Back Propagation Network (BPN) – Detailed Notes

Back Propagation Network (BPN)

Introduction

A **Back Propagation Network (BPN)** is a type of artificial neural network that utilizes the backpropagation learning algorithm to adjust weights in order to minimize error. It is widely used in supervised learning tasks such as classification, regression, and pattern recognition.

Architecture of BPN

A typical BPN consists of three layers:

1. **Input Layer:** Receives external inputs and forwards them to the hidden layer.
2. **Hidden Layer(s):** Processes the input using weighted connections and an activation function.
3. **Output Layer:** Produces the final result of the network.

Each neuron in a layer is connected to neurons in the next layer through weighted connections. The network learns by adjusting these weights based on error feedback.

Working of Backpropagation Algorithm

The backpropagation algorithm works in two phases:

1. **Forward Propagation**
    - Inputs are passed through the network layer by layer.
    - Each neuron computes a weighted sum of inputs and applies an activation function.
    - The output is calculated and compared with the desired target.
2. **Backward Propagation (Weight Update)**
    - The error is calculated as the difference between the predicted and actual output.
    - The gradient of the error function with respect to each weight is computed using **Gradient Descent**.

- Weights are updated in the reverse direction (from output to input) to minimize the error.

Mathematical Formulation

Let:

- x_i be the input to the network.
- w_{ij} be the weight between neuron i and neuron j.
- y_j be the output of a neuron.
- $f(x)$ be the activation function (commonly **sigmoid**: $f(x) = \frac{1}{1+e^{-x}}$).

Forward Pass

1. **Compute weighted sum:**

$$net_j = \sum (x_i w_{ij}) + b_j$$

2. **Apply activation function:**

$$y_j = f(net_j)$$

Backward Pass (Error Computation)

1. Compute error:

$$e = \frac{1}{2} \sum (y_{target} - y_{output})^2$$

2. Compute gradient of error w.r.t weights:

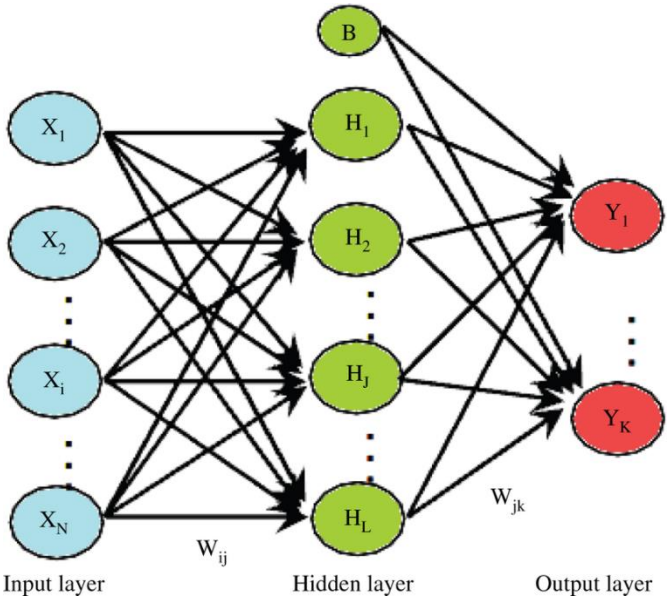
$$\frac{\partial e}{\partial w_{ij}} = (y_{target} - y_{output}) \cdot f'(net_j) \cdot y_j$$

3. Update weights using **Gradient Descent:**

$$w_{ij}^{new} = w_{ij}^{old} + \eta \cdot \frac{\partial e}{\partial w_{ij}}$$

where η is the learning rate.

Network Diagram of BPN



A typical BPN consists of:

- **Input Layer** (Receives input features)
- **Hidden Layer** (Processes data using weighted connections and activation functions)
- **Output Layer** (Produces the final output)

Advantages of BPN

- Efficient for complex pattern recognition.
- Can model non-linear relationships.
- Works well for supervised learning tasks.

Disadvantages of BPN

- Training can be slow for deep networks.
- Prone to getting stuck in local minima.
- Requires a large amount of labeled data.