Unsupervised Learning

Gustavo Velasco-Hernández

Pattern Recognition, 2014

Problems/Approaches in Machine Learning

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Problems/Approaches in Machine Learning

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

In supervised learning, every input is provided with an output. It is a problem about function approximation and there is a feedback on what response should be (target output).

Problems/Approaches in Machine Learning

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

In unsupervised learning, there is no outputs, just input data. It is a problem about describe the nature of the data and/or infer its internal structure.

Problems/Approaches in Machine Learning

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

A semi-supervised approach exists. It is about how to use labelled and unlabelled data to learn.

Problems/Approaches in Machine Learning

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

In reinforcement learning there is no supervisor. The learning is done based on interaction with the environment, through rewards and penalties. It is about learning of states, actions and its effect.

Unsupervised Learning

"*Unsupervised learning refers to any machine learning process that seeks to learn structure in the absence of either an identified output (supervised learning) or feedback (reinforcement learning). Three typical examples of unsupervised learning are: Clustering, Association rules and self-organization maps.*" - Encyclopedia of Machine Learning, Sammut, Webb, 2010

Unsupervised Learning in Literature

Due to machine learning includes many topics like pattern recognition, classification, signal processing, probability and statistics, and also it has many applications. Authors present their own approach in books and courses.
Here are some examples on how unsupervised learning topics are presented in literature:

Unsupervised Learning in Literature

Bishop, Pattern Recognition and Machine Learning:

- Ch. 9 Mixture Models and EM: K-means, MoG, EM.
- Ch. 12 Continuous Latent Variables: PCA, Probabilistic and Kernel PCA, NLCA, ICA.

Duda, Pattern Classification

- Ch. 3 ML and Bayesian Parameter estimation: EM.
- Ch. 10 U. Learning and Clustering: Mixtures, K-means, Hier. clustering, component analysis (PCA, NLCA, ICA), SOMs.

Unsupervised Learning in Literature

Theodoridis, Pattern Recognition:

- Ch. 6 Feature Selection: KLT (PCA), NLPCA, ICA.
- Ch. 11 Clustering: Basic Concepts.
- Ch. 12 Clustering I: Sequential Algorithms.
- Ch. 13 Clustering II: Hierarchical Algorithms.
- Ch. 14 Clustering III: Scheme Based and Function Optimization.
- Ch. 15 Clustering IV.
- Ch. 16 Clustering Validity.

Unsupervised Learning in Literature

Marques, Reconhecimento de padroes.

- Ch. 3: EM
- Ch. 4 Unsupervised classification: K-means, VQ, Hier. CLustering, SLC.
- Ch. 5 NN: Kohonen Maps.

Melin, Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing

- Ch. 5 Unsupervised learning neural networks: Kohonen, LVQ, Hopfield.
- Ch. 8 Clustering.

Unsupervised Learning in Literature

Webb, Statistical Pattern Recognition

- Ch. 2 Density estimation - Parametric: EM.
- Ch. 9 Feature Extraction and Selection.
- Ch. 10 Clustering.

Rencher, Methods of Multivariate Analysis

- Ch. 12 Principal Component Analysis.
- Ch. 14 Clustering Analysis.

Unsupervised Learning in Literature

- Clustering
  - Single Linkage
  - K-means
  - Soft Clustering
  - DBSCAN
- Feature Selection and Extraction
  - PCA
  - NLCA
  - ICA
  - SVD
- Networks for unsupervised Learning
  - Kohonen
  - LVQ
  - Hopfield

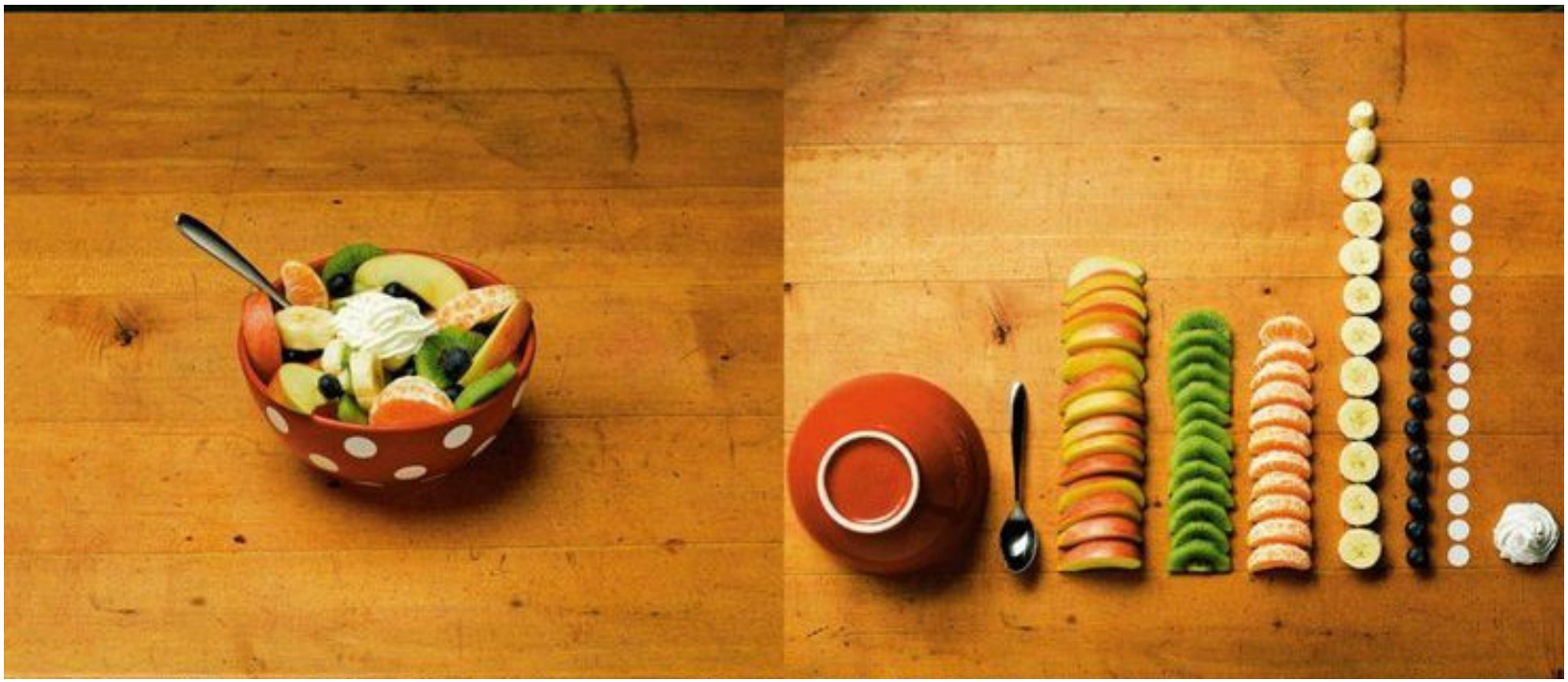
Introduction

Clustering: Take a set of objects and put them into groups in such a way that objects in the same group or cluster are more similar to each other that to those in other groups or clusters.

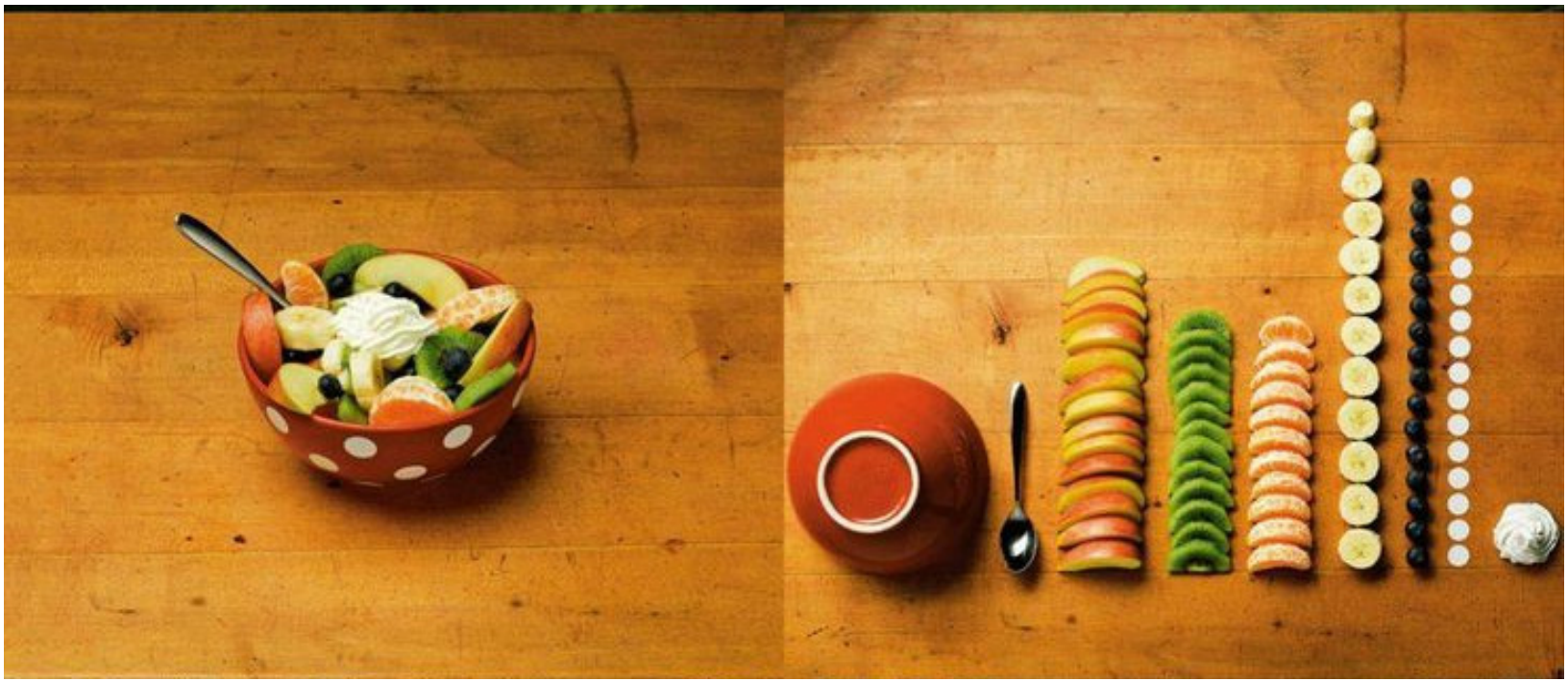
Our first clustering task

Our first clustering task

OCD clustering



OCD clustering



Basic Clustering Problem

Given:

A set of objects X

Inter-object distance matrix D, $D(x, y) = D(y, x) \forall \{x, y\} \in X$

Output:

A set of partitions $P_D(x) = P_D(y)$ if x and y in the same cluster.

Trivial clustering:

$\forall\, x \in X\ P_D(x) = 1$

$\forall\, x \in X\ P_D(x) = x$

Single Linkage Clustering

- Consider each object in a cluster (n objects)
- Define inter-cluster distance as the distance between the closest two point in the two clusters
- Merge two closest clusters
- Repeat $n - k$ times to make k clusters

K-means

- Pick k center at random
- Each center "claims" its closest points
- Recompute center by averaging clustered points
- Repeat until converge

K-means

- $P^t(x)$: Partition/Cluster of object x
- C_i^t: Set of points in cluster $i = \{x \mid P(x) = i\}$
- $Center_i^t : \frac{\sum_{y \in C_i^t} y}{|C_i|}$

K-means

- $Center_i^0$
- $P^t(x) = \underset{i}{\operatorname{argmin}} \|x - center_i^{t-1}\|^2$
- $Center_i^t : \frac{\sum_{y \in C_i^t} y}{|C_i|}$

Soft Clustering

Allows a point to be shared by two clusters, assuming that data was generated by:

- Select one of K-gaussians uniformly (Fixed known variance)
- Sample x_i from that gaussian
- Repeat n times

Soft Clustering

Task:

- Find a hypothesis $h =< \mu_1, \mu_2, \ldots, \mu_k >$ that maximises the probability of the data (Maximum Likelihood)

Expectation Maximization

Expectation:

- Expectation: $E[z_{ij}] = \frac{P(x{=}x_i|\mu{=}\mu_j)}{\sum_{i=1}^{k} P(x{=}x_i|\mu{=}\mu_j)}$

Maximization:

- Maximization: $\mu_j = \frac{\sum_i E[z_{ij}].x_i}{\sum_i E[z_{ij}]}$

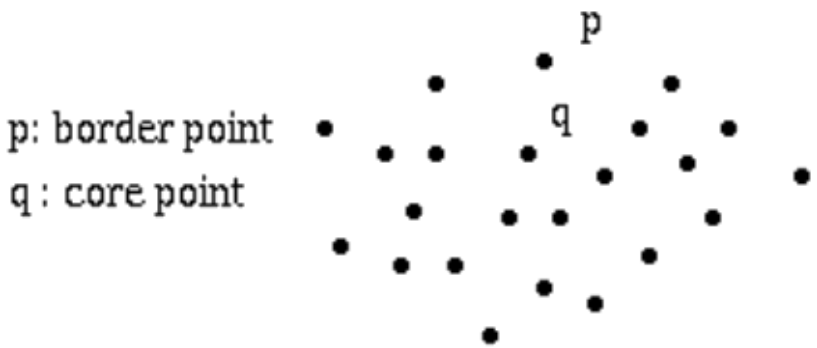
*$P(x = x_i|\mu = \mu_j) = e^{(-1/2)*\sigma^2*(x_j-u_j)^2}$

DBSCAN - Introduction

- Density-Based Spatial Clustering for Applications with Noise
- Proposed by Ester et al in 1996 in KDD conference [Ester96].

DBSCAN - Explanation

- The algorithm needs two parameters: *Eps*(ϵ) and *minPts*
- Also are defined two types of points: Core points and border points
- *p* is a core point if in its *Eps-Neighborhood* are at least *minPts* points.



Types of points [Ester96]

DBSCAN - Algorithm

- DBSCAN starts at an arbitrary point p, then evaluate if point's *Eps-Neighnorhood* contains at least *minPts* points
- If *True*, p is a core point (Is in a cluster)
  - Assign *clusterId* to p and its neighbour, and neighbours of its neighbours and so on.
  - Increase *clusterId*.
- If *False*, p is labelled as Noise
- Continue with an unlabelled point, until all points in dataset are labelled.

Introduction

Also referred as Karhunen-Loeve Transform, is a statistical procedure to convert a set of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

Procedure

Let Φ_k be the eigenvector corresponding to the kth eigenvalue λ_k of the covariance matrix Σ_x,i.e.:

$$\Sigma_x \Phi_k = \lambda_k \Phi_k \ (k = 1, \ldots, N)$$

As the covariance matrix Σ_x is hermitian, eigenvectors ϕ_i's are ortogonals.

Procedure

We can define the matrix Φ as:

$$\Phi = [\phi_1, \ldots, \phi_N]$$

satisfying:

$$\Phi^T \Phi = I \ i.e. \ \Phi^{-1} = \Phi^T$$

The N eigenequations above can be combined to expressed as:

$$\Sigma_x \Phi = \Phi \Lambda$$

where $\Lambda = diag(\lambda_1, \ldots, \Lambda_N)$

Procedure

Now, given a signal vector x, Karunen-Loeve Transform is defined as:

$$y = \Phi^T x$$

where the i^{th} component y_i of the transform vector, is the projection of x_i onto ϕ_i:

$$y_i = \langle \phi_i, x_i \rangle = \phi_i^T x$$

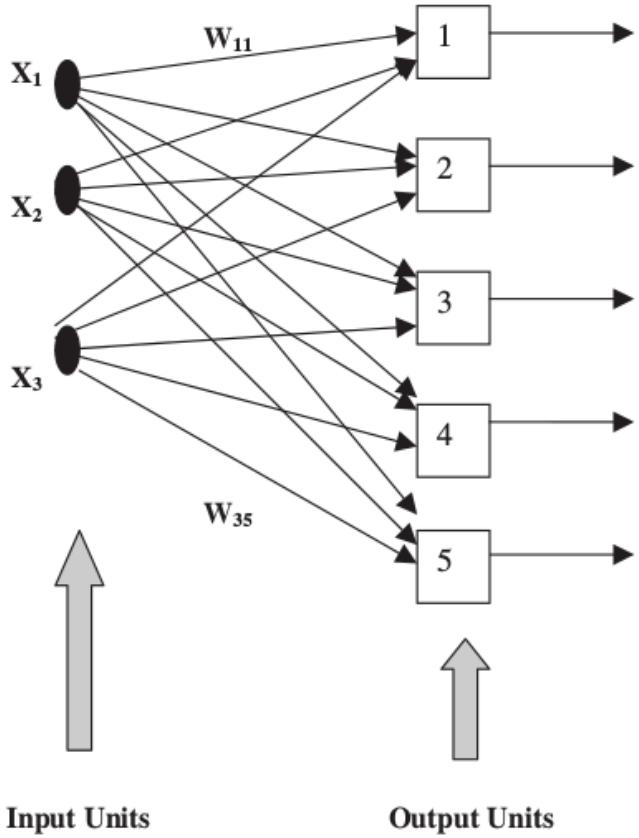
Introduction

Unsupervised learning Neural Networks attempt to learn to respond to different input patterns with different parts of the network. The network is often trained to strengthen firing to respond to frequently occurring patterns. In this manner, the network develops certain internal representations for encoding input patterns.

Competitive Networks

With no available information regarding the desired outputs, unsupervised learning networks update weights only on the basis of the input patterns. The competitive learning network is a popular scheme to achieve this type of unsupervised data clustering or classification.

Competitive Networks



Competitive Learning Network [MallinXX]

Input vector:

$$X = [x_1, x_2, x_3]^T$$

Weight vector for output unit j:

$$W_j = [w_{1j}, w_{2j}, w_{3j}]^T$$

Activation value for output unit j;

$$a_j = \sum_{i=1}^{3} x_i w_{ij} = X^T W_j = W_j^T X$$

Competitive Networks

The output unit with the highest activation must be selected for further processing. Assuming that output unit k has the maximal activation, the weights leading to this unit are updated according to the competitive or the so-called winner-take-all learning rule:

$$w_k(t+1) = \frac{w_k(t) + \eta(x(t) - w_k(t))}{\|w_k(t) + \eta(x(t) - w_k(t))\|}$$

Competitive Networks

Using the Euclidean distance as a dissimilarity measure is a more general scheme of competitive learning, in which the activation of output unit j is

$$a_j = (\sum_{i=1}^{3} (x_i - w_{ij})^2)^{0.5} = \|x - w_j\|$$

The weights of the output unit with the smallest activation are updated according to:

$$w_k(t+1) = w_k(t) + \eta(x(t) - w_k(t))$$

Competitive Networks

Here two metrics of similarity are introduced: the similarity measure of inner product and the dissimilarity measure of the Euclidean distance. Obviously, other metrics can be used instead, and different selections lead to different clustering results.

Competitive Networks

When the Euclidean distance is adopted, it can be proved that the update formula is actually an on-line version of gradient descent that minimizes the following objection function:

$$E = \sum_{p} \| w_{f(x_p)} - x_p \|^2$$

$f(x_p)$ is the wining neuron when input x_p is presented. $w_{f(x_p)}$ is the center of the class where x_p belongs to.

Competitive Networks

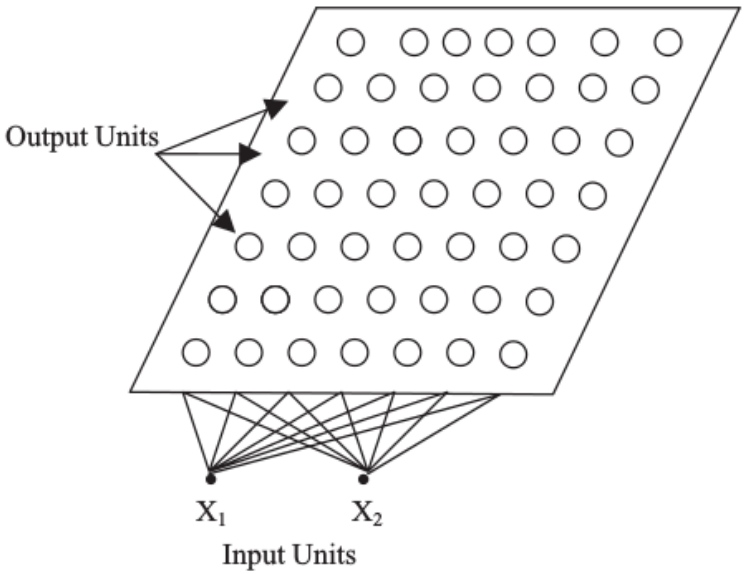
Dynamically changing the learning rate η in the weight update formula is generally desired. An initial large value of η explores the data space widely; later on, a progressively smaller value refines the weights.

If the output units of a competitive learning network are arranged in a geometric manner (such as in a one-dimensional vector or two-dimensional array), then we can update the weights of the winners as well as the neighboring losers.

Introduction

Kohonen self-organizing networks, also known as Kohonen feature maps or topology-preserving maps, are another competition-based network paradigm for data clustering (Kohonen, 1982, 1984). Networks of this type impose a neighborhood constraint on the output units, such that a certain topological property in the input data is reflected in the output units' weights.

Introduction



Kohonen Map with 2 input and 49 output units.

The learning procedure of Kohonen feature maps is similar to that of competitive learning networks. That is, a similarity (dissimilarity) measure is selected and the winning unit is considered to be the one with the largest (smallest) activation.

Training

- Select wining output unit between all weight vectors w_i and the input vector x:

$$\|x - w_c\| = \min_i \|x - w_i\|$$

Training

- Let NB_c denote a set of index corresponding to a neighborhood around winner c. New weights are updated by:

$$\Delta w_i = \eta(x - w_i), \ i \in NB_c$$

Where η is a small positive learning rate.

- Also is possible not define a neighborhood and use a Gaussian function instead:

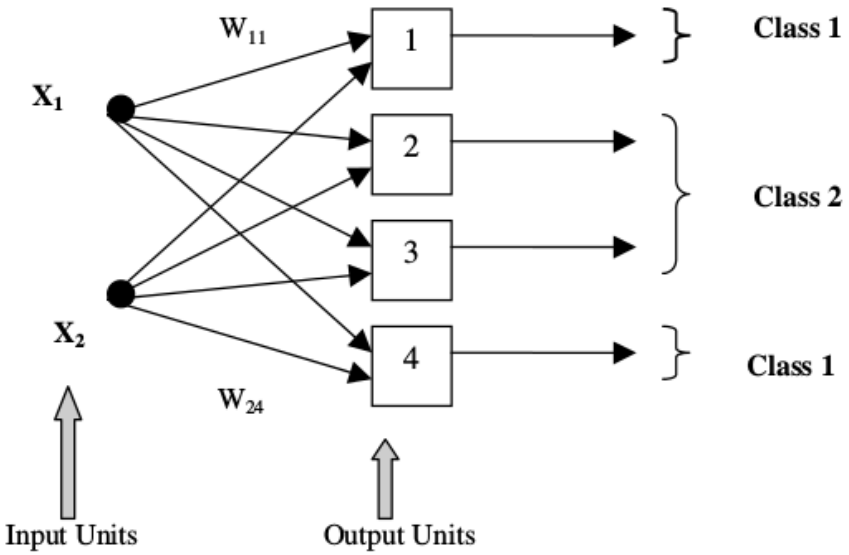
$$\Omega_{c(i)} = exp(\frac{-\|p_i - p_c\|^2}{2\sigma^2})$$

$$\Delta w_i = \eta\Omega_{c(i)}(x - w_i)$$

Introduction

Learning Vector Quantization (LVQ) is an adaptive data classification method based on training data with desired class information (Kohonen, 1989). Although a supervised training method, LVQ employs unsupervised data-clustering techniques (e.g., competitive learning) to preprocess the data set and obtain cluster centers. LVQ's network architecture closely resembles that of a competitive learning network, except that each output unit is associated with a class.

Introduction



LVQ network representation.

LVQ's network architecture closely resembles that of a competitive learning network, except that each output unit is associated with a class. Figure 5.11 presents an example, where the input dimension is 2 and the input space is divided into four clusters. Two of the clusters belong to class 1, and the other two clusters belong to class 2.

Training

The LVQ learning algorithm involves two steps. In the first step, an unsupervised learning data clustering method is used to locate several cluster centers without using the class information. In the second step, the class information is used to fine-tune the cluster centers to minimize the number of misclassified cases.

Training

- Initialise the cluster centers by a clustering method.
- Label each cluster by the voting method
- Randomly select a training input vector x and find k such that $\|x - w_k\|$ is a minimum.
- If x and w_k belong to the same class, update w_k by:

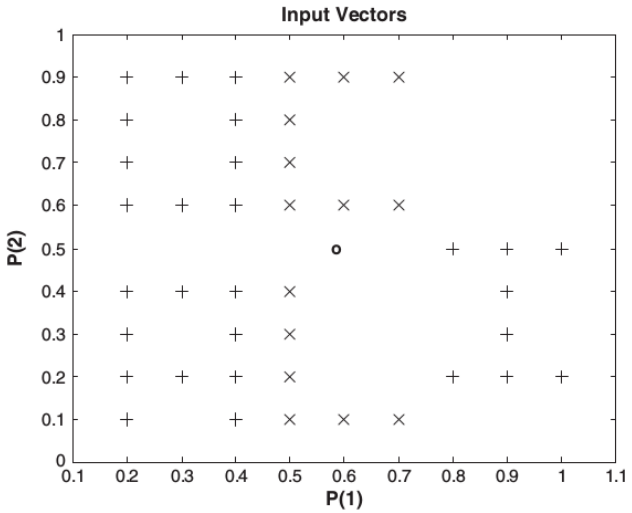
$$\Delta w_k = \eta(x - w_k)$$

Otherwise, update w_k by:

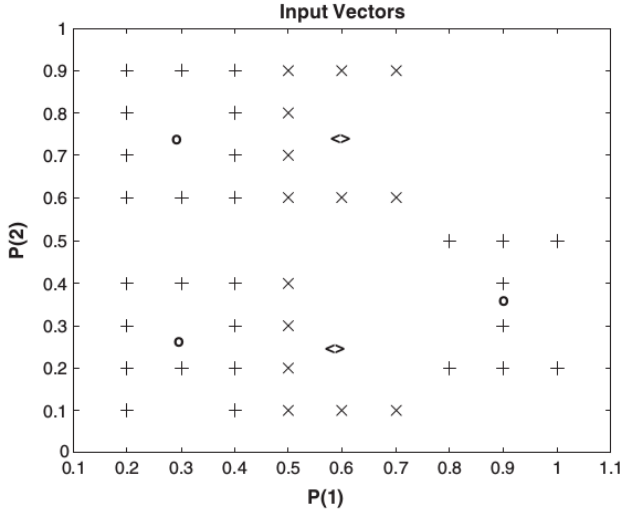
$$\Delta w_k = -\eta(x - w_k)$$

*The learning rate η is a positive small constant and should decrease in value with each respective iteration.

Example



Initial Approximation of LVQ.



Final Classification achieved

Code available at:
http://github.com/gustavovelascoh/unsupervised-learning-class