UNIT 4- Competitive Learning Neural Network

TE AIDS

Prof. Shilpa Gaikwad

Components of Competitive Learning Neural Network

Competitive learning is a type of unsupervised learning in neural networks where neurons compete with each other to become active based on input stimuli. The most commonly used competitive learning neural network is the Self-Organizing Map (SOM). The components of a competitive learning neural network, specifically the SOM, typically include:

1. Neurons (Nodes): Neurons represent the processing units in the network. In the case of a SOM, neurons are organized in a two-dimensional grid or lattice.

2. Input Data: Competitive learning neural networks receive input data, which could be a vector or a set of features. Each input is presented to the network for processing.

3. Weight Vectors: Each neuron in the competitive learning network has an associated weight vector. The weight vector represents the internal state of the neuron and is usually the same length as the input data. The weights are adjusted during the learning process.

4. Distance Metric: A distance metric or similarity measure is used to quantify the similarity between the input data and the weight vectors of the neurons. Common distance metrics include Euclidean distance or cosine similarity.

5. Activation Function: An activation function determines the level of activation or output produced by a neuron based on its input and internal state. In competitive learning, the activation function often follows a winner-takes-all approach, where the neuron with the highest activation becomes the winner.

6. Learning Rule: The learning rule defines how the weights of the neurons are updated based on the input data and the winning neuron. The learning rule aims to adjust the weights to increase the similarity between the winning neuron and the input data.

7. Neighborhood Function: In a competitive learning network, neighboring neurons of the winning neuron are also influenced during the learning process. The neighborhood function determines the degree of influence or adjustment applied to the neighboring neurons' weights.

8. Learning Rate: The learning rate determines the speed at which the weights of the neurons are updated during the learning process. It controls the magnitude of weight adjustments based on the input data.

These components work together in a competitive learning neural network to facilitate self-organization and clustering of input data, where similar inputs are grouped together in the network's output space.

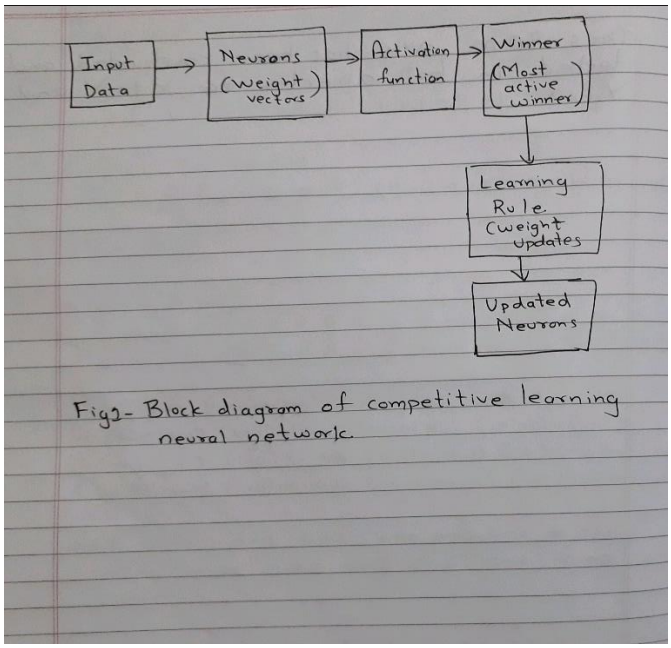


Fig2- Block diagram of competitive learning neural network.

In this diagram, the input data is fed into the neurons, each with its weight vector. The activation function calculates the level of activation for each neuron based on its internal state and the input data. The neuron with the highest activation is determined as the winner. The learning rule updates the weights of the winning neuron and potentially its neighboring neurons based on a learning rate and neighborhood function. This process iterates over multiple input data samples, allowing the network to self-organize and form clusters based on similarities in the input data.

Pattern clustering and feature mapping network

Pattern clustering and feature mapping networks typically refer to Self-Organizing Maps (SOMs). SOMs are a type of competitive learning neural network used for unsupervised learning and dimensionality reduction. They are capable of clustering similar patterns in the input data and mapping them onto a lower-dimensional grid, often visualized as a two-dimensional map.

What are Self-Organizing Maps?

Invented by Prof. Teuvo Kohonen, Kohonen artificial neural networks or Kohonen maps is another breed of Artificial Neural Networks (ANNs). More famously known as the Self-Organizing Map (SOMs) it is a very rare facet of unsupervised learning and is a data visualization technique.

The goal of the technique is to reduce dimensions and detect features. The maps help to visualize high-dimensional data. It represents the multidimensional data in a two- dimensional space using the self-organizing neural networks. The technique is used for data mining, face recognition, pattern recognition, speech analyses, industrial and medical diagnostics, anomalies detection.

The beauty of SOMs is that it preserves the topological relationship of the training dataset (or the input). Implying that the underlying properties of the input data are not affected even after the continuous change in the shape or the size of the figure. SOMs help to reveal correlations that are not easily identified.

The Architecture of Self-Organizing Maps

The self-organizing map's structure is a lattice of neurons. In the grid, X and Y coordinate, each node has a specific topological position and comprises a vector of weights of the same dimension as that of the input variables.

The input training data is a set of vectors of n dimensions: x1, x2, x3…xn, and each node consists of a corresponding weight vector W of n dimensions: w1, w2, w3… wn

In the illustration below, the lines between the nodes are simply a representation of adjacency and not signifying any connection as in other neural networks.

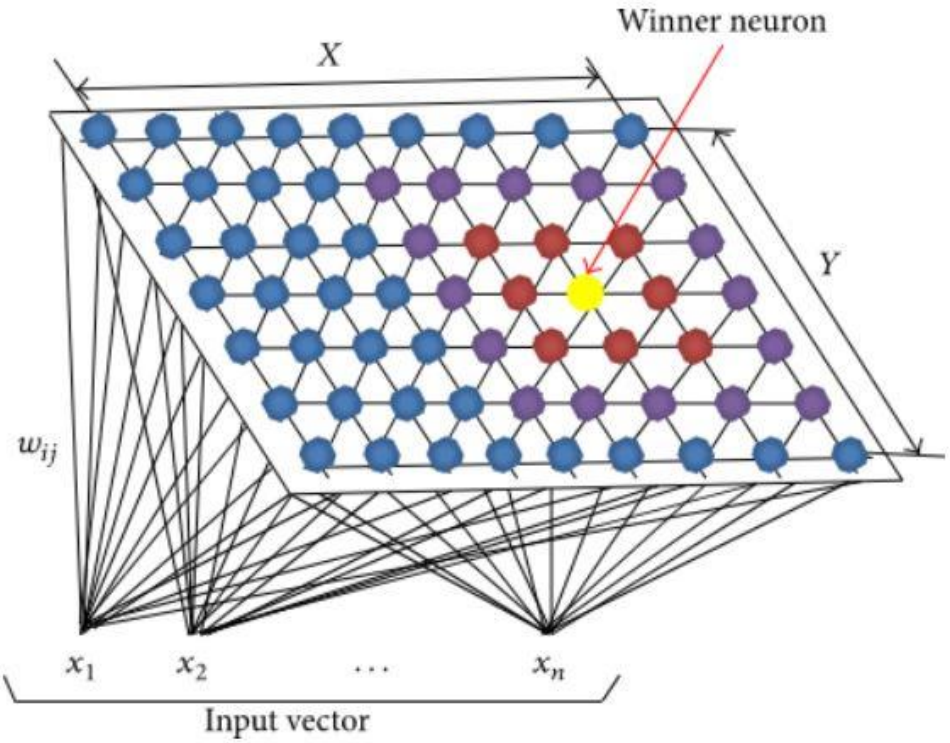


Fig. 1. Architecture of Self-Organizing Maps

Self-Organizing Maps are a lattice or grid of neurons (or nodes) that accepts and responds to a set of input signals. Each neuron has a location, and those that lie close to each other represent clusters with similar properties. Therefore, each neuron represents a cluster learned from the training.

In SOMs, the responses are compared and a 'winning' neuron is selected from the lattice. This selected neuron is activated together with the neighborhood neurons. SOMs return a two-dimensional map for any number of indicators as output where there are no lateral connections between output nodes.

Here's a question for you, what do you think is the loss function that needs to be computed for SOMs? Well, as it is an unsupervised learning technique, there is no target variable and hence, we do not even calculate the loss function and therefore there is no backward propagation process also needed for SOMs.

How do Self-Organizing Maps Learn?

What is the learning mechanism of the self-organizing maps algorithm?

The underlying idea of the SOMs training process is to examine every node and find the one node whose weight is most like the input vector. The training is carried out in a few steps and over many iterations. Let's see this in detail below:

We have three input signals x1, x2, and x3 (Fig.2). This input vector is chosen at random from a set of training data and presented to the lattice. So, we have a lattice of neurons in the form of a 3*3 two-dimensional array having nine nodes with three rows and three columns depicted like below:

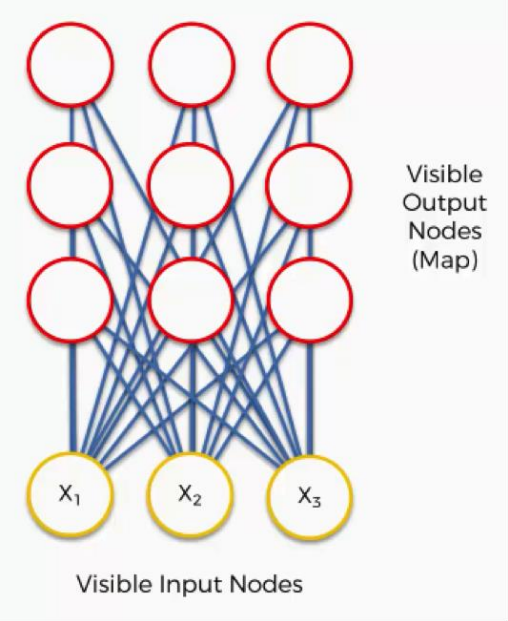


Fig.2

For understanding purposes, visualize the above 3*3 matrix as below (Fig.3):

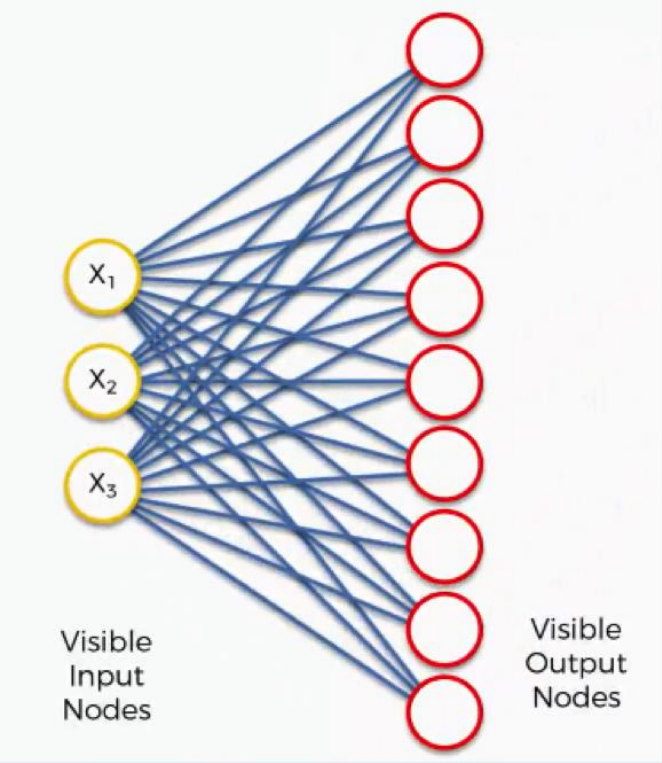


Fig.3

Each node has some random values as weights (Fig.4). These weights are not of the neural network as shown as:

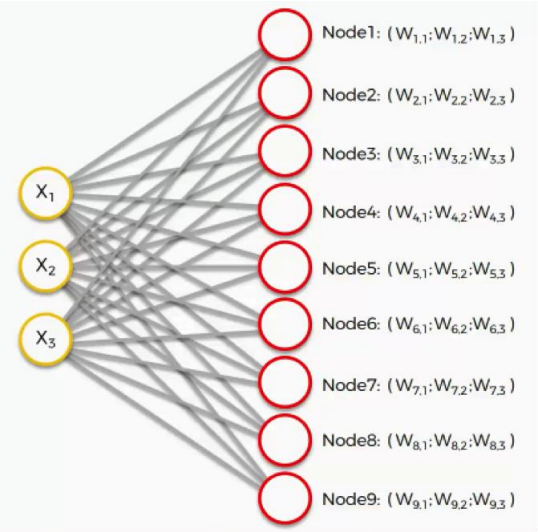


Fig.4

From these weights, can calculate the Euclidean distance as:

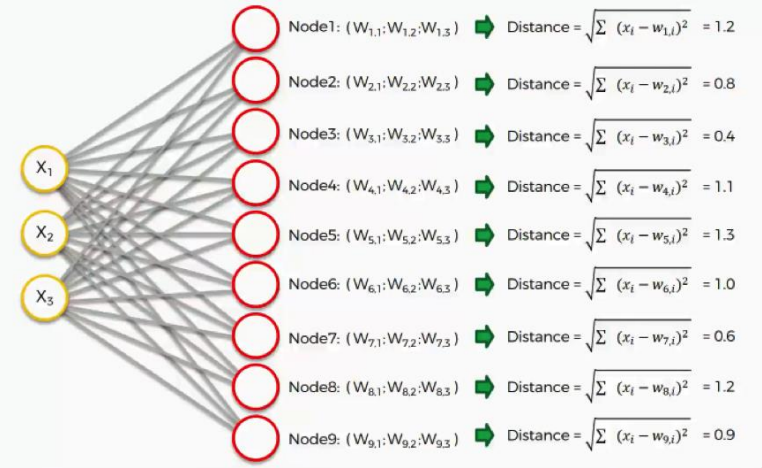


Fig.5

The training process of SOMs follows:

1. Firstly, randomly initialize all the weights.
2. Select an input vector x = [x1, x2, x3, … , xn] from the training set.
3. Compare x with the weights wj by calculating Euclidean distance for each neuron j. The neuron having the least distance is declared as the winner. The winning neuron is known as Best Matching Unit (BMU)
4. Update the neuron weights so that the winner becomes and resembles the input vector x.
5. The weights of the neighboring neuron are adjusted to make them more like the input vector. The closer a node is to the BMU, the more its weights get altered. The parameters adjusted are learning rate and neighborhood function. (more below on neighborhood function)
6. Repeat from step 2 until the map has converged for the given iterations or there are no changes observed in the weights.

Best Matching Unit (BMU)

As we saw above the distance of each of the nodes (or raw data point) is calculated from all the output nodes. The node or neuron which is identified with the least distance is the Best Matching Unit (BMU) or neighborhood. It is depicted as (m) in the following figure:

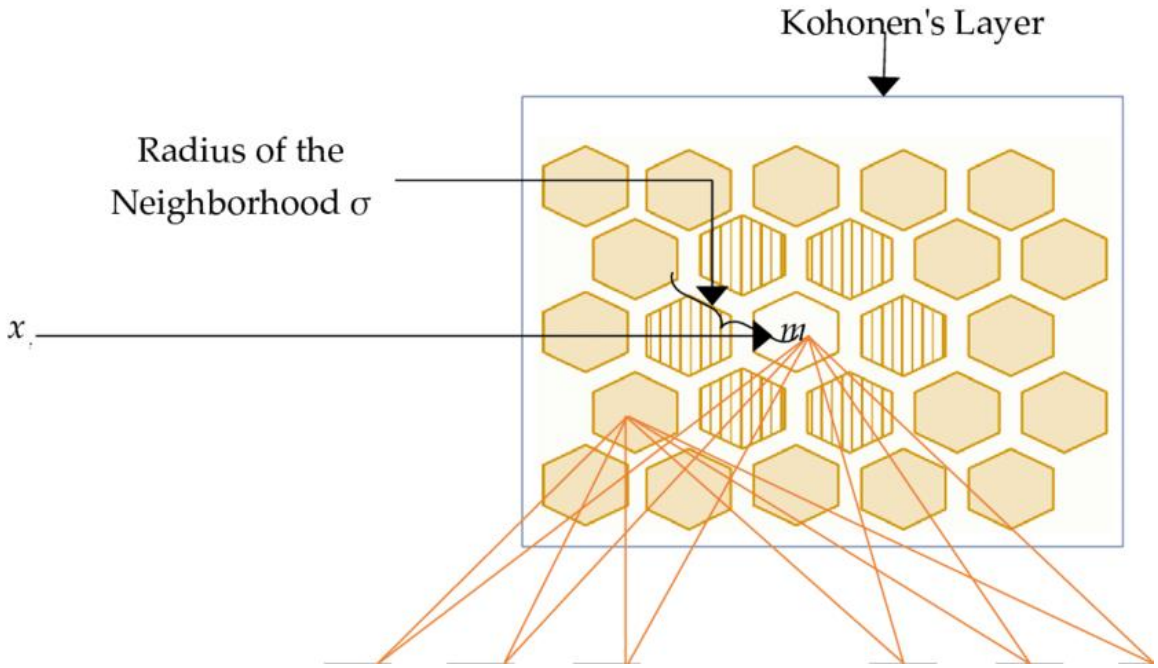


Fig.6

From this BMU, a radius (or sigma) is defined. All the nodes that fall in the radius of the BMU get updated according to their respective distance from the BMU.

So, now the point is that a neuron in the output map can be a part of the radius of many different Best Matching Units. This leads to a constant push and pulls effect on the neuron however, the behavior of the neuron will be more similar to the Best Matching Unit near it.

After each iteration, the radius shrinks and the BMU pulls lesser nodes so the process becomes more and more accurate and that shrinks the neighborhood that reduces the mapping of those segments of those data points which belong to that particular segment. This makes the map more and more specific and ultimately it starts converging and becomes like the training data (or input).

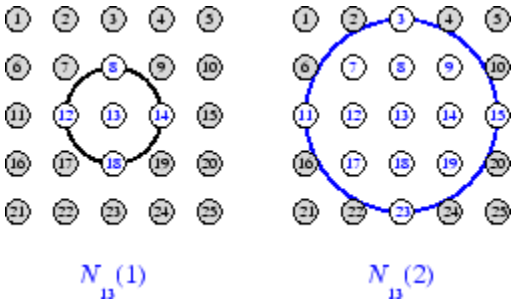


Fig.7

In the above figure, the radius of the neighborhood of the Best Matching Unit (BMU) is calculated. This is a value that starts large as on the right side of the panel, typically set to the radius of the lattice but diminishes each time step. Any nodes found within this radius are deemed to be inside the BMU's neighborhood.

The Analogy of SOMs to K-means Clustering

Both K-Means clustering and SOMs are a means to convert the higher dimensional data to lower dimensional data. Though one is a machine learning algorithm and another is a deep learning algorithm yet there is an analogy between the two.

In K-Means clustering, when new data points are presented, the centroids are updated and there are methods such as the elbow method to find the value of K. Whereas in Self Organizing Maps, the architecture of the neural network is changed to update the weights and the number of neurons is a tuning parameter.

SOM Algorithm

Training:

Step 1: Initialize the weights wij random value may be assumed. Initialize the learning rate α.

Step 2: Calculate squared Euclidean distance.

$$D(j) = \Sigma (wij - xi)^2 \quad \text{where } i=1 \text{ to } n \text{ and } j=1 \text{ to } m$$

Step 3: Find index J, when D(j) is minimum that will be considered as winning index.

Step 4: For each j within a specific neighborhood of j and for all i, calculate the new weight.

$$wij(new)=wij(old) + \alpha[xi - wij(old)]$$

Step 5: Update the learning rule by using:

$$\alpha(t+1) = 0.5 * t$$

Step 6: Test the Stopping Condition

Two basic feature mapping models:

Two basic feature mapping models commonly used in neural networks are the Self-Organizing Map (SOM) and the Gaussian Mixture Model (GMM).

Two basic feature mapping models commonly used in neural networks are the Self-Organizing Map (SOM) and the Gaussian Mixture Model (GMM).

1. Self-Organizing Map (SOM):

The Self-Organizing Map, also known as Kohonen network, is a type of unsupervised learning model that uses a grid of neurons to map input patterns onto a lower-dimensional space. The SOM clusters similar patterns together and preserves the topological relationships between the patterns in the input space. It achieves this by iteratively adjusting the weight vectors of the neurons based on a competitive learning process.

The SOM consists of a two-dimensional grid of neurons, with each neuron having an associated weight vector. During training, input patterns are presented to the SOM, and the neuron with the closest weight vector to the input pattern is selected as the winner. The weights of the winning neuron and its neighboring neurons are updated to move closer to the input pattern, allowing for clustering and

mapping of similar patterns onto the grid. The SOM can be visualized as a map, where similar patterns are located close to each other on the grid.

2. Gaussian Mixture Model (GMM):

A Gaussian Mixture Model is a probabilistic model that represents the distribution of data as a mixture of several Gaussian distributions. It is often used for clustering and density estimation tasks. In the context of feature mapping, a GMM can be used to represent the underlying structure of the input data.

A GMM assumes that the input data is generated from a combination of Gaussian distributions, each characterized by its mean and covariance. The model learns the parameters of the Gaussian components, such as the means and covariances, based on the input data. Once trained, the GMM can be used to estimate the likelihood or probability of new data points belonging to each Gaussian component, allowing for clustering and feature mapping.

The GMM can be used to map input data onto a lower-dimensional space by considering the most likely Gaussian component for each data point. This can provide a compact representation of the input data while preserving the inherent structure captured by the GMM.

Both the Self-Organizing Map and the Gaussian Mixture Model offer feature mapping capabilities, albeit using different approaches. The choice between them depends on the specific requirements of the task at hand and the nature of the input data.

There are several other feature mapping models used in neural networks for various tasks. Here are a few examples:

1.Autoencoders:

Autoencoders are neural network models that aim to learn efficient representations or encodings of the input data. They consist of an encoder network that maps the input data to a lower-dimensional latent space, and a decoder network that reconstructs the input data from the latent representation. The latent space acts as a feature mapping, capturing essential features and reducing dimensionality. Autoencoders can be trained using unsupervised learning and are often used for tasks such as dimensionality reduction, data compression, and anomaly detection.

2.t-SNE (t-Distributed Stochastic Neighbor Embedding):

t-SNE is a nonlinear dimensionality reduction technique that is commonly used for visualizing high-dimensional data. It maps the input data to a lower-dimensional space while preserving the pairwise similarities between data points. t-SNE is particularly effective at preserving the local structure and clusters in the data. It is widely used for visualization and exploratory data analyses.

3.Deep Belief Networks (DBNs):

Deep Belief Networks are generative models that consist of multiple layers of restricted Boltzmann machines (RBMs) or other unsupervised learning models. DBNs can learn hierarchical representations of the input data by iteratively training each layer based on the output of the previous layer. The hidden layers of the DBN act as a feature mapping, gradually capturing more complex and abstract features. DBNs have been used for tasks such as feature learning, classification, and generative modeling.

4.Independent Component Analyses (ICA):

ICA is a statistical technique used to separate a multivariate signal into independent components. It assumes that the observed signals are linear mixtures of unknown source signals and aims to find a linear transformation that maximizes the independence between the components. ICA can be used for feature mapping, as it identifies statistically independent components that represent different sources or factors in the data. It has applications in signal processing, blind source separation, and image analyses.

These are just a few examples of feature mapping models used in neural networks. Different models have their strengths and are suitable for different types of data and tasks. The choice of model depends on the specific requirements and characteristics of the problem at hand.

Properties of Feature Map

The properties of a feature map in the context of neural networks depend on the specific model and the purpose of feature mapping. However, here are some general properties that are often desirable in feature maps:

1. Dimensionality Reduction: Feature maps aim to reduce the dimensionality of the input data. They map high-dimensional input space to a lower-dimensional feature space, typically with fewer dimensions than the original data. This reduction in dimensionality helps in simplifying the representation of data, reducing computational complexity, and extracting essential features.

2. Preservation of Information: A good feature map should preserve important information from the input data. It should capture the underlying structure, patterns, and relationships present in the data. This property ensures that relevant features are retained, enabling effective downstream tasks such as classification, clustering, or visualization.

3. Discriminative Power: Feature maps should possess discriminative power, meaning they should enhance the differences between different classes or categories in the data. This property enables better separation and distinction of data points belonging to different classes, making subsequent tasks like classification more accurate and reliable.

4. Invariance: Invariance refers to the property of a feature map where it remains robust to certain transformations or variations in the input data. For example, an image feature map that is invariant to translation, rotation, or scale changes can still recognize the same object regardless of its position or orientation. Invariance makes feature maps more generalizable and capable of handling variations in the input data.

5. Nonlinearity: Nonlinearity in feature maps allows for capturing complex relationships and dependencies in the data. Linear feature maps are limited to capturing linear patterns, while nonlinear feature maps can model more intricate and nonlinear patterns. Nonlinearity is often introduced using activation functions or through the architecture of the model.

6. Interpretability: Interpretability is an important property in certain applications, where the feature map should be human-understandable or interpretable. This allows users or domain experts to gain insights

into the underlying factors or representations captured by the feature map, aiding in decision-making or problem understanding.

It's important to note that not all feature maps possess all of these properties simultaneously. Different feature mapping techniques and models prioritize different properties based on the specific task and requirements. The choice of feature map depends on the trade-offs and considerations relevant to the application at hand.

Computer simulation in competitive learning neural network

Computer simulations play a vital role in studying and understanding competitive learning neural networks. They enable researchers and practitioners to explore the behavior, dynamics, and performance of these networks under different conditions and input data. Through simulations, various aspects of competitive learning can be investigated, such as convergence properties, clustering behavior, and the impact of different parameters.

Computer simulations are typically used in competitive learning neural networks:

1. Network Architecture and Initialization: The simulation begins by defining the architecture of the competitive learning neural network, including the number of neurons, their arrangement (e.g., a grid or lattice), and the dimensions of the input data. The initial weights of the neurons are usually randomized or set according to specific initialization strategies.

2. Input Data Generation: Simulations require input data to be generated or provided. The input data could be synthetic data designed to exhibit specific properties or real-world data samples. The data is preprocessed and formatted appropriately for presentation to the network.

3. Learning Process: The simulation iteratively presents the input data to the network, one sample at a time or in batches. The network calculates the activations of neurons based on their weight vectors and the similarity measure (e.g., Euclidean distance). The neuron with the highest activation becomes the winner. The winning neuron, along with its neighbors, undergo weight updates based on the learning rule and the neighborhood function. This process continues for multiple iterations or until convergence criteria are met.

4. Visualization and Analyses: Simulations often involve visualizing the results to gain insights into the behavior of the competitive learning network. The evolution of the weight vectors and the mapping of input data onto the neural network's grid can be visualized to observe clustering patterns and feature mapping. Various performance metrics, such as quantization error or topographic error, can be calculated to evaluate the network's performance and convergence.

5. Parameter Tuning and Sensitivity Analyses: Simulations provide the flexibility to explore the effects of different parameters on the network's behavior. Parameters such as learning rate, neighborhood size, and initialization schemes can be adjusted and tested to understand their impact on the learning process and the quality of the feature mapping.

6. Experimentation and Comparison: Computer simulations allow for conducting experiments to compare different variants of competitive learning networks or to compare them with other algorithms or models. Through systematic comparisons, researchers can gain insights into the strengths and weaknesses of different approaches and refine their understanding of competitive learning.

Overall, computer simulations provide a valuable tool for studying and experimenting with competitive learning neural networks, allowing researchers to explore their behavior, optimize parameters, and gain insights into their application to various tasks.

Adaptive pattern classification

Adaptive pattern classification refers to the ability of a classification system or model to adapt and learn from incoming data to improve its classification accuracy over time. In adaptive pattern classification, the model dynamically adjusts its decision boundaries or classification rules based on new or evolving patterns in the data.

Here are some key concepts and techniques related to adaptive pattern classification:

1. Online Learning: Adaptive pattern classification often involves online learning, where the model incrementally updates its parameters or decision boundaries with each new data point. Online learning allows the model to adapt in real-time as new data becomes available.

2. Incremental Learning: Incremental learning is a technique used in adaptive pattern classification to update the model's parameters or decision boundaries without retraining the entire model from scratch. It involves incorporating new data samples into the existing model while preserving previously learned knowledge.

3. Concept Drift Detection: Concept drift refers to the situation where the underlying patterns or relationships in the data change over time. Adaptive pattern classification systems need to be able to detect concept drift to recognize when the current model's performance may be deteriorating due to changes in the data. Various drift detection methods, such as statistical tests or change detection algorithms, can be used to monitor and detect concept drift.

4. Ensemble Methods: Ensemble methods combine the predictions of multiple individual classifiers or models to improve classification performance. In adaptive pattern classification, ensemble methods can be used to create a diverse set of models that adapt and specialize in different subsets of the data or in different time periods. The ensemble can then dynamically adjust the combination or weighting of individual models based on the current data characteristics.

5. Transfer Learning: Transfer learning involves leveraging knowledge or models learned from one task or domain to improve performance on a related but different task or domain. In adaptive pattern classification, transfer learning techniques can be employed to transfer knowledge from a well-established model to a new model and adapt it to changing patterns in the data.

6. Reinforcement Learning: Reinforcement learning is a learning paradigm where an agent learns to make decisions or take actions based on feedback from its environment. In the context of adaptive pattern classification, reinforcement learning techniques can be used to adapt the classification rules or decision boundaries based on feedback or rewards received from the environment.

Adaptive pattern classification allows classification models to adapt to changing patterns and improve their performance over time. It is particularly useful in domains where data distributions may change, or where the classification system needs to continuously learn and adapt to new information.

Adaptive pattern classification in competitive learning neural network

Adaptive pattern classification can be achieved within a competitive learning neural network by integrating mechanisms that allow the network to adapt its decision boundaries or clustering patterns based on the incoming data. Here's how adaptive pattern classification can be implemented in a competitive learning neural network:

1. Competitive Learning Mechanism: The competitive learning mechanism is the core of a competitive learning neural network. It involves the competition among neurons to select a winner based on their similarity to the input data. The winning neuron represents a cluster or class to which the input data belongs.

2. Weight Update Rule: The weight update rule determines how the weights of the winning neuron and its neighbors are adjusted based on the input data. In adaptive pattern classification, the weight update rule can be modified to allow the network to adapt to changing patterns in the data. For example, the learning rate can be dynamically adjusted based on the rate of change or concept drift in the data.

3. Dynamic Neighborhood Function: The neighborhood function defines the spatial extent or influence of neighboring neurons in the weight update process. In adaptive pattern classification, the neighborhood function can be made dynamic, allowing it to adapt based on the local density or distribution of the data. This enables the network to adjust the clustering boundaries or decision boundaries as the data changes.

4. Concept Drift Detection: To adapt to changing patterns in the data, a competitive learning neural network can incorporate concept drift detection mechanisms. These mechanisms monitor the data for concept drift or changes in the underlying patterns and trigger adaptation processes in the network when such changes are detected. This ensures that the network can update its decision boundaries or clustering patterns to reflect the evolving data.

5. Online Learning and Incremental Updates: Adaptive pattern classification in competitive learning networks often involves online learning, where the network learns and updates its weights with each new data point. By continuously integrating new data and performing incremental updates, the network can adapt its clustering or classification boundaries to changing patterns in the data.

6. Ensemble of Networks: An ensemble of competitive learning networks can be used for adaptive pattern classification. Each network in the ensemble may specialize in different subsets of the data or time periods, and their outputs can be combined dynamically based on the current data characteristics. The ensemble can adaptively adjust the contribution or weighting of individual networks to improve classification accuracy in evolving data environments.

By incorporating these mechanisms, a competitive learning neural network can adaptively classify patterns and dynamically adjust its decision boundaries or clustering patterns based on changing data characteristics. This adaptability allows the network to maintain high accuracy and performance in the presence of concept drift or evolving patterns in the data.

Adaptive Resonance Theory Model (ART)

This network was developed by Stephen Grossberg and Gail Carpenter in 1987. It is based on competition and uses unsupervised learning model. Adaptive Resonance Theory ART networks, as the name suggests, is always open to new learning adaptive without losing the old patterns resonance.

Basically, ART network is a vector classifier which accepts an input vector and classifies it into one of the categories depending upon which of the stored pattern it resembles the most.

Operating Principle

The main operation of ART classification can be divided into the following phases −

1. **Recognition phase –** The input vector is compared with the classification presented at every node in the output layer. The output of the neuron becomes “1” if it best matches with the classification applied, otherwise it becomes “0”.
2. **Comparison phase –** In this phase, a comparison of the input vector to the comparison layer vector is done. The condition for reset is that the degree of similarity would be less than vigilance parameter.
3. **Search phase –** In this phase, the network will search for reset as well as the match done in the above phases. Hence, if there would be no reset and the match is quite good, then the classification is over. Otherwise, the process would be repeated and the other stored pattern must be sent to find the correct match.

ART1

It is a type of ART, which is designed to cluster binary vectors. We can understand about this with the architecture of it.

Architecture of ART1 (Fig.1)

It consists of the following two units –

1.Computational Unit − It is made up of the following −

Input unit (F1 layer) – It further has the following two portions −

F1a layer Input portion

− In ART1, there would be no processing in this portion rather than having the input vectors only. It is connected to F1b layer interface portion.

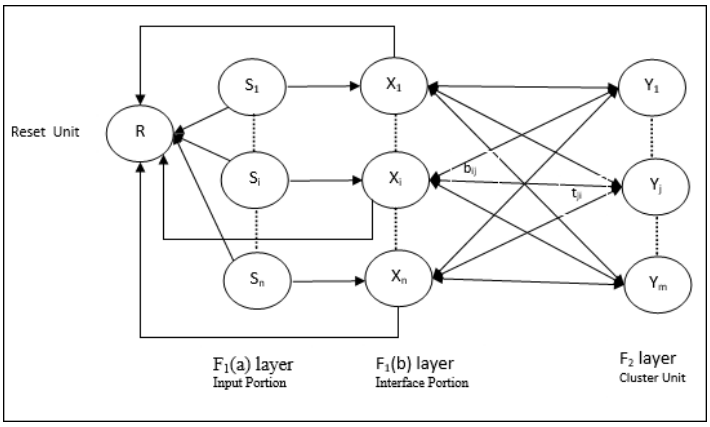
F1b layer Interface portion

− This portion combines the signal from the input portion with that of F2 layer. F1b layer is connected to F2 layer through bottom-up weights bij and F2 layer is connected to F1b layer through top-down weights tj

2.Cluster Unit (F2 layer) – This is a competitive layer. The unit having the largest net input is selected to learn the input pattern. The activation of all other cluster unit is set to 0.

3.Reset Mechanism – The work of this mechanism is based upon the similarity between the top-down weight and the input vector. Now, if the degree of this similarity is less than the vigilance parameter, then the cluster is not allowed to learn the pattern and a rest would happen.

4.Supplement Unit – Actually the issue with Reset mechanism is that the layer F2 must have to be inhibited under certain conditions and must also be available when some learning happens. That is why two supplemental units namely, G1 and G2 is added along with reset unit, R. They are called gain control units. These units receive and send signals to the other units present in the network. '+' indicates an excitatory signal, while '–' indicates an inhibitory signal.



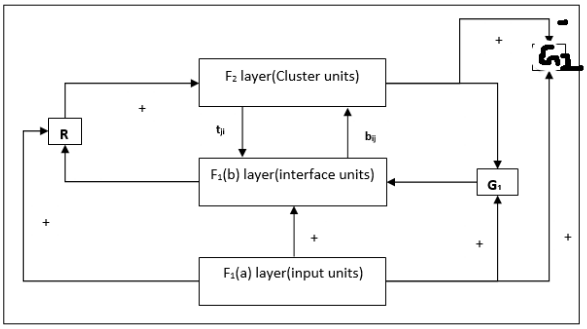


Fig.1.Architecture of ART1

Parameters Used

Following parameters are used −

n − Number of components in the input vector

m − Maximum number of clusters that can be formed

bij − Weight from F1b to F2 layer, i.e. bottom-up weights

tji − Weight from F2 to F1b layer, i.e. top-down weights

ρ − Vigilance parameter

||x|| − Norm of vector x

ART Algorithm

Step 1 − Initialize the learning rate, the vigilance parameter, and the weights as follows –

$$\alpha > 1 \ and \ 0 < \rho \leq 1$$

$$0 < b_{ij}(0) < \frac{\alpha}{\alpha - 1 + n} \ and \ t_{ij}(0) = 1$$

Step 2 − Continue step 3-9, when the stopping condition is not true.

Step 3 − Continue step 4-6 for every training input.

Step 4 − Set activations of all F1a and F1 units as follows F2 = 0 and F1a = input vectors

Step 5 − Input signal from F1a to F1b layer must be sent like

si=xi

Step 6 − For every inhibited F2 node

$$y_j = \sum_i b_{ij} x_i$$ the condition is yj ≠ -1

Step 7 − Perform step 8-10, when the reset is true.

Step 8 − Find J for $y_J \geq y_j$ for all nodes j

Step 9 − Again calculate the activation on F1b as follows

xi=si*tJi

Step 10 − Now, after calculating the norm of vector x and vector s, we need to check the reset condition as follows −

If ||x||/ ||s||< vigilance parameter ρ, then inhibit node J and go to step 7.

Else If ||x||/ ||s|| ≥ vigilance parameter ρ, then proceed further.

Step 11 − Weight updating for node J can be done as follows –

$$b_{ij}(new) = \frac{\alpha x_i}{\alpha - 1 + ||x||}$$

$$t_{ij}(new) = x_i$$

Step 12 − The stopping condition for algorithm must be checked and it may be as follows −

1. Do not have any change in weight.
2. Reset is not performed for units.
3. Maximum number of epochs reached.

Features of ART model

Adaptive Resonance Theory (ART) is a neural network model designed to perform unsupervised learning and pattern recognition tasks. It was developed by Stephen Grossberg and Gail Carpenter in the 1980s. ART models exhibit several key features that make them suitable for adaptive pattern recognition:

1. Vigilance Parameter: ART models incorporate a vigilance parameter that controls the level of specificity or selectivity in pattern recognition. The vigilance parameter determines the minimum similarity between an input pattern and the prototype patterns stored in the network. It allows ART models to adaptively adjust their recognition boundaries, ensuring that only sufficiently similar patterns are recognized.

2. Competitive Learning: ART models utilize competitive learning, where neurons in the network compete to become active based on their similarity to the input pattern. The winning neuron is selected as the best match for the input pattern, and the corresponding prototype pattern is updated or created if necessary. Competitive learning enables the network to adapt to new patterns and update its knowledge dynamically.

3. Bottom-Up and Top-Down Processing: ART models employ both bottom-up and top-down processing. Bottom-up processing involves the flow of information from input neurons to higher-level neurons, while top-down processing involves feedback signals from higher-level neurons to lower-level neurons. This bidirectional flow of information enables ART models to refine and revise their recognition boundaries based on both local and global context.

4. Resonance and Reset: ART models exhibit the property of resonance, where the winning neuron becomes active and resonates with the input pattern. Resonance allows the network to reinforce the recognition of similar patterns and inhibit the response to dissimilar patterns. In addition, ART models include a reset mechanism that allows the network to reset its state and adapt to new or unexpected patterns.

5. Stable and Plastic Prototypes: ART models maintain stable prototype patterns that represent learned categories. These prototype patterns are updated through plasticity mechanisms when a new pattern is presented. The stability of the prototype patterns ensures robust recognition, while their plasticity allows the network to adapt and refine its knowledge as new patterns are encountered.

6. Adaptive Learning and Recognition: ART models are adaptive and capable of learning and recognizing patterns in an incremental and adaptive manner. They can continuously update their internal representations and recognition boundaries based on new input patterns. This adaptability allows ART models to handle concept drift, accommodate new patterns, and maintain stability in the face of changing input statistics.

The features of ART models make them well-suited for tasks requiring real-time, online learning, and robust pattern recognition. They have been successfully applied in various domains, such as image recognition, speech processing, and anomaly detection.

Types of ART NETWORK

There are several types of Adaptive Resonance Theory (ART) networks that have been developed to address different requirements and applications. Here are some commonly known types of ART networks:

1. ART1: ART1 is the original and most fundamental type of ART network. It is designed for binary input patterns and is used for clustering and pattern recognition tasks. ART1 networks employ a binary weight matrix and use a threshold parameter called the vigilance parameter to control the selectivity of pattern recognition.

2. ART2: ART2 networks are an extension of ART1 and are designed to handle continuous-valued input patterns. ART2 networks use a continuous weight matrix and incorporate a fuzzy matching algorithm to adaptively learn and recognize patterns. They are suitable for applications such as image and speech recognition.

3. Fuzzy ART: Fuzzy ART networks combine the advantages of ART networks and fuzzy logic. They can handle both binary and continuous-valued inputs and employ fuzzy logic to provide a more flexible and gradual decision-making process. Fuzzy ART networks are robust to noise and uncertainty in the input data.

4. ARTMAP: ARTMAP networks integrate ART networks with a supervised learning component. They are used for classification tasks where labeled training data is available. ARTMAP networks learn to map input patterns to output classes and can handle both binary and continuous-valued inputs. They are known for their ability to perform incremental and fast learning.

5. ART-EMAP: ART-EMAP networks combine the principles of ART and Expectation-Maximization (EM) algorithms. They are used for clustering tasks and can handle continuous-valued inputs. ART-EMAP networks employ the EM algorithm to refine the prototype patterns and learn the distribution of input patterns within each cluster.

6. ART-2A: ART-2A networks are an advanced version of ART2 networks that introduce a search mechanism to improve the efficiency of pattern recognition. They incorporate a search process to identify a potential winner among the neurons, reducing the computational complexity of the matching process.

7. ART-3: ART-3 networks are designed for unsupervised learning and recognition tasks with binary input patterns. They extend the capabilities of ART1 networks by incorporating an additional level of vigilance to control the clustering process and provide more flexibility in recognizing complex patterns.

These are some of the notable types of ART networks. Each type of ART network has its own specific characteristics and is suited for different applications and data types. The choice of ART network depends on the requirements of the task at hand, such as the nature of input data, the need for supervised or unsupervised learning, and the desired level of adaptability and robustness.

Character recognition using ART network

Adaptive Resonance Theory (ART) networks have been widely used for character recognition tasks due to their ability to adaptively learn and recognize patterns. Here's how an ART network can be used for character recognition:

1. Network Architecture: The ART network for character recognition typically consists of two layers: the input layer and the recognition layer. The input layer represents the pixel values of the character images, while the recognition layer contains the prototype patterns or categories that the network learns to recognize.

2. Preprocessing: Before presenting the character images to the ART network, some preprocessing steps may be applied. This can include resizing the images to a consistent size, converting them to grayscale, and normalizing the pixel values to a common scale.

3. Competitive Learning: The ART network uses competitive learning to determine the best matching prototype pattern for each input character image. The competitive learning process involves calculating the similarity between the input image and each prototype pattern in the recognition layer. The prototype pattern with the highest similarity, meeting a specified vigilance threshold, is selected as the winner.

4. Matching and Learning: If the winner's similarity exceeds the vigilance threshold, the input image is classified as belonging to the category represented by the winning prototype pattern. In this case, the weights associated with the winning prototype pattern are updated to better represent the input image. If the winner's similarity is below the vigilance threshold, a new category is created in the recognition layer, and the weights associated with this new prototype pattern are adjusted to match the input image.

5. Recognition and Classification: Once the ART network has learned the prototype patterns for various character classes, it can perform recognition and classification of new character images. The input image is presented to the network, and the winner is determined through the competitive learning process. The recognized character class is then determined by matching the winning prototype pattern with the corresponding class label.

6. Training and Adaptation: The ART network can be trained in an incremental and adaptive manner. It can be presented with a set of labeled character images during the training phase to learn the initial prototype patterns. During the recognition phase, the network can continue to adapt and update its prototype patterns based on new or unseen character images to improve recognition accuracy.

By using ART networks for character recognition, the network can dynamically adjust its recognition boundaries and adapt to variations in character images, such as different writing styles or variations in stroke thickness. This adaptability makes ART networks effective for handling the complexity and variability of character recognition tasks.

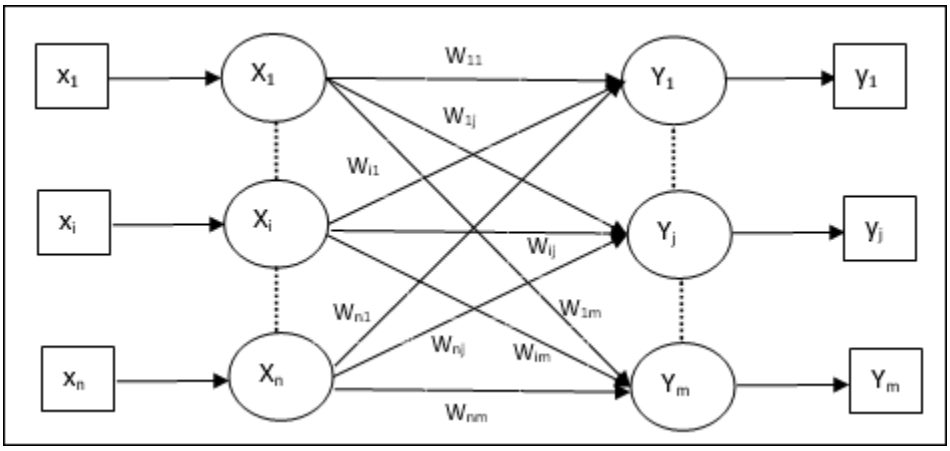
LEARNING VECTOR QUANTIZATION (LVQ)

Learning Vector Quantization (LVQ) is a type of supervised learning algorithm used for classification tasks. It is a variation of the popular Self-Organizing Map (SOM) algorithm. LVQ organizes data into prototypes or codebook vectors and uses a nearest-neighbor approach to classify new data points.

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 of LVQ

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 (x1,...,xi,...,xn)

T = class for training vector x

wj = weight vector for jth output unit

Cj = class associated with the jth output unit

Training Algorithm

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

Step 1a − 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 1b − Assign the initial weight and classification randomly.

Step 1c − Apply K-means clustering method.

Step 2 − Initialize reference vector α

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

$$D(j) = \sum_{i=1}^{n}\sum_{j=1}^{m}(x_i - w_{ij})^2$$

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 $w_j(new) = w_j(old) + \alpha[x - w_j(old)]$

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

Step 8 − Reduce the learning rate α .

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

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

Here's a step-by-step overview of the LVQ algorithm:

1. Initialization: Initialize a set of prototype vectors randomly or using a specific initialization strategy. Each prototype vector represents a class or category.

2. Training Data: Provide a labeled training dataset consisting of input patterns and their corresponding class labels.

3. Nearest-Neighbor Search: For each input pattern in the training dataset, compute the distance between the input pattern and each prototype vector using a distance metric such as Euclidean distance.

4. Winner Selection: Select the prototype vector with the smallest distance to the input pattern. This prototype vector is considered the winner or the best match for the input pattern.

5. Update Prototypes: Adjust the selected prototype vector based on the class label of the input pattern. If the class label matches the prototype's class, the prototype vector is moved closer to the input pattern. If the class label differs, the prototype vector is moved away from the input pattern. The adjustment is typically done by moving the prototype vector a small step towards or away from the input pattern.

6. Learning Rate: The adjustment step is controlled by a learning rate or step size parameter. The learning rate determines the magnitude of the prototype vector update.

7. Repeat: Repeat steps 3 to 6 for each input pattern in the training dataset. The process iteratively updates the prototype vectors based on the training data.

8. Classification: To classify a new, unlabeled input pattern, compute its distance to each prototype vector and assign it to the class associated with the closest prototype vector.

9. Evaluation: Assess the performance of the LVQ model using evaluation metrics such as accuracy, precision, recall, or F1 score.

10. Optimization: Adjust the parameters of the LVQ algorithm, such as the number of prototype vectors, learning rate, or initialization strategy, to optimize the model's performance on the training data.

The Learning Vector Quantization algorithm iteratively adjusts prototype vectors to improve classification accuracy. By organizing data into codebook vectors, LVQ provides an efficient and effective approach for supervised classification tasks.

Flowchart OF LVQ