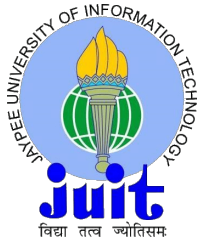


Machine Learning

(Course Code: 18B1WCI634)

Backpropagation

Mr. Sandeep Kumar Patel

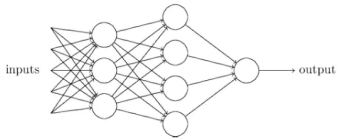
Assistant Professor

Jaypee University of Information Technology

Waknaghat, Solan, HP-173234

Multi-Layer Perceptrons (MLP)

- **Single-layer:** Simple decision boundaries.
- **Multi-layer:** Combination of perceptrons allows complex, non-linear decision-making.
- Layers:
  - Input layer — Receives data
  - Hidden layers — Process information
  - Output layer — Produces result
- Each layer feeds its output as input to the next.



COMPLEX PERCEPTRON

What is Backpropagation?

- -Algorithm to train multi-layer neural networks.
- -Extends Gradient Descent to networks with hidden layers.
- -Adjusts weights to minimize total error.

Algorithm Steps

- Forward pass: compute outputs of all neurons.
- Compute δ for output layer.
- Compute δ for hidden layers.
- Update weights: $w_{ji} := w_{ji} + \eta \delta_j o_i$.
- Repeat until total error is minimized.

Intuition

- Output layer: directly compares prediction with target.
- Hidden layer: adjusts based on contribution to output error.
- Weight changes proportional to:

$$\text{learning rate} \times \text{error} \times \text{input activation}$$

Forward Pass (Idea)

- Input data flows from input layer to output layer.
- Each neuron computes weighted sum + activation.

For neuron j:

$$net_j = \sum_i w_{ji} o_i + b_j$$

$$o_j = f(net_j)$$

- w_{ji} = weight
- o_i = input from previous layer
- f = activation function

Loss Function

We define error as:

$$E = \frac{1}{2}\sum_{j}(t_j - o_j)^2$$

- t_j = target output
- o_j = predicted output

Backward Pass (Idea)

- Goal: reduce error E
- Compute gradients of error w.r.t weights
- Propagate error from output layer to hidden layers

Chain Rule Derivation (General)

We compute gradient of weight:

$$\frac{\partial E}{\partial w_{ji}}$$

Using chain rule:

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial o_j} \times \frac{\partial o_j}{\partial net_j} \times \frac{\partial net_j}{\partial w_{ji}}$$

Output Layer Derivation

By the chain rule:

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial o_j} \times \frac{\partial o_j}{\partial net_j} \times \frac{\partial net_j}{\partial w_{ji}}$$

Substitute simple parts:

$$\frac{\partial E}{\partial o_j} = -(t_j - o_j), \quad \frac{\partial o_j}{\partial net_j} = f'(net_j), \quad \frac{\partial net_j}{\partial w_{ji}} = o_i$$

Combine them:

$$\frac{\partial E}{\partial w_{ji}} = -(t_j - o_j) f'(net_j) o_i$$

Define the error signal:

$$\delta_j = (t_j - o_j) f'(net_j)$$

Final weight update rule:

$$\boxed{\Delta w_{ji} = \eta \, \delta_j \, o_i}$$

Hidden Layer Derivation

Hidden units do not see the target directly — their error depends on the next layer.
We again use the chain rule:

$$\delta_h = f'(net_h) \sum_k (\delta_k \, w_{kh})$$

That means:

- Each hidden neuron gets error signals from next layer.
- It learns based on contribution to output error.

Weight update rule for hidden layer:

$$\boxed{\Delta w_{hi} = \eta \, \delta_h \, o_i}$$

Hidden Layer Derivation (Step-by-Step)

Goal: Compute hidden layer error signal $\delta_h = \frac{\partial E}{\partial net_h}$

Step 1: Chain Rule

$$\delta_h = \frac{\partial E}{\partial o_h} \cdot \frac{\partial o_h}{\partial net_h}$$

$$\Rightarrow \delta_h = f'(net_h) \frac{\partial E}{\partial o_h}$$

Step 2: Error from next layer

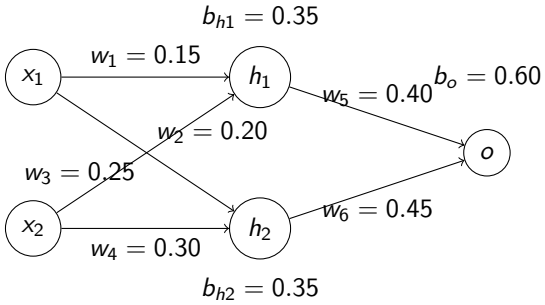
$$\frac{\partial E}{\partial o_h} = \sum_k \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial o_h}$$

$$= \sum_k \delta_k \cdot w_{kh}$$

Final Result

$$\boxed{\delta_h = f'(net_h) \sum_k (\delta_k w_{kh})}$$

Neural Network Diagram with Given Weights



Network and Parameters

We use a 2–2–1 network (2 inputs, 2 hidden units, 1 output). Activation: sigmoid

$$f(z) = \frac{1}{1 + e^{-z}}, \qquad f'(z) = f(z)(1 - f(z)).$$

Inputs / target / learning rate:

$$x_1 = 0.05, \quad x_2 = 0.10, \quad t = 0.01, \quad \eta = 0.5.$$

Initial weights and biases:

$$w_1 = 0.15,\ w_2 = 0.20,\ w_3 = 0.25,\ w_4 = 0.30 \quad (\text{input} \to \text{hidden})$$
$$b_{h1} = b_{h2} = 0.35$$
$$w_5 = 0.40,\ w_6 = 0.45 \quad (\text{hidden} \to \text{output})$$
$$b_o = 0.60$$

Forward Pass: Hidden Layer (Step-by-step)

Compute net inputs to hidden units:

$$\text{net}_{h1} = x_1 w_1 + x_2 w_2 + b_{h1} = 0.05(0.15) + 0.10(0.20) + 0.35 = 0.3775,$$

$$\text{net}_{h2} = x_1 w_3 + x_2 w_4 + b_{h2} = 0.05(0.25) + 0.10(0.30) + 0.35 = 0.3925.$$

Apply sigmoid activation:

$$o_{h1} = f(0.3775) \approx 0.5932699921,$$

$$o_{h2} = f(0.3925) \approx 0.5968843783.$$

(Values shown to 10 decimal places where helpful.)

Forward Pass: Output Layer & Error

Net input to output neuron:

$$\begin{aligned}\text{net}_o &= o_{h1}w_5 + o_{h2}w_6 + b_o \\ &= 0.5932699921(0.40) + 0.5968843783(0.45) + 0.60 \\ &\approx 1.1059059671.\end{aligned}$$

Output activation:

$$o_o = f(\text{net}_o) \approx 0.7513650696.$$

Squared error for this example:

$$E = \tfrac{1}{2}(t - o_o)^2 = \tfrac{1}{2}(0.01 - 0.7513650696)^2 \approx 0.2748110832.$$

Backpropagation: Output Layer

Compute output-layer delta:

$$\delta_o = (t - o_o)\, f'(\text{net}_o) = (0.01 - 0.7513650696) \cdot o_o(1 - o_o).$$

Numeric value:

$$\delta_o \approx -0.1384985616.$$

Gradients for weights w_5, w_6 and bias b_o:

$$\Delta w_5 = \eta\, \delta_o\, o_{h1} = 0.5 \times (-0.1384985616) \times 0.5932699921 \approx -0.04108352028,$$

$$\Delta w_6 = \eta\, \delta_o\, o_{h2} = 0.5 \times (-0.1384985616) \times 0.5968843783 \approx -0.04133381392,$$

$$\Delta b_o = \eta\, \delta_o = 0.5 \times (-0.1384985616) \approx -0.06924928081.$$

Backpropagation: Hidden Layer

Hidden deltas use downstream deltas and weights:

$$\delta_h = f'(\text{net}_h) \sum_k \delta_k w_{kh}.$$

Compute for each hidden unit:
For h_1:

$$\delta_{h1} = o_{h1}(1 - o_{h1}) \cdot (\delta_o \cdot w_5) \approx 0.59327(1 - 0.59327) \cdot (-0.13849856)(0.40) \approx -0.01336792042.$$

For h_2:

$$\delta_{h2} = o_{h2}(1 - o_{h2}) \cdot (\delta_o \cdot w_6) \approx 0.59688(1 - 0.59688) \cdot (-0.13849856)(0.45) \approx -0.01499607549.$$

Gradients for Input→Hidden Weights

Compute weight changes from inputs to hidden units:

$$\Delta w_{ji} = \eta\, \delta_j\, x_i.$$

Thus:

$$\Delta w_1 = 0.5 \times (-0.01336792042) \times 0.05 \approx -0.00033419801,$$

$$\Delta w_2 = 0.5 \times (-0.01336792042) \times 0.10 \approx -0.00066839602,$$

$$\Delta w_3 = 0.5 \times (-0.01499607549) \times 0.05 \approx -0.00037490189,$$

$$\Delta w_4 = 0.5 \times (-0.01499607549) \times 0.10 \approx -0.00074980377.$$

Hidden biases:

$$\Delta b_{h1} = \eta\, \delta_{h1} \approx -0.00668396021, \quad \Delta b_{h2} = \eta\, \delta_{h2} \approx -0.00749803774.$$

Update Weights: One Gradient Step

Add deltas to original weights (new = old + delta):

$$
\begin{aligned}
w_1' &= 0.15 + (-0.00033419801) \approx 0.14966580199, \\
w_2' &= 0.20 + (-0.00066839602) \approx 0.19933160398, \\
w_3' &= 0.25 + (-0.00037490189) \approx 0.24962509811, \\
w_4' &= 0.30 + (-0.00074980377) \approx 0.29925019623, \\
w_5' &= 0.40 + (-0.04108352028) \approx 0.35891647972, \\
w_6' &= 0.45 + (-0.04133381392) \approx 0.40866618608, \\
b_{h1}' &\approx 0.35 + (-0.00668396021) \approx 0.34331603979, \\
b_{h2}' &\approx 0.35 + (-0.00749803774) \approx 0.34250196226, \\
b_o' &\approx 0.60 + (-0.06924928081) \approx 0.53075071919.
\end{aligned}
$$

Second Forward Pass (After Update)

Compute new hidden nets and activations using updated weights:

$$\text{net}'_{h1} = x_1 w'_1 + x_2 w'_2 + b'_{h1} = 0.05(0.149665802) + 0.10(0.199331604) + 0.34331603979 \approx 0.3707324903,$$

$$\text{net}'_{h2} = x_1 w'_3 + x_2 w'_4 + b'_{h2} = 0.05(0.2496250981) + 0.10(0.2992501962) + 0.34250196226 \approx 0.3849082368.$$

Activations:

$$o'_{h1} \approx f(0.3707324903) \approx 0.5916359620, \quad o'_{h2} \approx f(0.3849082368) \approx 0.5950563624.$$

Second Forward Pass: Output and Error

Output net and activation with updated hidden outputs and output weights:

$$\begin{aligned}\text{net}'_o &= o'_{h1}w'_5 + o'_{h2}w'_6 + b'_o \\ &\approx 0.5916359620(0.3589164797) + 0.5950563624(0.4086661861) + 0.53075071919 \\ &\approx 0.9862780301,\end{aligned}$$

$$o'_o = f(\text{net}'_o) \approx 0.7283521371.$$

New error:

$$E' = \tfrac{1}{2}(t - o'_o)^2 = \tfrac{1}{2}(0.01 - 0.7283521371)^2 \approx 0.2580148964.$$

Interpretation: Error decreased from ≈ 0.2748110832 to ≈ 0.2580148964 after one backpropagation update.