ARTIFICIAL NEURAL NETWORK

Subject Code: 21CSE326T

UNIT -4

Presented by:

Ms.Rupam kumari

Assistant Professor

Computer Science and Engineering Department

TOPICS COVERED

- Unsupervised neural networks
- Hebbian learning rule
- Principle component learning
- Learning vector quantizer
- Self organizing Maps
- Functionality training
- Topology Function
- Decreasing learning rate
- Variations of SOMS
- Neural Gas Multi-SOM
- Multi neural Gas

TOPICS COVERED

- Adaptive resonance theory
- Recurrent networks
- Orienting sub systems
- Learning laws

Unsupervised Neural Networks

- Unsupervised neural networks are a class of artificial neural networks that are trained on data without explicit supervision or labeled outputs. Unlike supervised learning where the network learns from labeled examples (input-output pairs), unsupervised learning aims to find hidden structure or patterns within the input data itself.
- Types of unsupervised neural networks
- Autoencoders: These networks are designed to encode the input data into a lower-dimensional representation and then decode it back to the original input. They consist of an encoder network that compresses the input data into a latent representation and a decoder network that reconstructs the original input from the latent representation. Autoencoders can be used for tasks such as data denoising, dimensionality reduction, and feature learning.

Cont.

- Generative Adversarial Networks (GANs): GANs consist of two neural networks, a generator and a discriminator, which are trained simultaneously through a minimax game. The generator learns to generate realistic samples from random noise, while the discriminator learns to distinguish between real and generated samples. GANs are widely used for generating realistic synthetic data, image-to-image translation, and data augmentation.

- Self-Organizing Maps (SOMs): SOMs are a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional, discretized representation of the input space. They organize the input data based on similarity, preserving the topological properties of the input space. SOMs are often used for tasks such as clustering, visualization, and exploratory data analysis.

Cont.

- Deep Belief Networks (DBNs): DBNs are probabilistic generative models composed of multiple layers of stochastic, latent variables. They consist of a stack of restricted Boltzmann machines (RBMs), which are trained layer-wise using unsupervised learning. DBNs can be used for tasks such as feature learning, dimensionality reduction, and density estimation.

- These are just a few examples of unsupervised neural networks, and there are many other variations and architectures designed for specific tasks and applications. Unsupervised learning is particularly useful in scenarios where labeled data is scarce or expensive to obtain, as it can automatically discover meaningful patterns and structure in the data.

Hebbian learning rule

- The Hebbian learning rule is a principle in neuroscience and artificial neural networks that states "cells that fire together, wire together." Proposed by Donald Hebb in 1949, this rule suggests that when two connected neurons are activated simultaneously, the strength of the connection between them is increased.

- In simpler terms, if neuron A repeatedly fires just before neuron B, the connection between A and B strengthens, making it more likely that A's activity will trigger B's activity in the future. This principle is fundamental to understanding how neural networks learn and adapt to input patterns over time.

Cont.

- Hebbian learning has been influential in the development of various models of associative memory, unsupervised learning algorithms, and theories of synaptic plasticity—the ability of synapses to strengthen or weaken over time in response to activity. However, it's important to note that while Hebbian learning provides a basic framework, actual learning in biological systems and artificial neural networks involves more complex mechanisms and additional factors.
- Hebbian Learning Rule Algorithm :

- Set all weights to zero, wi = 0 for i=1 to n, and bias to zero.
- For each input vector, S(input vector) : t(target output pair), repeat steps 3-5.
- Set activations for input units with the input vector Xi = Si for i = 1 to n.
- Set the corresponding output value to the output neuron, i.e. y = t.
- Update weight and bias by applying Hebb rule for all i = 1 to n:

Cont.

$$w_i(\text{new}) = w_i(\text{old}) + x_i y$$

$$b(\text{new}) = b(\text{old}) + y$$

Principle component learning

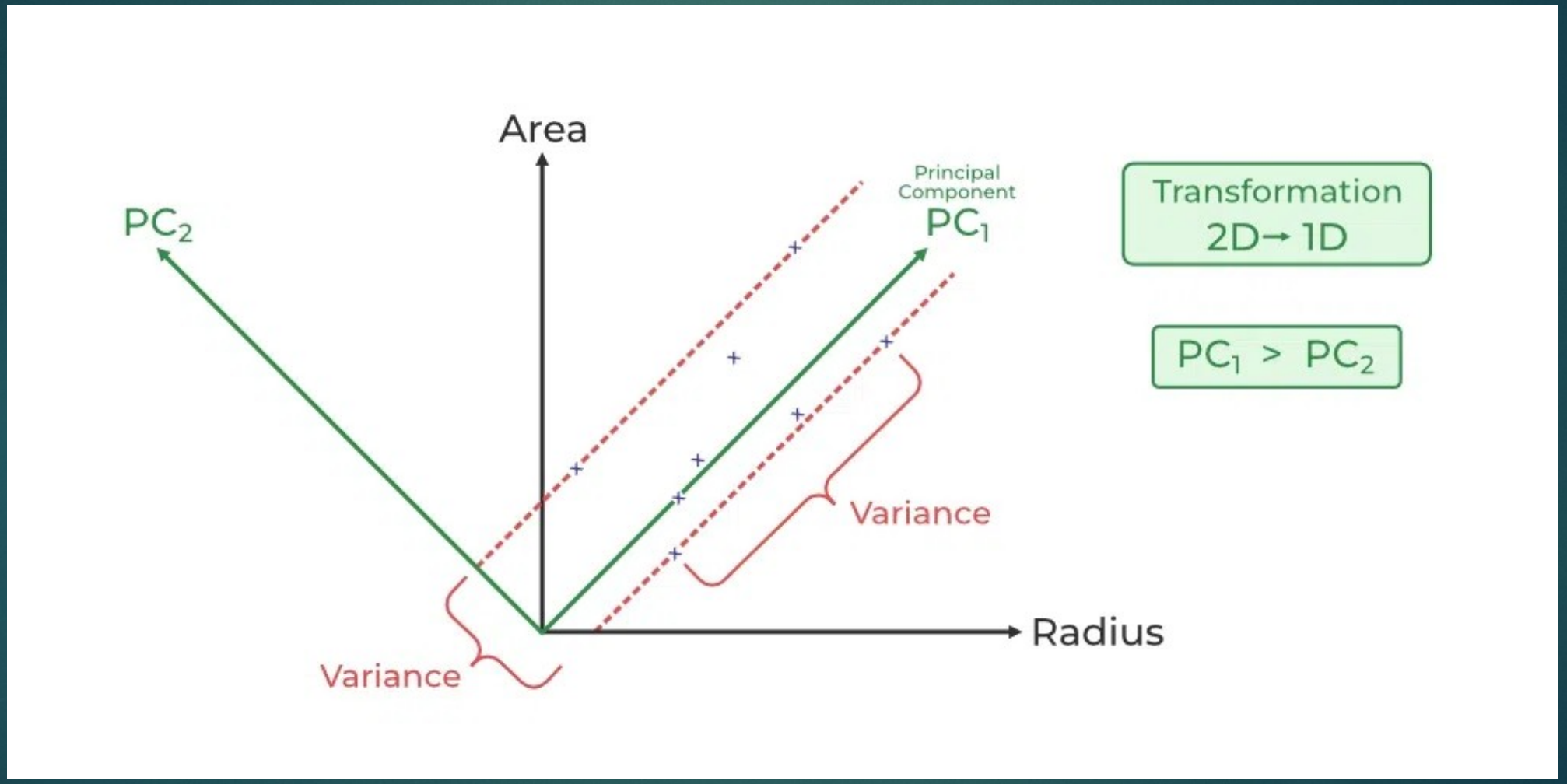
- Principal Component Analysis (PCA) rather than "Principle component learning." PCA is a dimensionality reduction technique commonly used in machine learning and data analysis to reduce the number of features in a dataset while preserving the most important information.

- However, if you're referring to a specific concept or technique within neural networks that you're calling "principle component learning," it's possible that it's a term or approach I'm not familiar with. Could you provide more context or clarify what you mean by "principle component learning" in the context of neural networks?
- Principal components analysis can also be implemented within a neural network. However, as this process is irreversible, the data's reduction should be done only for the inputs and not for the target variables. Principal component analysis allows us to reduce the size of a data set without much loss of information.

Cont.

- PCA is a dimensionality reduction technique that aims to reduce the number of input features while retaining as much of the original information as possible.
- It works by finding a new set of variables (called principal components) that capture the maximum variance in the data.
- The main goal of PCA is to reduce the dimensionality of a dataset while preserving important patterns or relationships between variables.
- PCA is widely used in exploratory data analysis and machine learning for predictive models.
- Principal Component Analysis (PCA) is used to reduce the dimensionality of a data set by finding a new set of variables, smaller than the original set of variables, retaining most of the sample's information, and useful for the regression and classification of data.

Cont.


Area
Principal Component
PC_1
PC_2
Transformation
2D→ 1D
$PC_1 > PC_2$
Variance
Variance
Radius

Learning vector quantizer

- A learning vector quantizer (LVQ) is a type of artificial neural network used in machine learning for classification tasks. LVQ is a supervised learning algorithm that falls under the category of prototype-based classifiers.

- In LVQ, the network learns to classify input patterns by comparing them to a set of prototype vectors, also known as codebook vectors or centroids. During the training process, these prototype vectors are adjusted to better represent the input patterns in the feature space.

- Here's a simplified overview of how LVQ works:

- Initialization: Initialize the prototype vectors randomly or based on some heuristic.

- Training: For each training example, compute the distance between the input pattern and each prototype vector. The prototype vector that is closest to the input pattern (according to some distance metric, often Euclidean distance) is identified.

Cont.

- Update: If the identified prototype vector belongs to the same class as the input pattern, it is moved closer to the input pattern. If it belongs to a different class, it is moved away from the input pattern. The amount of adjustment is determined by a learning rate and possibly a neighborhood function, which determines how much influence neighboring prototype vectors have on the update.

- Iterate: Repeat steps 2 and 3 for a fixed number of iterations or until convergence criteria are met.

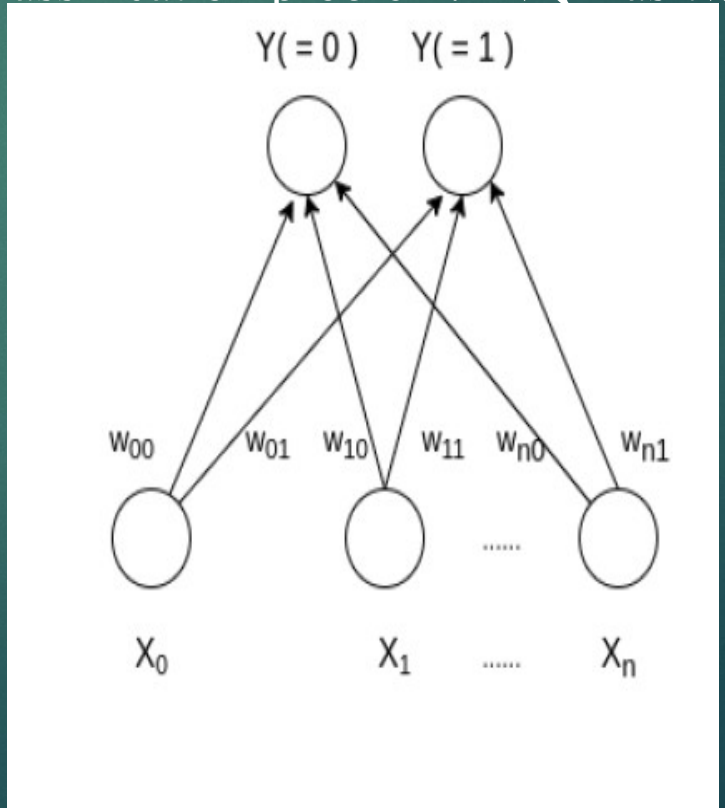
- Classification: Once the prototype vectors are trained, classification of new input patterns is performed by assigning them to the class associated with the nearest prototype vector.

- LVQ can be seen as a hybrid between k-nearest neighbors (KNN) and artificial neural networks. It maintains prototypes similar to KNN but updates them based on a learning rule akin to neural networks.

- Different variants of LVQ exist, such as LVQ1, LVQ2, and LVQ3, each with its own learning rules and update strategies. LVQ has been used in various applications, including pattern recognition, image classification, and data compression.

Cont

- Learning Vector Quantization (or LVQ) is a type of Artificial Neural Network which also inspired by biological models of neural systems. It is based on prototype supervised learning classification algorithm and trained its network through a competitive learning algorithm similar to Self Organizing Map. It can also deal with the multiclass classification problem. LVQ has two layers, one is the Input layer and the other one is the Output layer



Learning Vector Quantization works

- Step 1: Initialize reference vectors.
- from a given set of training vectors, take the first “n” number of clusters training vectors and use them as weight vectors, the remaining vectors can be used for training.
- Assign initial weights and classifications randomly
- Step 2: Calculate Euclidean distance for i=1 to n and j=1 to m,
- $D(j) = \Sigma\Sigma (xi-Wij)^2$
- find winning unit index j, where D(j) is minimum

Cont.

- Step 3: Update weights on the winning unit wi using the following conditions:
- if $T = J$ then $wi(new) = wi(old) + \alpha[x - wi(old)]$
- if $T \neq J$ then $wi(new) = wi(old) - \alpha[x - wi(old)]$
- Step 4: Check for the stopping condition if false repeat the above steps.

Self-Organizing Maps (SOMs)

- Self-Organizing Maps (SOMs), also known as Kohonen maps, are a type of artificial neural network used for dimensionality reduction and visualization of high-dimensional data. They were introduced by Finnish professor Teuvo Kohonen in the 1980s.

- SOMs are unsupervised learning algorithms, meaning they learn to represent the structure of the input data without requiring explicit labels. The main idea behind SOMs is to map high-dimensional input space onto a low-dimensional grid of neurons, typically organized in a two-dimensional lattice. Each neuron in the grid is associated with a weight vector of the same dimensionality as the input data.

- Here's a high-level overview of how SOMs work:

- Initialization: Initialize the weight vectors of neurons randomly or using some initialization method.

Cont.

- Competition: For each input data point, find the neuron whose weight vector is most similar to the input data point. This is typically done by computing the Euclidean distance between the input vector and the weight vectors of all neurons and selecting the neuron with the closest weight vector. This neuron is called the "best matching unit" (BMU).

- Cooperation: Update the weights of the BMU and its neighboring neurons to make them more similar to the input data point. The extent of the update decreases with distance from the BMU, following a neighborhood function.

- Iteration: Repeat steps 2 and 3 for all input data points for a fixed number of iterations or until convergence criteria are met. As the algorithm progresses, the SOM gradually organizes itself to represent the structure of the input data.

Cont.

- Visualization: After training, the SOM can be visualized by representing each neuron in the grid as a point in a lower-dimensional space, such as a 2D or 3D scatter plot. This visualization helps to reveal the underlying structure and relationships within the high-dimensional data.

- SOMs are widely used for tasks such as clustering, visualization, and data exploration. They have applications in various fields, including data mining, pattern recognition, and feature extraction.

Neural Gas

- Neural Gas is a type of neural network algorithm that is often used for clustering and data visualization tasks. It was introduced by Thomas Martinetz and Klaus Schulten in 1991 as a competitive learning algorithm similar to Kohonen's Self-Organizing Maps (SOM). The Neural Gas algorithm is particularly useful when the underlying structure of the data is not well-known beforehand.
- How it works:

- Initialization: Initialize a set of neurons randomly or using some heuristic method. Each neuron represents a prototype or a cluster center.

- Input Presentation: Present input data to the network.

Cont.

- Competition: Compute the distance between the input data and each neuron. Neurons compete to be the best match for the input data. Usually, the neuron with the smallest distance (i.e., the closest prototype) wins.
- Update: Update the winning neuron (and sometimes its neighbors) to better represent the input data. Typically, this involves moving the winning neuron closer to the input data.
- Adaptation: Adjust the learning rate and neighborhood function parameters. This step ensures that the network gradually refines its representation of the data over time.
- Repeat: Steps 2-5 are repeated for multiple iterations or until convergence.

Growing Neural Gas

- Growing Neural Gas (GNG) is an extension of the Neural Gas algorithm designed to dynamically adapt its structure to better represent the input data. It was proposed by Bernd Fritzke in 1995 as a method for incremental, self-organizing network growth.
- Here's how Growing Neural Gas typically works:
- Initialization: Initialize a set of neurons with random positions in the input space.
- Input Presentation: Present input data to the network.

Cont.

1. Competition and Adaptation: Similar to Neural Gas, neurons compete to be the best match for the input data. The winning neuron and its neighboring neurons are updated to better represent the input data. Additionally, the age of the connections between winning neurons and their neighbors is increased.
2. Neuron Insertion: Periodically, based on some criteria (e.g., network error, age of neurons), new neurons are inserted into the network. These neurons are placed at positions where the network has the largest error or where data density is high. The connections to neighboring neurons are adjusted accordingly.
3. Neuron Removal: Also periodically, neurons that are not highly activated or have become redundant (e.g., due to changes in data distribution) are removed from the network. This helps to keep the network size manageable and prevents overfitting.

Cont.

Repeat: Steps 2-5 are repeated for multiple iterations or until convergence.

Growing Neural Gas is advantageous because it can automatically adjust its structure to capture the underlying complexity of the data. By growing and pruning neurons as needed, it can efficiently adapt to changes in the data distribution and handle high-dimensional input spaces.

GNG has found applications in various fields, including data clustering, visualization, and feature extraction. It's particularly useful in scenarios where the data distribution is not known a priori or may change over time.

Multi Neural Gas

- "Multi-Neural Gas" isn't a specific, widely recognized term in neural network literature as of my last update. However, it's possible to speculate on what it might entail based on existing concepts.

- One interpretation could be an extension of Neural Gas or Growing Neural Gas to handle multiple modalities or sources of data. In this context, the network would learn representations from diverse data types simultaneously, perhaps merging them in some way to create a unified representation space. This could be useful in scenarios where you have data coming from different sensors or sources and you want to capture the relationships between them.

Cont.

- Another interpretation could be applying Neural Gas or Growing Neural Gas separately to different subsets of the data or different layers of a larger neural network. This approach might be used in a hierarchical or modular architecture where each "neural gas" instance focuses on learning specific aspects of the data or solving different subproblems.

- Without a specific definition or context, it's challenging to provide a precise explanation of "Multi-Neural Gas," but these are some directions where the concept could potentially lead. If you have a specific context or definition in mind, I'd be happy to provide further insights!

Growing Neural Gas

- Neuron Insertion: Periodically, based on certain criteria (such as the age of neurons or the level of error), new neurons are inserted into the network. These new neurons are placed in areas of the input space where the network lacks representation or where the data density is high. Connections to neighboring neurons are adjusted accordingly.

- Neuron Removal: Also periodically, neurons that are deemed unnecessary or redundant are removed from the network. This helps maintain a compact and efficient network structure. Neurons may be removed based on criteria like low activity or being surpassed by newer neurons in terms of representing the input data.

Cont.

- Repeat: Steps 2-5 are repeated iteratively for multiple epochs or until convergence criteria are met.

- The ability of GNG to dynamically adjust its structure makes it suitable for applications where the data distribution may change over time or where the complexity of the data is not known beforehand. It's commonly used for tasks such as data clustering, visualization of high-dimensional data, and feature extraction.
- Overall, Growing Neural Gas is a powerful technique for unsupervised learning that offers flexibility and adaptability in modeling complex datasets.

Adaptive resonance theory

- Adaptive Resonance Theory (ART) is a class of neural network models proposed by Stephen Grossberg and Gail Carpenter in the 1980s. The fundamental idea behind ART is to provide a biologically plausible framework for unsupervised learning, particularly in situations where there is a need for stable category learning despite changes in input patterns or environmental conditions.

- The key concept in ART is the notion of resonance, which refers to the match between the current input pattern and the network's learned categories. When an input pattern matches an existing category, the network resonates, reinforcing the category. If the input pattern does not match any existing category sufficiently, the network creates a new category to represent the input. This adaptive mechanism allows the network to learn and adapt to new input patterns while maintaining stability in learned categories.

Cont.

- ART models typically consist of two main components:

- Recognition Network: This network component processes incoming input patterns and compares them to the existing category prototypes. It computes a measure of similarity between the input pattern and each category prototype and selects the best matching category based on a comparison with a vigilance parameter.

- Learning Network: This component is responsible for updating category prototypes based on input patterns. When a match occurs between the input pattern and an existing category, the learning network adjusts the category prototype to better match the input. If no match is found, the learning network creates a new category prototype based on the input pattern.

Cont.

- There are several variants of ART models, each with its specific architecture and learning dynamics. Some common variants include ART1, ART2, and ARTMAP, each tailored for different types of data and learning tasks.

- ART has found applications in various domains, including pattern recognition, classification, clustering, and cognitive modeling. Its ability to adaptively learn and categorize input patterns while maintaining stability makes it suitable for tasks where the input distribution may change over time or where there is a need for incremental learning.

Orienting sub systems

Adaptive Resonance Theory (ART), "orienting subsystems" refer to mechanisms that facilitate the selection and processing of relevant input patterns. These subsystems play a crucial role in directing attention towards salient features of the input and in determining which patterns should be attended to and categorized by the network.

In ART models, particularly ART1 and ART2, orienting subsystems serve several functions:

Selective Attention: Orienting subsystems help filter incoming sensory information, focusing attention on relevant features or patterns while disregarding irrelevant or noisy input. This selective attention mechanism enables the network to prioritize important information for learning and categorization.

Cont.

- Vigilance Control: Vigilance refers to the degree of strictness in matching input patterns to existing categories. Orienting subsystems regulate the vigilance parameter, which determines the threshold for accepting a new input pattern as a member of an existing category. Adjusting vigilance allows the network to maintain stability in learned categories while remaining flexible to accommodate new patterns.

- Adaptive Gain Control: Orienting subsystems also modulate the sensitivity of the network to input patterns. By dynamically adjusting the gain or amplification of input signals, these subsystems regulate the network's responsiveness to changes in input patterns or environmental conditions. Adaptive gain control helps ensure that the network can effectively adapt to varying levels of input intensity or noise.

Cont.

- Inhibitory Interactions: In some ART variants, orienting subsystems may involve inhibitory interactions between neurons or categories. These inhibitory connections serve to suppress the activation of certain neurons or categories, preventing them from being selected or reinforced during learning. Inhibitory interactions contribute to the competitive dynamics within the network, facilitating the emergence of distinct and stable categories.

- Overall, orienting subsystems in ART models play a critical role in shaping the network's learning dynamics, attentional mechanisms, and adaptive behavior. By controlling attention, vigilance, gain, and inhibition, these subsystems enable the network to effectively process and categorize complex input patterns in a dynamic and adaptive manner.

Learning Laws

- "learning laws" refer to mathematical rules or algorithms that dictate how the network's parameters, such as connection weights, are updated during the learning process. These learning laws are essential for adjusting the network's behavior based on input-output relationships and minimizing the difference between the network's predictions and the desired outputs.

- There are several types of learning laws used in artificial neural networks, including:

- Hebbian Learning: Hebbian learning is a basic principle in neuroscience that states "cells that fire together, wire together." In artificial neural networks, Hebbian learning refers to a rule where connections between neurons are strengthened if the neurons on both ends are activated simultaneously. This learning law is often used for unsupervised learning tasks, such as associative memory and feature extraction.

Cont.

- Error-Correction Learning (Supervised Learning): In supervised learning, the network is provided with input-output pairs, and the goal is to minimize the error between the network's predictions and the desired outputs. Learning laws such as the delta rule (also known as the Widrow-Hoff rule) and backpropagation are commonly used for adjusting connection weights to reduce this error. Backpropagation is particularly popular in multilayer perceptron (MLP) networks and deep learning models.

- Competitive Learning: Competitive learning is a form of unsupervised learning where neurons compete to be activated in response to input patterns. The winner-takes-all mechanism is employed, and the connection weights to the winning neuron are adjusted to enhance its response to similar input patterns while suppressing responses from other neurons. The Kohonen Self-Organizing Map (SOM) is an example of a network that uses competitive learning.

Cont.

- Reinforcement Learning: In reinforcement learning, the network learns to take actions in an environment to maximize some notion of cumulative reward. Learning laws in reinforcement learning algorithms, such as Q-learning and policy gradients, govern how the network updates its policies or action-selection strategies based on received rewards or penalties.

- Online Learning vs. Batch Learning: Learning laws can also differ based on whether the network updates its weights incrementally after each input sample (online learning) or after processing a batch of input samples (batch learning). Online learning is often used for tasks with streaming data, while batch learning is common in offline settings where all data is available at once.

Thankyou