Module -3

Self-Organizing Maps (SOM)

- Self-Organizing Maps (SOM), also known as Kohonen Maps, are a type of artificial neural network used for unsupervised learning. They help in visualizing and clustering high-dimensional data by mapping it onto a lower-dimensional (typically 2D) grid.
- It’s called "self-organizing" because it learns to group similar data points together without explicit instructions.

Purpose of SOM

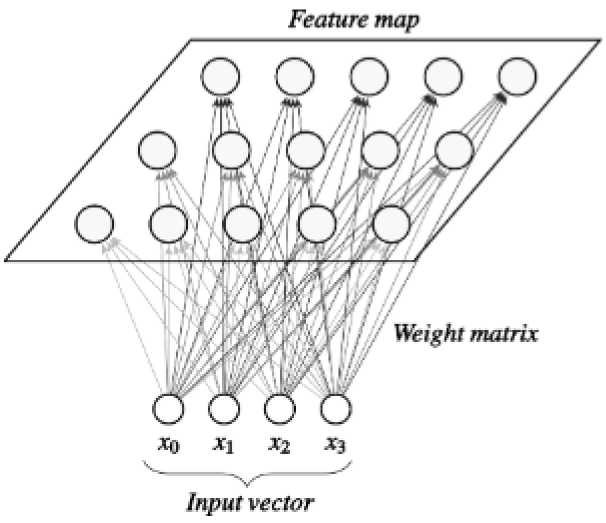
1. **Dimensionality Reduction**: Converts high-dimensional data into a 2D or 3D map for easy visualization.
2. **Clustering**: Groups similar data points together based on feature similarity.
3. **Feature Extraction**: Identifies essential patterns in data without requiring labeled examples.
4. **Data Visualization**: Projects complex data distributions into an interpretable space.
5. **Pattern Recognition**: Used in applications like speech recognition, medical diagnosis, and financial analysis.

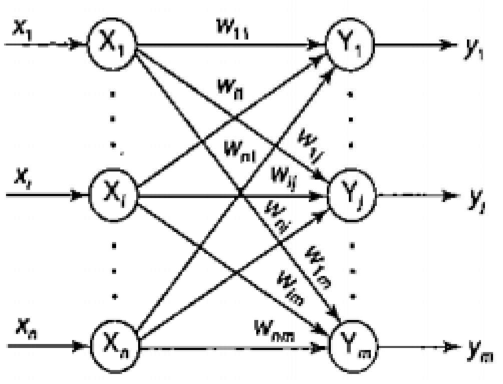
Architecture of SOM

A Self-Organizing Map consists of:

1. **Input Layer**: A set of neurons, each representing a feature in the input vector.
2. **Competitive Layer (Grid of Neurons)**: A 2D (or sometimes 3D) grid where each neuron (node) represents a weight vector.
3. **Connections**: Each neuron in the grid is connected to all input neurons.

Representation of SOM Architecture




x_1 X_1 w_{11} Y_1 y_1
w_{i1} w_{n1} w_{1j} w_{ij} w_{nj} w_{1m} w_{im} w_{nm}
x_i X_i Y_j y_j
x_n X_n Y_m y_m

General Architecture of SOM

Training Procedure of SOM

The training of a Self-Organizing Map follows an unsupervised learning approach. It mainly involves:

1. Initialization

- Each neuron in the SOM grid is assigned a random weight vector of the same dimension as the input data.

2. Competition Phase

- For each input vector, compute the **Best Matching Unit (BMU)** by calculating the squared Euclidean distance between the input vector and all neuron weight vectors:
- $d_i = \sum(x_j - w_{ij})^2$ where:
  - x_j is the input feature,
  - w_{ij} is the weight of the neuron,
  - d_i is the Euclidean distance of neuron ii from the input.
- The neuron with the **minimum distance** is chosen as the BMU.

3. Adaptation (Weight Update)

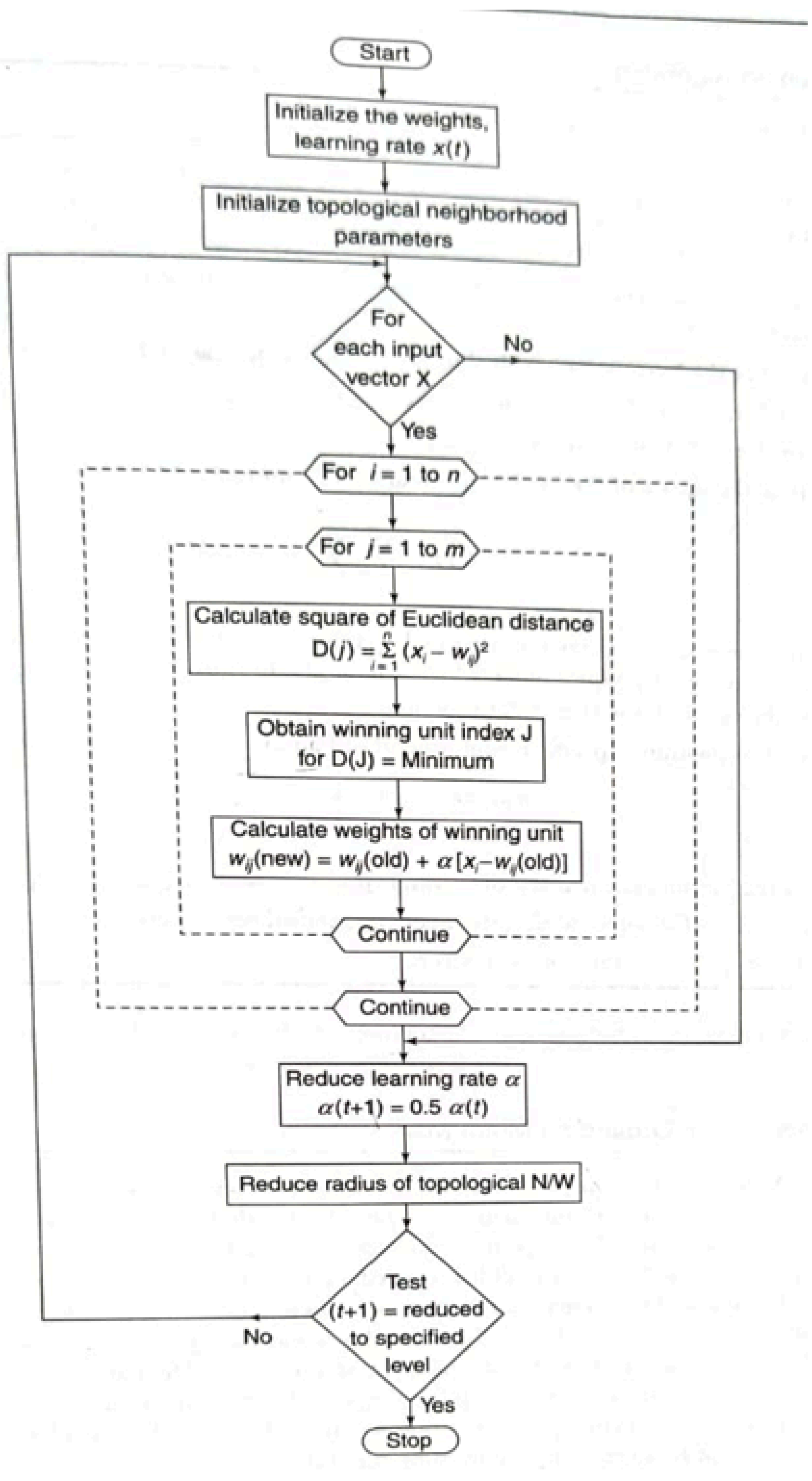
- The BMU and its neighbors adjust their weights using the update rule:
  $w_i(t+1) = w_i(t) + \theta(t) \cdot \alpha(t) \cdot (x - w_i(t))$ where:
    - $w_i(t)$ is the weight at time t,
    - $\alpha(t)$ is the learning rate (decreasing over time),
    - $\theta(t)$ is the neighborhood function (Gaussian or exponential decay) that determines how much neighboring neurons update.

4. Neighborhood Function

- The neighborhood function ensures that neurons close to the BMU are also updated. A common function is the Gaussian:
- $\theta(t) = e^{-\frac{d^2}{2\sigma^2(t)}}$
- where:
    - d is the distance from BMU,
    - $\sigma(t)$ is the neighborhood radius (decreasing over time).

5. Convergence

- Repeat steps 2–4 for multiple iterations.
- Over time, the map stabilizes, with neurons representing clusters in the input space.


Start
Initialize the weights, learning rate $x(t)$
Initialize topological neighborhood parameters
For each input vector X
No
Yes
For $i = 1$ to n
For $j = 1$ to m
Calculate square of Euclidean distance $D(j) = \sum_{i=1}^{n} (x_i - w_{ij})^2$
Obtain winning unit index J for D(J) = Minimum
Calculate weights of winning unit $w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha[x_i - w_{ij}(\text{old})]$
Continue
Continue
Reduce learning rate α $\alpha(t+1) = 0.5\,\alpha(t)$
Reduce radius of topological N/W
Test $(t+1)$ = reduced to specified level
No
Yes
Stop

Summary

- **SOM is an unsupervised neural network** that maps high-dimensional data onto a lower-dimensional grid.
- **Uses competitive learning**, where neurons compete to represent input vectors.
- **Trained iteratively using the BMU and weight adaptation** based on a decaying learning rate and neighborhood function.

- **Commonly used for clustering, visualization, and pattern recognition** in various domains.

Adaptive Resonance Theory (ART)

Adaptive Resonance Theory (ART) is a class of unsupervised machine learning algorithms primarily used for pattern recognition and clustering.

- Developed by Stephen Grossberg and Gail Carpenter, ART models are designed to address the stability-plasticity dilemma, ensuring that the network remains stable (retaining previously learned information) while also being plastic (able to learn new patterns).
- The core motivation behind ART is to create a biologically plausible model of learning that mimics how the human brain processes information.
- ART algorithms allow for real-time, incremental learning without catastrophic forgetting, making them suitable for dynamic environments where data may change over time.

Key Features of ART

1. **Stability-Plasticity Balance**: ART ensures that the system can adapt to new patterns without forgetting previously learned ones.
2. **Unsupervised Learning**: ART operates without labeled data, grouping similar patterns into clusters.
3. **Real-Time Learning**: It processes data sequentially and incrementally, updating its internal representation as new data arrives.
4. **Vigilance Parameter**: A key concept in ART, vigilance determines the level of similarity required for a pattern to be classified into an existing cluster or form a new one.

Architecture of ART

ART networks typically consist of two main layers:

1. **Comparison Layer (F1 Layer)**:
   - Receives input patterns and broadcasts them to the **Recognition Layer**.
   - Performs normalization of input vectors to ensure comparability.
2. **Recognition Layer (F2 Layer)**:
   - Contains neurons that represent categories or clusters.

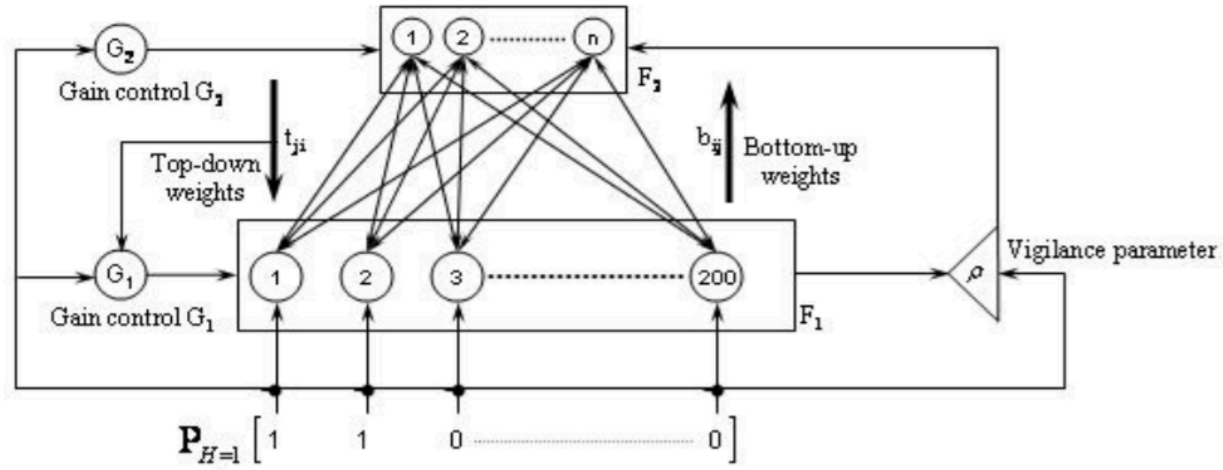
    - Each neuron in this layer has a weight vector associated with it, which is updated during learning.

3. **Control Mechanism**:

- Controls degree of similarity of patterns placed on the same cluster

Connections:

- **Bottom-Up Connections**: Transfer input patterns from the Comparison Layer to the Recognition Layer.
- **Top-Down Connections**: Feedback connections from the Recognition Layer to the Comparison Layer, used to verify resonance (matching between input and category).
-



Types of ART: ART 1 and ART 2

Adaptive Resonance Theory (ART) has several variants, with ART 1 and ART 2 being the most prominent.

ART 1

- **Purpose**: Designed for **binary input data**.
- **Input Representation**: Input vectors consist of binary values (0s and 1s).
- **Weight Initialization**:
    - Bottom-up weights (b_{ij}) are initialized to small random values, typically ($b_{ij} = \frac{1}{1+n}$), where (n) is the number of input features.
    - Top-down weights (t_{ij}) are initialized to 1.
- **Vigilance Parameter**:
    - A key parameter (ρ) controls the similarity threshold for clustering. Higher (ρ) leads to finer clusters.

- **Training Process**:
  - Input patterns are presented sequentially.
  - The neuron with the highest activation (BMU) is selected.
  - If the similarity between the input and the BMU's weights exceeds (ρ), weights are updated; otherwise, a new neuron is created.
- **Weight Update**:
  - Bidirectional weight updates: both bottom-up and top-down weights are adjusted based on the input pattern.
- **Applications**:
  - Used in binary pattern recognition tasks like document clustering and binary image classification.
- **Advantages**:
  - Simple and efficient for binary data.
  - Avoids catastrophic forgetting.
- **Limitations**:
  - Limited to binary inputs; cannot handle continuous or analog data.

ART 2

- **Purpose**: Designed for **continuous or analog input data**.
- **Input Representation**: Input vectors can contain real-valued features.
- **Normalization**:
  - Input vectors are normalized to handle variations in magnitude.
- **Weight Initialization**:
  - Weights are initialized randomly or with small values.
- **Vigilance Parameter**:
  - Similar to ART 1, (ρ) controls the clustering granularity.
- **Training Process**:
  - Input patterns are normalized and presented sequentially.
  - The BMU is selected based on activation.
  - Resonance occurs if the similarity exceeds (ρ); otherwise, a new neuron is created.
- **Weight Update**:
  - Both bottom-up and top-down weights are updated iteratively.
- **Applications**:
  - Used in continuous data tasks like speech recognition, sensor data analysis, and time-series clustering.
- **Advantages**:
  - Handles continuous data effectively.

    - Maintains stability-plasticity balance.
- **Limitations**:
    - More computationally complex than ART 1 due to normalization and continuous data processing.

Key Differences

Feature	ART 1	ART 2
Input Type	Binary data	Continuous/analog data
Normalization	Not required	Required
Complexity	Simpler	More complex
Applications	Binary pattern recognition	Continuous data tasks
Computation	Efficient for binary inputs	Requires more computational power

Both ART 1 and ART 2 are powerful tools for unsupervised learning, with ART 1 excelling in binary data tasks and ART 2 extending capabilities to continuous data. They share the core ART principles of stability-plasticity balance, real-time learning, and vigilance-controlled clustering.

Training Procedure of ART-1 (Binary Data)

1. Initialization

1. **Initialize Weights**:
    - Initialize the **bottom-up weights** (b_{ij}) and **top-down weights** (t_{ij}) to small random values.
    - Typically, $b_{ij} = \frac{1}{1+n}$, where (n) is the number of input features.
    - $t_{ij} = 1$ (initially, all top-down weights are set to 1).
2. **Set Vigilance Parameter (ρ)**:
    - Choose a vigilance parameter (ρ) (e.g., ($rho = 0.7$)). This determines the level of similarity required for a pattern to match a category.

---

2. Input Presentation

Present the input vectors sequentially to the network. For each input vector (x), perform the following steps.

---

3. Compute Bottom-Up Activation

3. For each neuron (j) in the Recognition Layer (F2), compute the bottom-up activation (T_j):

$$T_j = \sum_{i=1}^{n} x_i \cdot b_{ij}$$

   where:
   - (x_i) is the (i)-th feature of the input vector,
   - (b_{ij}) is the bottom-up weight from feature (i) to neuron (j).
4. Select the neuron with the highest activation (T_j) as the **winner**.

---

4. Check Vigilance

1. Compute the similarity between the input vector (x) and the top-down weight vector (t_j) of the winning neuron:

$$\text{Similarity} = \frac{\sum_{i=1}^{n} x_i \cdot t_{ij}}{\sum_{i=1}^{n} x_i}$$

2. If the similarity is greater than or equal to the vigilance parameter (ρ) (i.e., $(\text{Similarity} \geq \rho)$), the input **resonates** with the winning neuron. Proceed to update the weights.
3. If the similarity is less than (ρ), **reset** the winning neuron and repeat the process with the next highest activated neuron. If no neuron satisfies the vigilance condition, create a new neuron to represent the input.

---

5. Update Weights (Bidirectional Update)

If resonance occurs, update both the bottom-up and top-down weights of the winning neuron using the following rules:

4. **Update Top-Down Weights**:

$$t_{ij}(t+1) = x_i \cdot t_{ij}(t)$$

   This ensures that the top-down weights reflect the intersection of the input vector and the existing weights.

5. **Update Bottom-Up Weights**:

$$b_{ij}(t+1) = \frac{\alpha \cdot t_{ij}(t+1)}{(\alpha - 1) + ||t_{ij}(t+1)||}$$

   This normalizes the bottom-up weights to ensure stability.

---

6. Repeat for All Inputs

Repeat steps 2–5 for all input vectors ($x_1, x_2, x_3, \ldots x_n$). The network will learn to categorize the inputs based on their similarity.

Motivation Behind ART

The development of ART was motivated by the need to address several challenges in machine learning:

1. **Stability-Plasticity Dilemma**:
   - Traditional neural networks often face the issue of catastrophic forgetting, where learning new information causes the network to forget previously learned patterns. ART avoids this by only updating weights when resonance occurs.
2. **Biological Plausibility**:
   - ART models are inspired by the human brain's ability to learn incrementally and adapt to new information while retaining old knowledge.
3. **Real-Time Learning**:
   - ART is designed to handle streaming data, making it suitable for applications like online monitoring, anomaly detection, and adaptive control systems.
4. **Vigilance Control**:
   - The vigilance parameter provides a mechanism to control the granularity of clustering, allowing the user to decide how similar patterns need to be to belong to the same cluster.

Applications of ART

1. **Pattern Recognition**: Used in handwriting recognition, speech recognition, and image classification.
2. **Clustering**: ART is widely used for clustering high-dimensional data in fields like bioinformatics and finance.

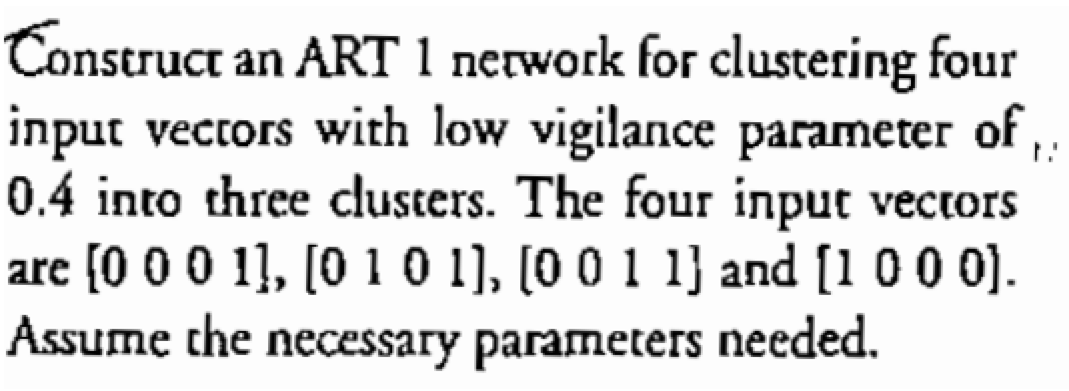
3. **Anomaly Detection**: Identifies outliers or anomalies in data streams.
4. **Robotics**: ART is applied in sensor fusion and adaptive control systems for robots.
5. **Medical Diagnosis**: Helps in categorizing medical data for disease diagnosis.

Comparison with SOM

Feature	Self-Organizing Maps (SOM)	Adaptive Resonance Theory (ART)
Learning Type	Unsupervised	Unsupervised
Stability-Plasticity	No explicit mechanism to prevent forgetting	Explicitly addresses the stability-plasticity dilemma
Clustering	Maps high-dimensional data onto a grid	Forms clusters based on vigilance
Real-Time Learning	Requires batch processing	Supports online, incremental learning
Vigilance Parameter	Not applicable	Central to the learning process

Adaptive Resonance Theory (ART) provides a powerful framework for unsupervised learning, particularly in scenarios where stability and adaptability are critical. Its ability to handle real-time data streams and avoid catastrophic forgetting makes it a valuable tool in dynamic environments. While SOM is excellent for visualization and dimensionality reduction, ART excels in tasks requiring incremental learning and fine-grained control over clustering.

Exercise Problem:



Construct an ART 1 network for clustering four input vectors with low vigilance parameter of 0.4 into three clusters. The four input vectors are [0 0 0 1], [0 1 0 1], [0 0 1 1] and [1 0 0 0]. Assume the necessary parameters needed.

Summary

- ART 1 uses a bidirectional weight update mechanism to ensure stability and plasticity.
- The vigilance parameter (ρ) controls the granularity of clustering.
- The network learns incrementally, adapting to new inputs without forgetting previous knowledge.

By following this procedure, ART 1 can effectively categorize binary input vectors like ($x_1, x_2, x_3 ... x_n$) into clusters based on their similarity.

Learning Vector Quantization (LVQ): Overview

Learning Vector Quantization (LVQ) is a supervised neural network algorithm used for classification tasks.

- It is inspired by the Self-Organizing Map (SOM) but differs in that it uses labeled data to train the network.
- LVQ is particularly effective for pattern classification and is widely used in applications like speech recognition, medical diagnosis, and image processing.

Motivation Behind LVQ

1. **Supervised Learning**: Unlike SOM, which is unsupervised, LVQ leverages labeled data to improve classification accuracy.
2. **Efficiency**: LVQ is computationally efficient and works well with high-dimensional data.
3. **Interpretability**: The resulting prototypes (weight vectors) are easy to interpret and can provide insights into the structure of the data.
4. **Adaptability**: LVQ can adapt to new data incrementally, making it suitable for dynamic environments.

Training Procedure of LVQ

The LVQ algorithm involves the following steps:

1. Initialization

1. **Initialize Prototypes**:

- Choose (k) prototypes (weight vectors) from the training data. Each prototype is associated with a class label.
- Alternatively, initialize the prototypes randomly or using a clustering algorithm like k-means.

2. Training Loop

For each training iteration:

1. **Select an Input Vector**:
   - Randomly select an input vector (x) from the training dataset with its corresponding class label (y).
2. **Find the Closest Prototype**:
   - Compute the Euclidean distance between (x) and each prototype:

     $$d_j = \sqrt{\sum_{i=1}^{n}(x_i - w_{ij})^2}$$

     where (w_{ij}) is the (i)-th component of the (j)-th prototype.
   - Identify the prototype (w_c) with the smallest distance to (x).
3. **Update the Prototype**:
   - If the class label of (w_c) matches the label of (x), move the prototype closer to (x):

     $$w_c(t+1) = w_c(t) + \alpha(t) \cdot (x - w_c(t))$$

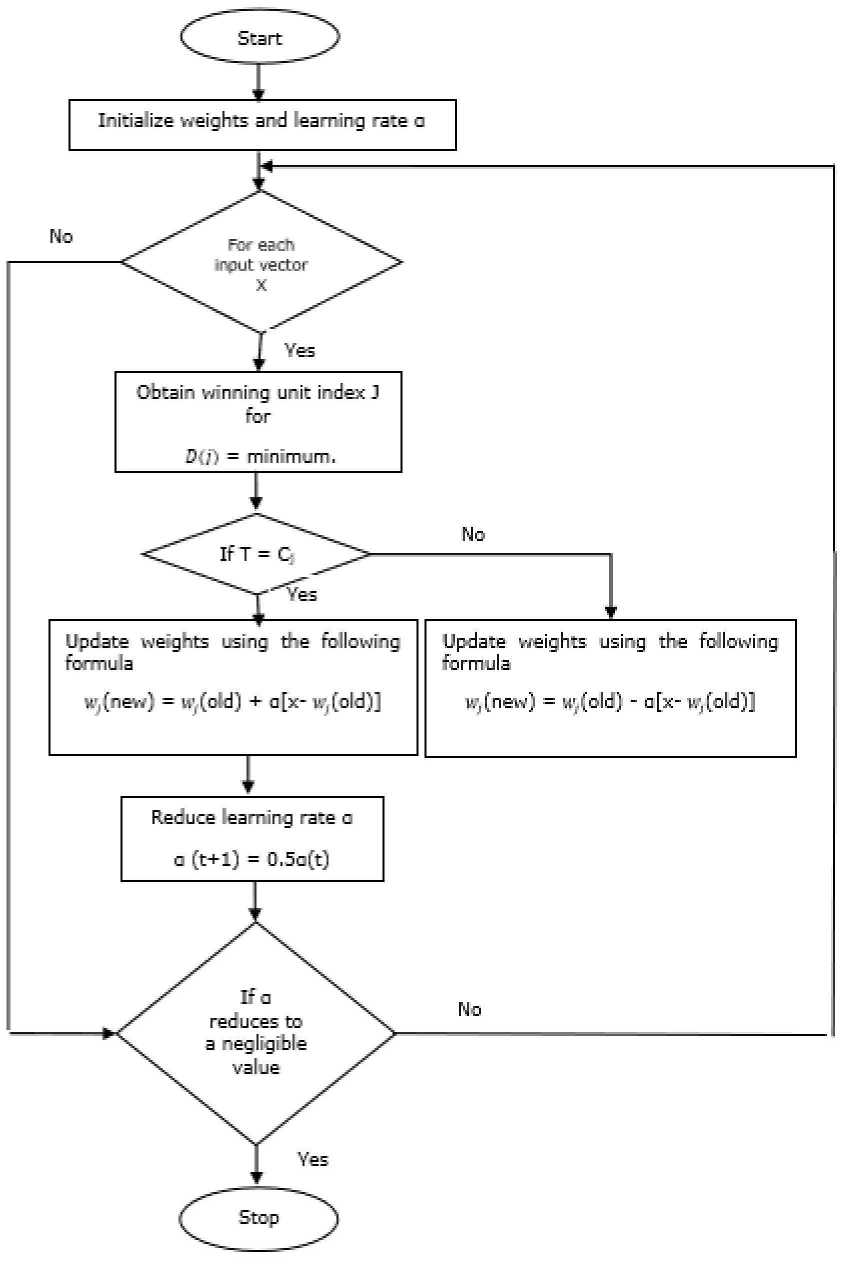
   - If the class label of (w_c) does not match the label of (x), move the prototype away from (x):

     $$w_c(t+1) = w_c(t) - \alpha(t) \cdot (x - w_c(t))$$

   - Here, ($\alpha(t)$) is the learning rate, which decreases over time.
4. **Repeat**:
   - Repeat steps 2–4 for a fixed number of iterations or until convergence.

3. Convergence

- The algorithm converges when the prototypes stabilize, i.e., they no longer change significantly with further iterations.



Start
Initialize weights and learning rate α
No
For each input vector X
Yes
Obtain winning unit index J for D(j) = minimum.
If T = C_j
No
Yes
Update weights using the following formula w_j(new) = w_j(old) + α[x- w_j(old)]
Update weights using the following formula w_j(new) = w_j(old) - α[x- w_j(old)]
Reduce learning rate α
α (t+1) = 0.5α(t)
If α reduces to a negligible value
No
Yes
Stop

Comparison to ART and Kohonen Nets

Feature	LVQ	ART	Kohonen Nets (SOM)
Learning Type	Supervised	Unsupervised	Unsupervised
Purpose	Classification	Clustering, Pattern Recognition	Clustering, Visualization
Data Requirement	Labeled data	Unlabeled data	Unlabeled data
Prototypes/Neurons	Prototypes with class labels	Neurons forming clusters	Neurons forming a topological map

Feature	LVQ	ART	Kohonen Nets (SOM)
Weight Update	Bidirectional (adjusts based on class labels)	Bidirectional (based on vigilance)	Neighborhood-based
Real-Time Learning	Yes	Yes	No
Stability-Plasticity	Not explicitly addressed	Explicitly addressed	Not explicitly addressed
Applications	Classification tasks	Pattern recognition, clustering	Data visualization, clustering

---

Key Differences

1. **Learning Paradigm**:
   - LVQ is supervised, while ART and SOM are unsupervised.
   - LVQ requires labeled data to train, whereas ART and SOM do not.
2. **Purpose**:
   - LVQ is primarily used for classification.
   - ART is focused on clustering and pattern recognition.
   - SOM is used for clustering and visualization of high-dimensional data.
3. **Weight Update Mechanism**:
   - LVQ updates weights based on class labels (move closer or farther from input).
   - ART updates weights based on the vigilance parameter and resonance.
   - SOM updates weights based on neighborhood functions.
4. **Real-Time Learning**:
   - LVQ and ART support real-time, incremental learning.
   - SOM typically requires batch processing.

Summary

- **LVQ** is a supervised learning algorithm used for classification, leveraging labeled data to adjust prototypes.
- **ART** is an unsupervised algorithm designed for clustering and pattern recognition, addressing the stability-plasticity dilemma.
- **SOM** is an unsupervised algorithm used for clustering and visualization, mapping high-dimensional data onto a lower-dimensional grid.

Each algorithm has its strengths and is suited to different types of tasks. LVQ is ideal for classification problems with labeled data, ART excels in dynamic environments requiring real-time learning, and SOM is best for visualizing and understanding the structure of high-dimensional data.