UNIT 6

Feedforward Networks & Unsupervised Learning in Neural Pattern Recognition Multilayer, Feedforward Network Structure, Training the Feedforward Network: The Delta Rule, Generalized Delta Rule (GDR), Extension of the DR for Units in the Hidden layers, Pattern Associator for Character Classification.

Unsupervised Learning in NeurPR: Self organizing Networks: Introduction, Adaptive Resonance Architectures, Self-Organizing Feature Maps (Kohonen).

Feedforward Networks:

What is a Feedforward Neural Network?

A feedforward neural network is one of the simplest types of artificial neural networks devised. In this network, the information moves in only one direction—forward—from the input nodes, through the hidden nodes (if any), and to the output nodes. There are no cycles or loops in the network. Feedforward neural networks were the first type of artificial neural network invented and are simpler than their counterparts like recurrent neural networks and convolutional neural networks.

Architecture of Feedforward Neural Networks

The architecture of a feedforward neural network consists of three types of layers: the input layer, hidden layers, and the output layer. Each layer is made up of units known as neurons, and the layers are interconnected by weights.

- **Input Layer:** This layer consists of neurons that receive inputs and pass them on to the next layer. The number of neurons in the input layer is determined by the dimensions of the input data.
- **Hidden Layers:** These layers are not exposed to the input or output and can be considered as the computational engine of the neural network. Each hidden layer's neurons take the weighted sum of the outputs from the previous

layer, apply an activation function, and pass the result to the next layer. The network can have zero or more hidden layers.

- **Output Layer:** The final layer that produces the output for the given inputs. The number of neurons in the output layer depends on the number of possible outputs the network is designed to produce.

Each neuron in one layer is connected to every neuron in the next layer, making this a fully connected network. The strength of the connection between neurons is represented by weights, and learning in a neural network involves updating these weights based on the error of the output.

How Feedforward Neural Networks Work

The working of a feedforward neural network involves two phases: the feedforward phase and the backpropagation phase.

- **Feedforward Phase:** In this phase, the input data is fed into the network, and it propagates forward through the network. At each hidden layer, the weighted sum of the inputs is calculated and passed through an activation function, which introduces non-linearity into the model. This process continues until the output layer is reached, and a prediction is made.
- **Backpropagation Phase:** Once a prediction is made, the error (difference between the predicted output and the actual output) is calculated. This error is then propagated back through the network, and the weights are adjusted to minimize this error. The process of adjusting weights is typically done using a gradient descent optimization algorithm.

Activation Functions

Activation functions play a crucial role in feedforward neural networks. They introduce non-linear properties to the network, which allows the model to learn more complex patterns. Common activation functions include the sigmoid, tanh, and ReLU (Rectified Linear Unit).

Training Feedforward Neural Networks

Training a feedforward neural network involves using a dataset to adjust the weights of the connections between neurons. This is done through an iterative process where the dataset is passed through the network multiple times, and each time, the weights are updated to reduce the error in prediction. This

process is known as gradient descent, and it continues until the network performs satisfactorily on the training data.

Applications of Feedforward Neural Networks

Feedforward neural networks are used in a variety of machine learning tasks including:

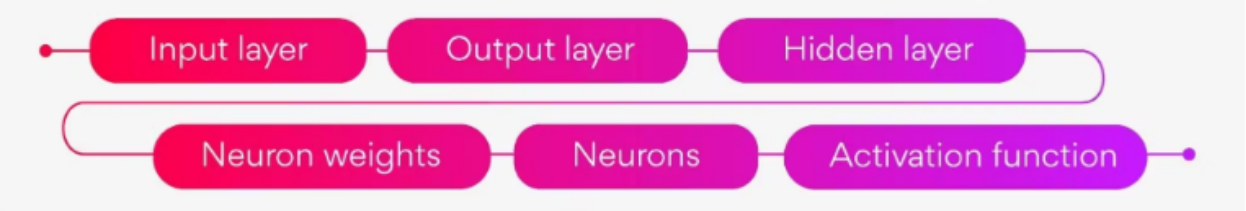
- Pattern recognition
- Classification tasks
- Regression analysis
- Image recognition
- Time series prediction

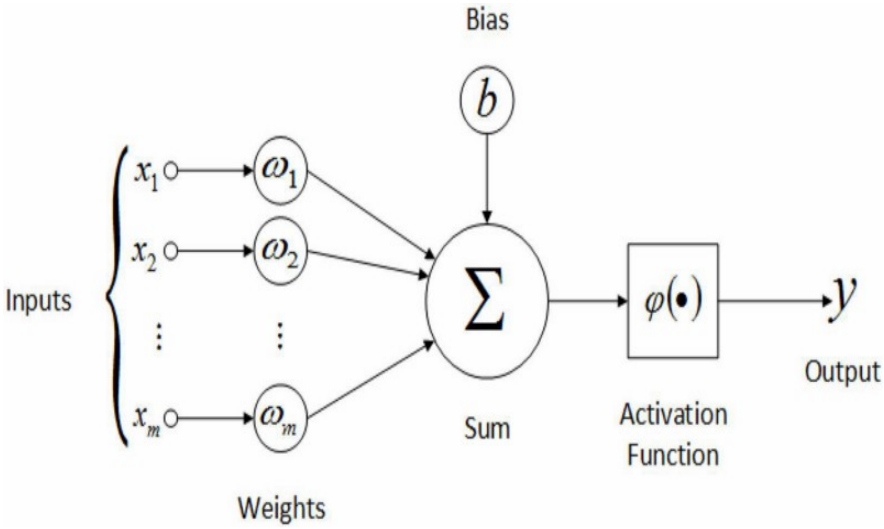
Despite their simplicity, feedforward neural networks can model complex relationships in data and have been the foundation for more complex neural network architectures.

Challenges and Limitations

While feedforward neural networks are powerful, they come with their own set of challenges and limitations. One of the main challenges is the choice of the number of hidden layers and the number of neurons in each layer, which can significantly affect the performance of the network. Overfitting is another common issue where the network learns the training data too well, including the noise, and performs poorly on new, unseen data.

In conclusion, feedforward neural networks are a foundational concept in the field of neural networks and deep learning. They provide a straightforward approach to modeling data and making predictions and have paved the way for more advanced neural network architectures used in modern artificial intelligence applications.





Unsupervised Learning in Neural Pattern Recognition Multilayer:

Unsupervised learning is a way of making full use of unlabeled data. K-means is the most popular clustering algorithm because of its efficacy and simplicity.

Unsupervised learning plays a significant role in neural pattern recognition, particularly in the context of multilayer networks like auto encoders or certain types of deep learning architectures. In these cases, unsupervised learning focuses on identifying patterns or structures in data without labeled outputs, making it highly useful in applications like feature extraction, dimensionality reduction, and clustering.

Key Concepts of Unsupervised Learning in Neural Pattern Recognition (Multilayer Networks)

1. **Auto encoders:**
    - **Architecture:** Auto encoders are a type of neural network used for unsupervised learning, consisting of two main parts:

        - **Encoder**: Compresses the input into a lower-dimensional representation (latent space).
        - **Decoder**: Reconstructs the original input from the compressed representation.
    - **Goal**: The network learns to represent the data more compactly while minimizing the difference between the input and its reconstruction.
    - **Applications**: Feature extraction, dimensionality reduction, denoising, and anomaly detection.

2. Self-Organizing Maps (SOMs):

- SOMs are a type of unsupervised neural network that projects high-dimensional data into a lower-dimensional space, often a 2D grid, while preserving the topological properties of the input data.
- They are commonly used for clustering and visualization of complex, high-dimensional data.

3. Generative Adversarial Networks (GANs):

- Although GANs are mainly used in supervised tasks, unsupervised variations exist. In unsupervised GANs, the model learns to generate new data samples from a latent space without relying on labeled data.
- The Generator creates data samples, and the Discriminator tries to distinguish between real and generated data, forcing the generator to learn the underlying patterns in the dataset.

4. Unsupervised Feature Learning:

- Neural networks can learn useful features from unlabeled data in a completely unsupervised way. For instance, in the first few layers of a convolutional neural network (CNN), neurons can automatically learn to detect edges, textures, and more abstract features without supervision.
- These learned features can then be transferred to downstream tasks like classification or clustering.

Feedforward Network Structure:

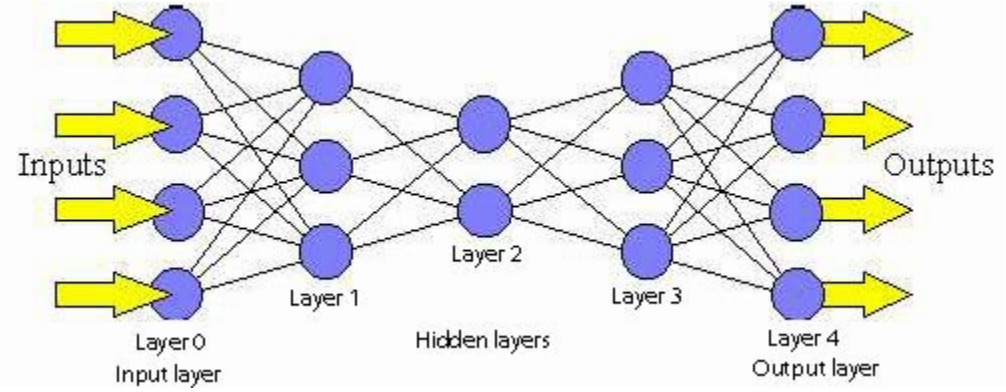


Fig.1 A feed-forward network.

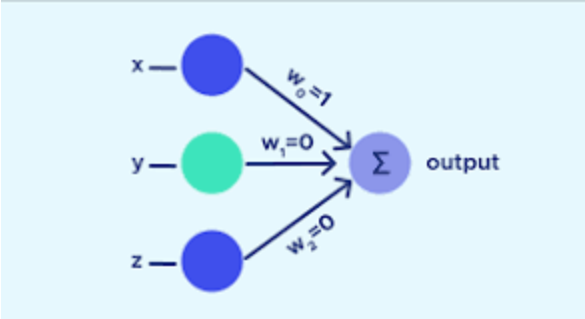
Feed-forward networks have the following characteristics:

1. Perceptrons are arranged in layers, with the first layer taking in inputs and the last layer producing outputs. The middle layers have no connection with the external world, and hence are called hidden layers.
2. Each perceptron in one layer is connected to every perceptron on the next layer. Hence information is constantly "fed forward" from one layer to the next., and this explains why these networks are called feed-forward networks.
3. There is no connection among perceptrons in the same layer.

What's about feed-forward networks?

Perceptron:

A Perceptron is an artificial neuron, and thus a neural network unit. It performs computations to detect features or patterns in the input data. It is an algorithm for supervised learning of binary classifiers. It is this algorithm that allows artificial neurons to learn and process features in a data set.



A single perceptron can classify points into two regions that are linearly separable. Now let us extend the discussion into the separation of points into two regions that are not linearly separable. Consider the following network:

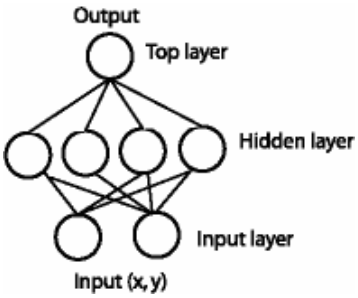


Fig.2 A feed-forward network with one hidden layer.

The same (x, y) is fed into the network through the perceptrons in the input layer. With four perceptrons that are independent of each other in the hidden layer, the point is classified into 4 pairs of linearly separable regions, each of which has a unique line separating the region.

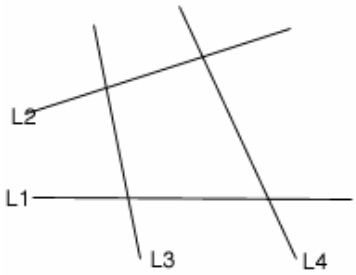
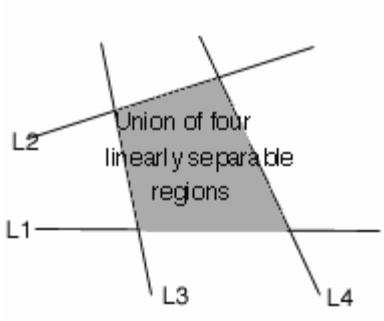


Fig.3- 4 lines each dividing the plane into 2 linearly separable regions.

The top perceptron performs logical operations on the outputs of the hidden layers so that the whole network classifies input points in 2 regions that might not be linearly separable. For instance, using the AND operator on these four outputs, one gets the intersection of the 4 regions that forms the center region.



(Fig.4) Intersection of 4 linearly separable regions forms the center region.

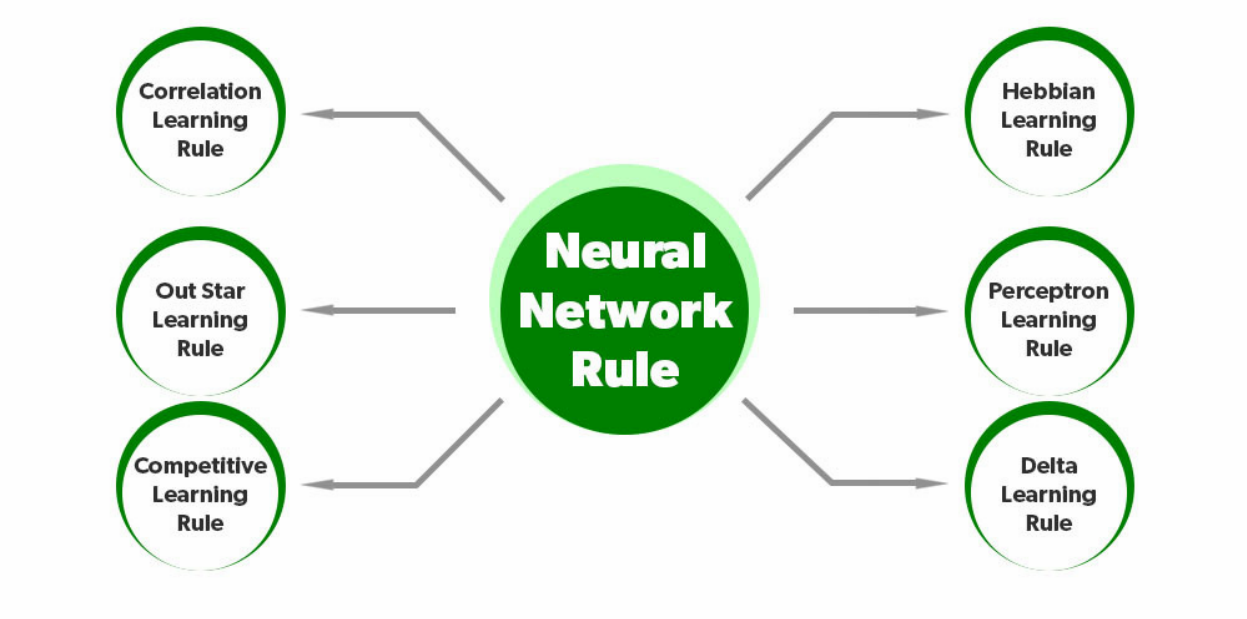
By varying the number of nodes in the hidden layer, the number of layers, and the number of input and output nodes, one can classification of points in arbitrary dimension into an arbitrary number of groups. Hence feed-forward networks are commonly used for classification.

Training the Feedforward Network-

Training a feedforward neural network involves using a dataset to adjust the weights of the connections between neurons. This is done through an iterative process where the dataset is passed through the network multiple times, and each time, the weights are updated to reduce the error in prediction.

Types of Learning Rule-

Learning rule enhances the Artificial Neural Network's performance by applying this rule over the network. Thus learning rule updates the weights and bias levels of a network when certain conditions are met in the training process. it is a crucial part of the development of the Neural Network.



1. The Delta Rule-

It depends on supervised learning and has a continuous activation function. It is also known as the Least Mean Square method and it minimizes error over all the training patterns.

It is based on a gradient descent approach which continues forever. It states that the modification in the weight of a node is equal to the product of the error and the input where the error is the difference between desired and actual output.

Computed as follows:

Assume (x1,x2,x3……..xn) –>set of input vectors

and (w1,w2,w3……...wn) –>set of weights

y=actual output

wo=initial weight

wnew=new weight

δw=change in weight

Error= ti-y

Learning signal(ej)=(ti-y)y'

y=f(net input)= ∫wixi

δw=αxiej=αxi(ti-y)y'

wnew=wo+δw

The updating of weights can only be done if there is a difference between the target and actual output(i.e., error) present:

case I: when t=y

then there is no change in weight

case II: else

wnew=wo+δw

2. Generalized Delta Rule (GDR)

A generalized delta rule is employed to determine how to update the neural network weights during the training process. The generalized delta rule refers to the backpropagation algorithm, which is basically an extension of the delta rule to deal with hidden layer problems.

***Delta Rule (For Single-Layer Networks)**:*

The delta rule is traditionally used in simple single-layer perceptrons (networks without hidden layers) and adjusts the weights based on the difference (error) between the actual output and the desired output. It's defined as:

$$\Delta w_i = \eta \cdot (t - y) \cdot x_i$$

Where:

- Δw_i is the weight update.
- η is the learning rate.
- t is the target (desired) output.
- y is the actual output.
- x_i is the input to the neuron.

This works well for simple networks, but it doesn't apply directly to networks with hidden layers since the error at the hidden layers isn't directly measurable.

2. Generalized Delta Rule (For Multilayer Networks):

To train **multilayer neural networks** with hidden layers, the **generalized delta rule** is used, which forms the basis of **backpropagation**. Backpropagation allows error signals to propagate backward from the output layer through the hidden layers to adjust weights in the entire network. Here's how it works:

Steps of the Backpropagation Algorithm:

Forward Pass:

The input is passed through the network, layer by layer, until an output is produced at the final layer.

For each neuron, the activation is calculated as the weighted sum of the inputs, followed by a non-linear activation function (such as sigmoid, ReLU, or tanh).

Error Calculation:

The error is computed as the difference between the actual output and the expected output at the output layer. For example, if the output neuron uses a sigmoid

activation function, the error at the output is computed using the squared error function:

$$E = \frac{1}{2}(t - y)^2$$

Backward Pass (Backpropagation):

- Starting from the output layer, the error gradients are propagated backward through the network.
- The chain rule of calculus is applied to calculate how the error changes with respect to the weights. This provides the gradient for adjusting the weights.

Weight Updates:

- The weights are updated using gradient descent, with the weight change for each weight wij between neuron iii and neuron j being proportional to the partial derivative of the error with respect to that weight:

$$\Delta w_{ij} = -\eta \cdot \frac{\partial E}{\partial w_{ij}}$$

Where η is the learning rate and $\frac{\partial E}{\partial w_{ij}}$ is the gradient of the error with respect to the weight w_{ij}.

3. Backpropagation and Hidden Layers:

In networks with hidden layers, the error at the hidden units is unknown, but backpropagation solves this by propagating the error from the output layer back through the hidden layers using the chain rule. For each hidden layer, the error gradient is computed based on the gradient from the layer ahead.

For hidden neurons, the error for neuron jjj in the hidden layer can be calculated as:

$$\delta_j = h'(z_j) \sum_k \delta_k w_{jk}$$

Where:

- δ_j is the error for neuron j,
- $h'(z_j)$ is the derivative of the activation function for the hidden layer,
- δ_k is the error at the next layer (in the forward direction),
- w_{jk} is the weight between neuron j and neuron k in the next layer.

The error gradients for the hidden layers help to adjust the weights, allowing the network to minimize the error.

Key Features of Backpropagation:

- **Gradient-Based**: Backpropagation relies on gradient descent to minimize the error function across the entire network.
- **Layer-by-Layer Weight Adjustment**: Weights are adjusted layer-by-layer in the network, ensuring the entire network learns the mapping from inputs to outputs effectively.
- **Computationally Efficient**: Backpropagation makes the training of deep neural networks computationally feasible by efficiently computing the gradients using the chain rule.