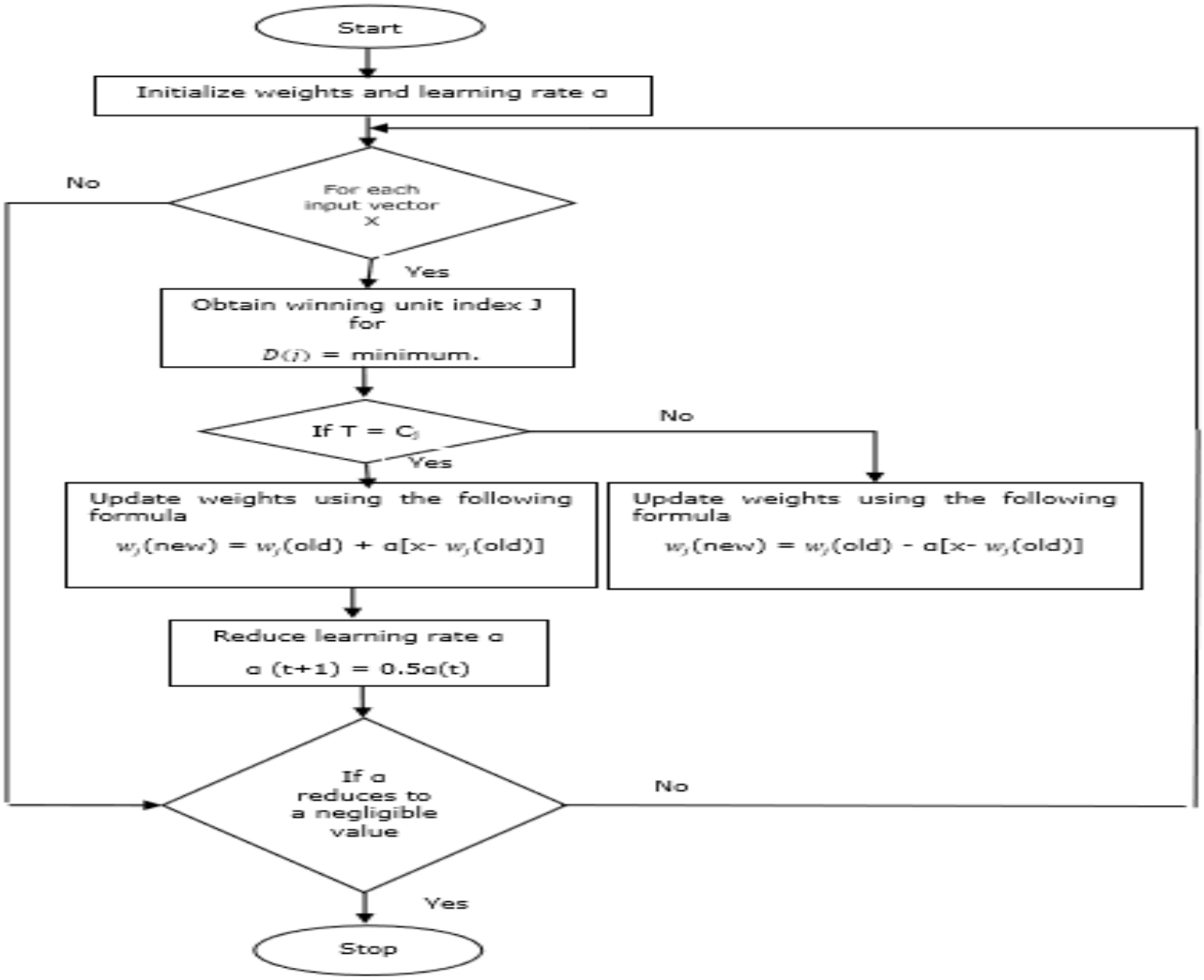
if T ≠ C_j then $wj(new)=wj(old)-\alpha[x-wj(old)]wj(new)=wj(old)-\alpha[x-wj(old)]$

Step 8 − Reduce the learning rate αα.

Step 9 − Test for the stopping condition. It may be as follows −

- Maximum number of epochs reached.
- Learning rate reduced to a negligible value.

Flowchart



3.7 Relational Clustering

Overview:

- Data clustering is the task of detecting patterns in a set of data.
- Most algorithms take non-relational data as input and are sometimes unable to find significant patterns.

- Many data sets can include relational information, as well as independent object attributes.
- Relational data clustering techniques can help find strong patterns in such sets.
- Two areas of interest in relational data clustering are: clustering heterogeneous data, and relation selection.

Heterogeneous Data

It can be very difficult to compare different typed objects. For example, how can actors be compared to directors? One possibility is an inter-cluster relation signature.

1.Cluster one set of homogeneous data. This is the reference clustering.

2. For each object, Create a vector that records the number of links from that object to each cluster discovered in step 1. This is the inter-cluster relation signature.

3. Cluster all objects based on the inter-cluster relation signatures.

Relation Selection

It is intuitive that, just as some features are not helpful for clustering a data set, some relations might provide little information for a relational clustering algorithm, or even harm the performance of an algorithm. As relational clustering algorithms continue to develop, detecting such graphs will become more important.

3.8 Partitioning Methods

i)K means method

- Establish k partitions for the given n tuples **k<=n**
- Maximize the intracluster similarity
- Minimize the intercluster similarity
- Mean value of data points in cluster à cluster similarity

- Selects k objects randomly à cluster center
- Remaining objects are assigned to nearest clusters
- Compute a new mean for the clusters
- Process gets repeated until the criteria function converges

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2$$

Algorithm k-means

Input

k number of clusters

D data set of objects

Output

set of k clusters with better quality

Method

arbitrarily choose k object from D as initial cluster centers

repeat

assign each object to the cluster to which the object is most similar

update the cluster mean

until no change in cluster mean

Strength

- Relatively scalable
- Efficient in processing large data set

Weakness

- Need to supply k / k mean values

– Not capable of forming the clusters with arbitrary shapes

– It is sensitive to noise, outliers

– Relatively scalable

Variations (EM method, k-modes method)

– Differed in

- Dissimilarity calculation Cluster mean calculation idea

ii) K – medoids method

– Also called as representative object based technique

– Influence of outliers in k means clustering algorithm gets suppressed

– Selects a representative data object per cluster

– Remaining objects are clustered based upon its similarity

– Minimize the dissimilarity between each object and its corresponding reference point (absolute error criterion)

– Process continues until every representative object is actually a medoid /most centrally located

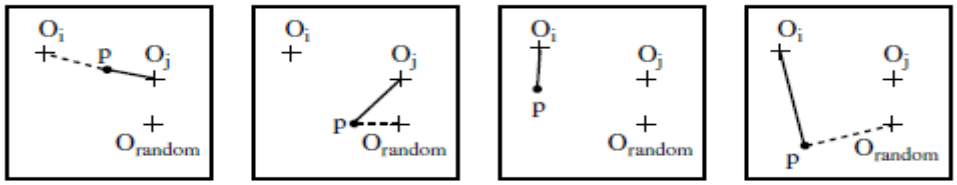
– Quality of clusters gets estimated by the cost function

– Four cases have to be examined for every data object p before O_j gets replaced by O_{random}

- Reassigned to Oi from Oj
- Reassigned to Orandom from Oj
- No change
- Reassigned to Orandom from Oi

– Cost incurred in swapping will gets computed.

– If the new value is optimal then replacement will be allowed



1. Reassigned to O_i
2. Reassigned to O_{random}
3. No change
4. Reassigned to O_{random}

- data object
+ cluster center
— before swapping
--- after swapping

Four cases of the cost function for k-medoids clustering.