Neural Networks

Prof. Hariprasad Kodamana

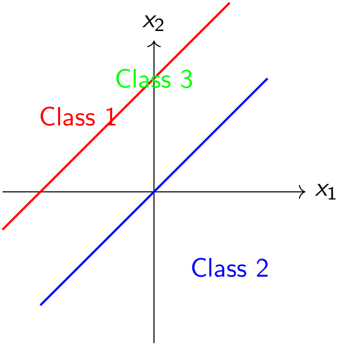
November 28, 2025

Introduction

- So far, we have learnt about linear models for regression.
- The goal in **regression** is to take an input vector x and predict output vector y (real numbers).
- The goal in **classification** is to take an input vector x and to assign it to one of K discrete classes C_k where $k = 1, ..., K$.
- The input space is thereby divided into decision regions whose boundaries are called **decision boundaries** or **decision surfaces**.

Linear Models for Classification

- We limit ourselves to linear models for classification.
- This means the decision surfaces are linear functions of the input vector x.
- Let us say, we want to classify samples to D classes; this would result in $(D-1)$-dimensional linear decision hyperplanes.



(Conceptual representation of linear boundaries)

Perceptron

1. It represents a two-class classifier using a generalized linear model structure.
2. Input vector x is first transformed using a fixed nonlinear transformation to give a feature vector $\phi(x)$, and this is then used to construct a generalized linear model.
$$y = f(w^T\phi(x))$$
3. If no modified features are used: $\phi(x) = x$
4. The non-linear activation function $f(.)$ is given by a step function of the form:
$$f(a) = \begin{cases} +1, & \text{if } a \geq 0 \\ -1, & \text{if } a < 0 \end{cases}$$
5. In earlier two class classification problem, we have focused on target coding scheme in which $y = \{0, 1\}$, however, it is more convenient to use target values $t_n = +1$ for class C_1 and $t_n = -1$ for class C_2 for perceptron.

Perceptron

1. We need to compute w by minimizing perceptron criterion.
2. For hard classification, for class C_1: $w^T\phi(x_n) > 0$ with $t_n = +1$, for class C_2: $w^T\phi(x_n) < 0$ with $t_n = -1$.
3. For correct classification, $w^T\phi(x_n)t_n > 0$, for incorrect classification $w^T\phi(x_n)t_n < 0$, always.
4. Therefore, in perceptron training, we minimize:

$$E_p(w) = \sum_{n \in M} -w^T\phi(x_n)t_n$$

   where, M are the misclassified points.
5. Stochastic gradient descent is used for optimization:

$$w = w - \alpha\nabla_w E_p(w) = w + \alpha\phi(x_n)t_n$$

6. Perform the above update till we get the desired classification accuracy.

Multi-Layer Perceptron (Feed Forward Neural Net)

1. The perceptron can only be expected to handle problems that are linearly separable.
2. To tackle more complicated (nonlinear) situations, we can increase the set of perceptrons in multiple layers.
3. The additional layers are called 'hidden' layers between the output and input layers.
4. Also commonly referred to as feed-forward neural nets or feed-forward networks.
5. According to Bishop- "Indeed, it has been used very broadly to cover a wide range of different models, many of which have been the subject of exaggerated claims regarding their biological plausibility."
6. The model comprises multiple layers of logistic regression models (with continuous nonlinearities) rather than multiple perceptrons (with discontinuous nonlinearities).

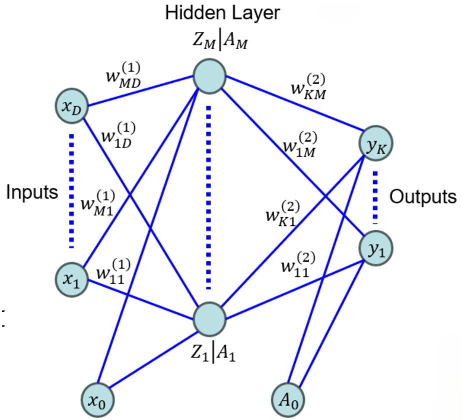
Multi-Layer Perceptron

Structure of basis functions:

$$Z_j(x, w) = \left(x_0 + \sum_{i=1}^{D} w_{ji}^{(1)} x_i \right); \quad j = 1, 2,$$

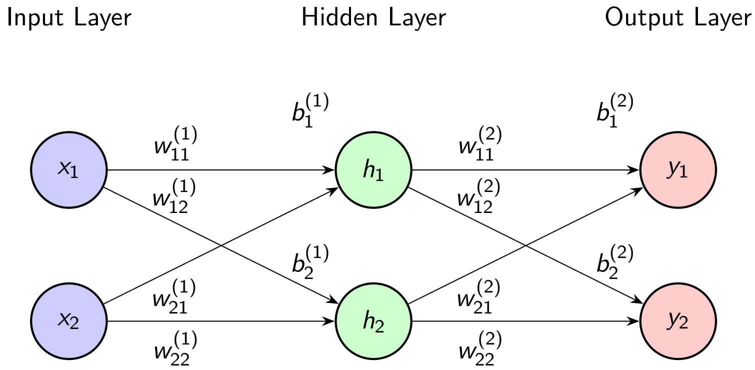
$$A_j(Z_j) = h(Z_j)$$

$$y_k(A, w) = \sigma \left(A_0 + \sum_{j=1}^{M} w_{kj}^{(2)} A_j \right); \quad k = :$$



In the above equation, $\sigma(.)$ is a non-linear activation function for classification and is the identity in the case of regression. In case of modified input features, $\phi_j(x)$ is considered in place of x.

Network Architecture with 2 inputs, 1 hidden layer, 2 outputs



Mathematical Notation

Layer Dimensions

- Input layer: 2 neurons (x_1, x_2)
- Hidden layer: 2 neurons (h_1, h_2)
- Output layer: 2 neurons (y_1, y_2)

Weight Matrices

- First layer weights: $W^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix}$
- Second layer weights: $W^{(2)} = \begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} \end{bmatrix}$

Bias Vectors

- Hidden layer biases: $b^{(1)} = [b_1^{(1)}, b_2^{(1)}]$
- Output layer biases: $b^{(2)} = [b_1^{(2)}, b_2^{(2)}]$

Activation Functions

General Activation Functions

- **Hidden layer activation:** $g^{(1)}$
- **Output layer activation:** $g^{(2)}$

Common Choices

- **Hidden layer:** ReLU, tanh, sigmoid, Leaky ReLU, etc.
- **Output layer:**
  - Softmax (multi-class classification)
  - Sigmoid (binary classification)
  - Linear (regression)
  - Tanh (bounded regression)

Properties

- Activation functions introduce non-linearity
- Enable learning complex patterns
- Different functions suit different tasks

Step 1: Input to Hidden Layer

Weighted Sum for Hidden Neurons

$$z_1^{(1)} = w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 + b_1^{(1)}$$
$$z_2^{(1)} = w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 + b_2^{(1)}$$

Matrix Form

$$z^{(1)} = \begin{bmatrix} z_1^{(1)} \\ z_2^{(1)} \end{bmatrix} = (W^{(1)})^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b^{(1)}$$

Activation Function

$$a_1 = g^{(1)}(z_1^{(1)})$$
$$a_2 = g^{(1)}(z_2^{(1)})$$

Step 2: Hidden to Output Layer

Weighted Sum for Output Neurons

$$z_1^{(2)} = w_{11}^{(2)} a_1 + w_{21}^{(2)} a_2 + b_1^{(2)}$$
$$z_2^{(2)} = w_{12}^{(2)} a_1 + w_{22}^{(2)} a_2 + b_2^{(2)}$$

Matrix Form

$$z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \end{bmatrix} = (W^{(2)})^T \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + b^{(2)}$$

Output Activation

$$y_1 = g^{(2)}(z_1^{(2)})$$
$$y_2 = g^{(2)}(z_2^{(2)})$$

Complete Forward Propagation

Step-by-Step Process

1. **Input:** $x = [x_1, x_2]$
2. **Hidden layer pre-activation:** $z^{(1)} = (W^{(1)})^T x + b^{(1)}$
3. **Hidden layer activation:** $a = g^{(1)}(z^{(1)})$
4. **Output layer pre-activation:** $z^{(2)} = (W^{(2)})^T a + b^{(2)}$
5. **Output:** $y = g^{(2)}(z^{(2)})$

Function Composition

$$y = f(x) = g^{(2)}\left((W^{(2)})^T \cdot g^{(1)}\left((W^{(1)})^T x + b^{(1)}\right) + b^{(2)}\right)$$

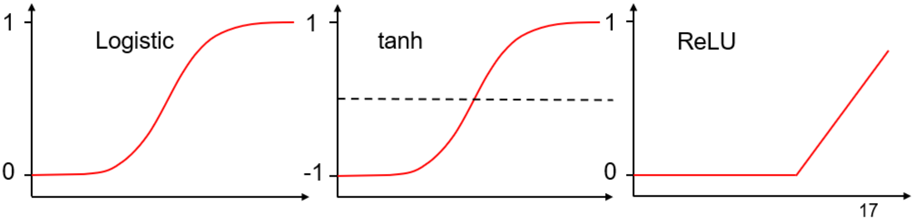
Multi-Layer Perceptron

1) Output activation function: mostly logistic function: $\sigma(z) = \frac{1}{1+\exp(-z)}$

2) Hidden Layer activation function:

- Logistic ([0,1]): $h(z) = \frac{1}{1+\exp(-z)}$
- tanh ([-1,1]): $h(z) = \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$
- ReLU:

$$\text{ReLU} = \begin{cases} 0, & \text{if } z \leq 0 \\ z, & \text{if } z > 0 \end{cases}$$



Multi-Layer Perceptron - Backpropagation

1. Since an error is computed at the output and distributed backwards throughout the network's layers, backpropagation of the error is performed while training the network.
2. The multilayer perceptron architecture is sometimes called a backpropagation network.
3. It involves two simple steps as given next:
4. In the first stage, the derivatives of the error function with respect to the weights must be evaluated.
5. In the second stage, the derivatives are then used to compute the updated weights.

Loss Function and Backpropagation Goal

General Loss Function

$$L = \mathcal{L}(a^{(2)}, t)$$

where $t = [t_1, t_2]$ are target values. $\mathcal{L}$ will be MSE for regression and cross entropy for classificaiton.

Backpropagation Objective

Compute gradients for all parameters:

$$\frac{\partial L}{\partial W^{(2)}} = \begin{bmatrix} \frac{\partial L}{\partial w_{11}^{(2)}} & \frac{\partial L}{\partial w_{12}^{(2)}} \\ \frac{\partial L}{\partial w_{21}^{(2)}} & \frac{\partial L}{\partial w_{22}^{(2)}} \end{bmatrix}, \qquad \frac{\partial L}{\partial b^{(2)}} = \begin{bmatrix} \frac{\partial L}{\partial b_1^{(2)}} \\ \frac{\partial L}{\partial b_2^{(2)}} \end{bmatrix}$$

$$\frac{\partial L}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial L}{\partial w_{11}^{(1)}} & \frac{\partial L}{\partial w_{12}^{(1)}} \\ \frac{\partial L}{\partial w_{21}^{(1)}} & \frac{\partial L}{\partial w_{22}^{(1)}} \end{bmatrix}, \qquad \frac{\partial L}{\partial b^{(1)}} = \begin{bmatrix} \frac{\partial L}{\partial b_1^{(1)}} \\ \frac{\partial L}{\partial b_2^{(1)}} \end{bmatrix}$$

Step 1: Define Error Terms $\delta^{(l)}$

Error Term Definition

For each layer l, define:

$$\delta^{(l)} = \frac{\partial L}{\partial z^{(l)}} = \begin{bmatrix} \frac{\partial L}{\partial z_1^{(l)}} \\ \frac{\partial L}{\partial z_2^{(l)}} \end{bmatrix}$$

Purpose of Error Terms

- $\delta^{(l)}$ measures sensitivity of loss to pre-activations $z^{(l)}$
- Enables efficient gradient computation via chain rule
- Error flows backward through the network

Step 2: Output Layer Error $\delta^{(2)}$

Chain Rule Application

$$\delta_i^{(2)} = \frac{\partial L}{\partial z_i^{(2)}} = \frac{\partial L}{\partial a_i^{(2)}} \cdot \frac{\partial a_i^{(2)}}{\partial z_i^{(2)}}$$

Component-wise Derivation

For $i = 1, 2$:

$$\delta_1^{(2)} = \frac{\partial L}{\partial a_1^{(2)}} \cdot \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} = \frac{\partial L}{\partial a_1^{(2)}} \cdot g^{(2)\prime}(z_1^{(2)})$$

$$\delta_2^{(2)} = \frac{\partial L}{\partial a_2^{(2)}} \cdot \frac{\partial a_2^{(2)}}{\partial z_2^{(2)}} = \frac{\partial L}{\partial a_2^{(2)}} \cdot g^{(2)\prime}(z_2^{(2)})$$

Step 3: Hidden Layer Error $\delta^{(1)}$

Chain Rule Through Layer 2

$$\delta_j^{(1)} = \frac{\partial L}{\partial z_j^{(1)}} = \sum_{k=1}^{2} \frac{\partial L}{\partial z_k^{(2)}} \cdot \frac{\partial z_k^{(2)}}{\partial a_j^{(1)}} \cdot \frac{\partial a_j^{(1)}}{\partial z_j^{(1)}}$$

Detailed Derivation

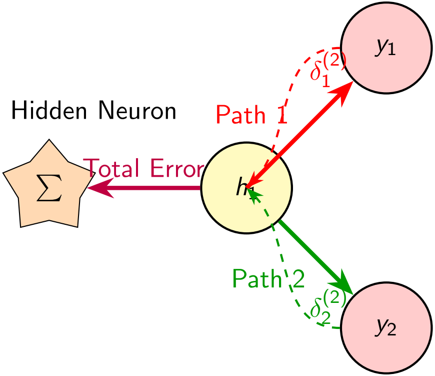
$$\begin{aligned}
\delta_j^{(1)} &= \sum_{k=1}^{2} \delta_k^{(2)} \cdot \frac{\partial z_k^{(2)}}{\partial a_j^{(1)}} \cdot \frac{\partial a_j^{(1)}}{\partial z_j^{(1)}} \\
&= \sum_{k=1}^{2} \delta_k^{(2)} \cdot w_{jk}^{(2)} \cdot g^{(1)\prime}(z_j^{(1)}) \\
&= g^{(1)\prime}(z_j^{(1)}) \cdot \sum_{k=1}^{2} w_{jk}^{(2)} \delta_k^{(2)}
\end{aligned}$$

Hidden Layer Error (Continued)

Component-wise Expressions

$$\delta_1^{(1)} = g^{(1)\prime}(z_1^{(1)}) \cdot \left(w_{11}^{(2)} \delta_1^{(2)} + w_{12}^{(2)} \delta_2^{(2)} \right)$$
$$\delta_2^{(1)} = g^{(1)\prime}(z_2^{(1)}) \cdot \left(w_{21}^{(2)} \delta_1^{(2)} + w_{22}^{(2)} \delta_2^{(2)} \right)$$

Why Summation? Multiple Influence Paths



Key Insight

- Each hidden neuron influences **multiple output neurons**
- Therefore, it receives error signals from **all outputs it affects**
- The **summation combines** these multiple error contributions
- Without summation, the network would miss important learning signals

Step 4: Weight Gradients $\frac{\partial L}{\partial W^{(2)}}$

Chain Rule for Output Layer Weights

$$\frac{\partial L}{\partial w_{ij}^{(2)}} = \frac{\partial L}{\partial z_i^{(2)}} \cdot \frac{\partial z_i^{(2)}}{\partial w_{ij}^{(2)}}$$

Detailed Derivation

$$\begin{aligned}
\frac{\partial L}{\partial w_{ij}^{(2)}} &= \delta_i^{(2)} \cdot \frac{\partial z_i^{(2)}}{\partial w_{ij}^{(2)}} \\
&= \delta_i^{(2)} \cdot \frac{\partial}{\partial w_{ij}^{(2)}} \left(\sum_{m=1}^{2} w_{mi}^{(2)} a_m^{(1)} + b_i^{(2)} \right) \\
&= \delta_i^{(2)} \cdot a_j^{(1)}
\end{aligned}$$

Step 5: Weight Gradients $\frac{\partial L}{\partial W^{(1)}}$

Chain Rule for Hidden Layer Weights

$$\frac{\partial L}{\partial w_{ij}^{(1)}} = \frac{\partial L}{\partial z_j^{(1)}} \cdot \frac{\partial z_j^{(1)}}{\partial w_{ij}^{(1)}}$$

Detailed Derivation

$$\begin{aligned}
\frac{\partial L}{\partial w_{ij}^{(1)}} &= \delta_j^{(1)} \cdot \frac{\partial z_j^{(1)}}{\partial w_{ij}^{(1)}} \\
&= \delta_j^{(1)} \cdot \frac{\partial}{\partial w_{ij}^{(1)}} \left(\sum_{m=1}^{2} w_{mj}^{(1)} a_m^{(0)} + b_j^{(1)} \right) \\
&= \delta_j^{(1)} \cdot a_i^{(0)}
\end{aligned}$$

Step 6: Bias Gradients

Output Layer Biases

$$\frac{\partial L}{\partial b_i^{(2)}} = \frac{\partial L}{\partial z_i^{(2)}} \cdot \frac{\partial z_i^{(2)}}{\partial b_i^{(2)}} = \delta_i^{(2)} \cdot 1 = \delta_i^{(2)}$$

Hidden Layer Biases

$$\frac{\partial L}{\partial b_j^{(1)}} = \frac{\partial L}{\partial z_j^{(1)}} \cdot \frac{\partial z_j^{(1)}}{\partial b_j^{(1)}} = \delta_j^{(1)} \cdot 1 = \delta_j^{(1)}$$

Vector Form

$$\frac{\partial L}{\partial b^{(2)}} = \delta^{(2)}$$
$$\frac{\partial L}{\partial b^{(1)}} = \delta^{(1)}$$

Complete Backpropagation Equations

Parameter Gradients

$$\frac{\partial L}{\partial W^{(2)}} = a^{(1)}(\delta^{(2)})^T$$
$$\frac{\partial L}{\partial W^{(1)}} = a^{(0)}(\delta^{(1)})^T$$
$$\frac{\partial L}{\partial b^{(2)}} = \delta^{(2)}$$
$$\frac{\partial L}{\partial b^{(1)}} = \delta^{(1)}$$

Multi-Layer Perceptron General Structure

1) Output of first hidden layer:

$$Z_j(x, w) = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

$$A_j(Z_j) = h(Z_j); \quad j = 1, 2, \ldots, M$$

Input variables: $x_1, x_2, \ldots, x_D$; bias: x_0, hidden layer units: $j = 1, 2, \ldots, M$; weights: $w_{ji}^{(1)}$; hidden layer activations: Z_j; hidden layer activation function: h.

2) Output unit activation Z_k:

$$Z_k(A, w) = \left(A_0 + \sum_{j=1}^{M} w_{kj}^{(2)} A_j \right); \quad k = 1, 2, \ldots, K$$

Total number of outputs: K; Bias: A_0; output weights: $w_{kj}^{(2)}$.

3) Outputs:

$$y_k = \begin{cases} Z_k, & \text{if regression} \\ \sigma(Z_k), & \text{if classification} \end{cases}$$

$\sigma(.)$ is the sigmoid function.

Multi-Layer Perceptron General Structure

1) Network output y_k:

$$y_k(x, w) = \sigma\left(A_0 + \sum_{j=1}^{M} w_{kj}^{(2)} h\left(x_0 + \sum_{l=1}^{D} w_{jl}^{(1)} x_l\right)\right)$$

1) Deep networks: generally used buzzword if there are multiple hidden layers.
2) Credit assignment path (CAP): chain of transformations from input to output, e.g., for a feedforward neural network, the depth of the CAP is the number of hidden layers plus one.
3) Deep network: CAP depth > 2.
4) CAP of depth 2 has been shown to satisfy the requirement of a universal approximator.
(The Universal Approximation Theorem tells us that Neural Networks have a kind of universality, i.e., no matter what $f(x)$ is, there is a network that can approximately approach the result. This result holds for any number of inputs and outputs.)

Multi-Layer Perceptron - Backpropagation General Structure

1) Notation k indicates output layer, notation j indicates hidden layers.
Error Function:

$$E(w) = \frac{1}{N} \sum_{n=1}^{N} E_n(w)$$

$$E_n(w) = \frac{1}{2} \sum_{k=1}^{K} (y_{nk} - t_{nk})^2$$

Note that this summation is not required for stochastic gradient descent.
2) Consider n^{th} data vector, then, for all outputs $y_1, y_2, \ldots, y_k$ y_{nk}: k^{th} output of the n^{th} data vector; t_{nk}: corresponding target of y_{nk}.
3) In a feed-forward network (forward propagation), the output of a r^{th} general hidden layer is obtained from activations at the $(r-1)^{th}$ layer:

$$A_j^{(r)} = h(Z_j^{(r)}) = h\left(A_0^{(r-1)} + \sum_{i=1}^{D} w_{ji}^{(r-1)} A_i^{(r-1)}\right)$$

Multi-Layer Perceptron - Backpropagation

In the given figure with just one hidden layer, we have:

$$A_j = h(Z_j) = h(x_0 + \sum_{i=1}^{D} w_{ji}^{(1)} x_i)$$

2) Gradient calculation for output layer: $\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \frac{\partial E_n}{\partial y_k} \times \frac{\partial y_k}{\partial w_{kj}^{(2)}}$

$$\frac{\partial E_n}{\partial y_k} = \frac{\partial}{\partial y_k} \left(\frac{1}{2} \sum_{k=1}^{K} (y_{nk} - t_{nk})^2 \right) = \sum_{k=1}^{K} (y_{nk} - t_{nk})$$

$$y_k(A, w) = A_0 + \sum_{j=1}^{M} w_{kj}^{(2)} A_j; \quad \frac{\partial y_k}{\partial w_{kj}^{(2)}} = \frac{\partial}{\partial w_{kj}^{(2)}} \left(A_0 + \sum_{j=1}^{M} w_{kj}^{(2)} A_j \right) = A_j$$

Multi-Layer Perceptron - Backpropagation

$$\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \frac{\partial E_n}{\partial y_k} \times \frac{\partial y_k}{\partial w_{kj}^{(2)}} = \sum_{k=1}^{K} (y_{nk} - t_{nk}) A_j$$

In case of batch gradient descent, E should be used instead of E_n.

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \frac{\partial E}{\partial y_k} \times \frac{\partial y_k}{\partial w_{kj}^{(2)}} = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_{nk} - t_{nk}) A_j$$

Multi-Layer Perceptron - Backpropagation

3) Gradient calculation for hidden layer:
For Regression Problem $\sigma(.) = 1$, and we get:

$$\frac{\partial A_j}{\partial Z_j} = \frac{\partial}{\partial Z_j} h(Z_j) = h'(Z_j)$$

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \frac{\partial E_n}{\partial y_k} \times \frac{\partial y_k}{\partial A_j} \times \frac{\partial A_j}{\partial Z_j} \times \frac{\partial Z_j}{\partial w_{ji}^{(1)}}$$

$$y_k(A, w) = (A_0 + \sum_{j=1}^{M} w_{kj}^{(2)} A_j) \Rightarrow \frac{\partial y_k}{\partial A_j} = w_{kj}^{(2)}$$

$$Z_j(x, w) = x_0 + \sum_{i=1}^{D} w_{ji}^{(1)} x_i \Rightarrow \frac{\partial Z_j}{\partial w_{ji}^{(1)}} = \frac{\partial}{\partial w_{ji}^{(1)}} \left(x_0 + \sum_{i=1}^{D} w_{ji}^{(1)} x_i \right) = x_i$$

Multi-Layer Perceptron - Backpropagation

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \sum_{k=1}^{K} (y_{nk} - t_{nk}) \times w_{kj}^{(2)} \times h'(Z_j) \times x_i$$

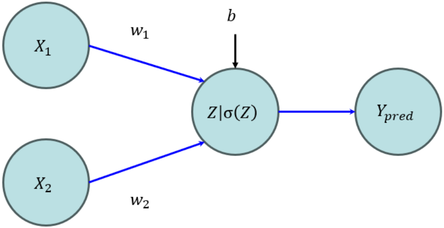
The above equation is valid only for stochastic gradient descent. For Batch Gradient Descent:

$$\frac{\partial E}{\partial w_{ji}^{(1)}} = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_{nk} - t_{nk}) \times w_{kj}^{(2)} \times h'(Z_j) \times x_i$$

Multi-Layer Perceptron – Example 1

True Underline expression is: $Y = X_1 + X_2$

SR. No.	X_1	X_2	Y
1	1	2	3
2	2	3	5
3	3	7	10



Step 1: Initialization
Weights and Bias: $w_1 = 0.5$, $w_2 = -0.5$, $b = 0.2$
Learning rate $\alpha = 0.10$
Activation function: sigmoid, $\sigma(Z) = \frac{1}{(1+e^{-Z})}$

Multi-Layer Perceptron – Example 1

$$Z = w_1X_1 + w_2X_2 + b; \quad Y_{pred} = \frac{1}{(1 + e^{-Z})}$$

Step 2: Calculate mean square error (MSE) loss

SR. No.	X_1	X_2	Y_{true}	Z	Y_{pred}
1	1	2	3	-0.3	0.425
2	2	3	5	-0.3	0.425
3	3	7	10	-1.8	0.142

$$MSE = \frac{1}{2n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i})^2$$

$$MSE = \frac{1}{2 \times 3}[(0.425 - 3)^2 + (0.425 - 5)^2 + (0.142 - 10)^2] = 20.79$$

Step 3: Calculate the Gradient For parameters:
$p = w_1, w_2, b \Rightarrow \frac{\partial MSE}{\partial p} = (\frac{\partial MSE}{\partial Y_{pred}}) \times (\frac{\partial Y_{pred}}{\partial Z}) \times (\frac{\partial Z}{\partial p})$

$$(\frac{\partial MSE}{\partial Y_{pred}}) = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}); \quad (\frac{\partial Y_{pred}}{\partial Z}) = Y_{pred}(1 - Y_{pred})$$

$$\frac{\partial Z}{\partial w_1} = X_1; \quad \frac{\partial Z}{\partial w_2} = X_2; \quad \frac{\partial Z}{\partial b} = 1$$

$$\frac{\partial MSE}{\partial w_1} = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times Y_{pred,i}(1 - Y_{pred,i}) \times X_{1,i} = \frac{1}{n}\sum_{i=1}^{n} P_i$$

$$\frac{\partial MSE}{\partial w_2} = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times Y_{pred,i}(1 - Y_{pred,i}) \times X_{2,i} = \frac{1}{n}\sum_{i=1}^{n} Q_i$$

$$\frac{\partial MSE}{\partial b} = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times Y_{pred,i}(1 - Y_{pred,i}) \times 1 = \frac{1}{n}\sum_{i=1}^{n} R_i$$

SR. No.	X_1	X_2	Y_{true}	Z	Y_{pred}	P	Q	R
1	1	2	3	-0.3	0.425	-0.629	-1.258	-0.629
2	2	3	5	-0.3	0.425	-2.236	-3.354	-1.118
3	3	7	10	-1.8	0.142	-3.603	-8.407	-1.201

Multi-Layer Perceptron – Example 1

$$\frac{\partial MSE}{\partial w_1} = \frac{1}{3}(-0.629 - 1.118 - 1.201) = -2.156$$

$$\frac{\partial MSE}{\partial w_2} = \frac{1}{3}(-1.258 - 3.354 - 8.407) = -4.339$$

$$\frac{\partial MSE}{\partial b} = \frac{1}{3}(-0.629 - 1.118 - 1.201) = -0.982$$

Step 4: Update the weights

$$w_1 = w_1 - \alpha\frac{\partial MSE}{\partial w_1} = 0.5 - 0.1 \times -2.156 = 0.7156$$

$$w_2 = w_2 - \alpha\frac{\partial MSE}{\partial w_2} = -0.5 - 0.1 \times -4.339 = -0.066$$

$$b = b - \alpha\frac{\partial MSE}{\partial b} = 0.2 - 0.1 \times -0.982 = 0.2982$$

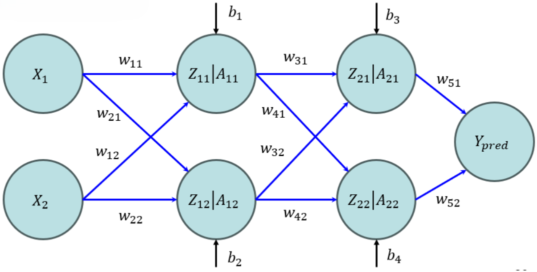
The procedure will be repeated for iteration 2 with new values of w_1, w_2, and b.

Multi-Layer Perceptron – Example 2

True Underline expression is:

$$Y = 0.25X_1^2 + 0.2X_1X_2 + 5$$

SR. No.	X_1	X_2	Y
1	1	3	5.85
2	2	4	7.6
3	3	5	10.25



Multi-Layer Perceptron – Example 2

Step 1: Initialization: Consider the weights from X_1 to X_2 to the first hidden layer.

$$w_{11} = -0.25, w_{21} = 0.901; w_{12} = 0.464; w_{22} = 0.197$$

Consider the weights from the first hidden layer to the second.

$$w_{31} = -0.688, w_{41} = -0.688; w_{32} = -0.884; w_{42} = 0.732$$

Consider the weights from the second hidden layer to the output layer.

$$w_{51} = -0.202, w_{52} = -0.416$$

Bias: $b_1 = -0.959$; $b_2 = 0.940$; $b_3 = 0.665$; $b_4 = -0.575$

Weights and biases are selected randomly between -1 and 1.
Learning rate $\alpha = 0.10$.
Activation function: sigmoid, $\sigma(Z) = \frac{1}{(1+e^{-Z})}$

Multi-Layer Perceptron – Example 2

Step 2: Calculate the outputs from inputs at each node in the forward direction.

$Z_{11} = w_{11}X_1 + w_{12}X_2 + b_1; A_{11} = \sigma(Z_{11})$

$Z_{12} = w_{21}X_1 + w_{22}X_2 + b_2; A_{12} = \sigma(Z_{12})$

$Z_{21} = w_{31}A_{11} + w_{32}A_{12} + b_3; A_{21} = \sigma(Z_{21})$

$Z_{22} = w_{41}A_{11} + w_{42}A_{12} + b_4; A_{22} = \sigma(Z_{22})$

$Y_{pred} = w_{51}A_{21} + w_{52}A_{22}$

First Hidden Layer
$w_{11} = -0.25, w_{21} = 0.901; w_{12} = 0.464; w_{22} = 0.197; b_1 = -0.959, b_2 = 0.940$

Second Hidden Layer
$w_{31} = -0.688, w_{41} = -0.688; w_{32} = -0.884; w_{42} = 0.732; b_3 = 0.665; b_4 = -0.575$

Output Layer
$w_{51} = -0.202, w_{52} = -0.416$

SN	X_1	X_2	Y_{true}	Z_{11}	A_{11}	Z_{12}	A_{12}	Z_{21}	A_{21}	Z_{22}	A_{22}	Y_{pred}
1	1	3	5.85	-0.959	0.277	0.940	0.719	-0.161	0.460	-0.239	0.440	0.276
2	2	4	7.6	-0.853	0.299	1.489	0.816	-0.262	0.435	-0.183	0.454	0.277
3	3	5	10.25	-0.746	0.322	2.038	0.885	-0.338	0.416	-0.148	0.463	0.277

Multi-Layer Perceptron – Example 2

Step 3: Backpropagation at output layer

$$\frac{\partial MSE}{\partial w_{51}} = \left(\frac{\partial MSE}{\partial Y_{pred}}\right) \times \left(\frac{\partial Y_{pred}}{\partial w_{51}}\right)$$

$$\frac{\partial MSE}{\partial Y_{pred}} = \frac{\partial}{\partial Y_{pred}}\left(\frac{1}{2n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i})^2\right) = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i})$$

$$\frac{\partial MSE}{\partial w_{51}} = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times \left(\frac{\partial Y_{pred}}{\partial w_{51}}\right)$$

$$= \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times \frac{\partial}{\partial w_{51}}(w_{51}A_{21} + w_{52}A_{22}) = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times A_{21}$$

$$\frac{\partial MSE}{\partial w_{51}} = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times A_{21} = -0.075$$

Similarly: $\frac{\partial MSE}{\partial w_{52}} = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times A_{22} = -0.089$

Multi-Layer Perceptron – Example 2

Step 4: Backpropagation from second hidden layer to first hidden layer

$$\frac{\partial MSE}{\partial w_{31}} = \left(\frac{\partial MSE}{\partial Y_{pred}}\right) \times \left(\frac{\partial Y_{pred}}{\partial A_{21}}\right) \times \left(\frac{\partial A_{21}}{\partial Z_{21}}\right) \times \left(\frac{\partial Z_{21}}{\partial w_{31}}\right)$$

$$\frac{\partial MSE}{\partial w_{32}} = \left(\frac{\partial MSE}{\partial Y_{pred}}\right) \times \left(\frac{\partial Y_{pred}}{\partial A_{21}}\right) \times \left(\frac{\partial A_{21}}{\partial Z_{21}}\right) \times \left(\frac{\partial Z_{21}}{\partial w_{32}}\right)$$

PLACEHOLDER FOR NETWORK DI

(Nodes X, $Z_{11}|A_{11}$, $Z_{21}|A_{21}$, Y_{pr}

Multi-Layer Perceptron – Example 2

Step 4 (continued): Derivation steps

$$\frac{\partial MSE}{\partial w_{31}} = \left(\frac{\partial MSE}{\partial Y_{pred}}\right) \times \left(\frac{\partial Y_{pred}}{\partial A_{21}}\right) \times \left(\frac{\partial A_{21}}{\partial Z_{21}}\right) \times \left(\frac{\partial Z_{21}}{\partial w_{31}}\right)$$

$$= \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times \frac{\partial}{\partial A_{21}}(w_{51}A_{21} + w_{52}A_{22}) \times \frac{\partial}{\partial Z_{21}}(\sigma(Z_{21}))$$

$$\times \frac{\partial}{\partial w_{31}}(w_{31}A_{11} + w_{32}A_{12} + b_3)$$

$$= \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times w_{51} \times \sigma(Z_{21})[1 - \sigma(Z_{21})] \times A_{11} = -0.003$$

Similarly:

$$\frac{\partial MSE}{\partial w_{32}} = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times w_{51} \times \sigma(Z_{21})[1 - \sigma(Z_{21})] \times A_{12} = -0.009$$

$$\frac{\partial MSE}{\partial b_3} = \left(\frac{\partial MSE}{\partial Y_{pred}}\right) \times \left(\frac{\partial Y_{pred}}{\partial A_{21}}\right) \times \left(\frac{\partial A_{21}}{\partial Z_{21}}\right) \times \left(\frac{\partial Z_{21}}{\partial b_3}\right)$$

$$= \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times w_{51} \times \sigma(Z_{21})[1 - \sigma(Z_{21})] \times 1 = -0.0092$$

Multi-Layer Perceptron – Example 2

$$\frac{\partial MSE}{\partial w_{41}} = (\frac{\partial MSE}{\partial Y_{pred}}) \times (\frac{\partial Y_{pred}}{\partial A_{22}}) \times (\frac{\partial A_{22}}{\partial Z_{22}}) \times (\frac{\partial Z_{22}}{\partial w_{41}})$$

$$= \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times \frac{\partial}{\partial A_{22}}(w_{51}A_{21} + w_{52}A_{22}) \times \frac{\partial}{\partial Z_{22}}(\sigma(Z_{22})) \times \frac{\partial}{\partial w_{41}}(w_{41}A_{11} + w_4$$

$$= \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times w_{52} \times \sigma(Z_{22})[1 - \sigma(Z_{22})] \times A_{11} = -0.007$$

Similarly:

$$\frac{\partial MSE}{\partial w_{42}} = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times w_{52} \times \sigma(Z_{22})[1 - \sigma(Z_{22})] \times A_{12} = -0.018$$

$$\frac{\partial MSE}{\partial b_{4}} = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times w_{52} \times \sigma(Z_{22})[1 - \sigma(Z_{22})] \times 1 = -0.020$$

Multi-Layer Perceptron – Example 2

Step 5: Backpropagation from first hidden layer to input layer

$$\frac{\partial MSE}{\partial w_{11}} = (\frac{\partial MSE}{\partial Y_{pred}}) \times (\frac{\partial Y_{pred}}{\partial A_{21}}) \times (\frac{\partial A_{21}}{\partial Z_{21}}) \times (\frac{\partial Z_{21}}{\partial A_{11}}) \times (\frac{\partial A_{11}}{\partial Z_{11}}) \times (\frac{\partial Z_{11}}{\partial w_{11}})$$

$$\frac{\partial MSE}{\partial w_{12}} = (\frac{\partial MSE}{\partial Y_{pred}}) \times (\frac{\partial Y_{pred}}{\partial A_{21}}) \times (\frac{\partial A_{21}}{\partial Z_{21}}) \times (\frac{\partial Z_{21}}{\partial A_{11}}) \times (\frac{\partial A_{11}}{\partial Z_{11}}) \times (\frac{\partial Z_{11}}{\partial w_{12}})$$

$$\frac{\partial MSE}{\partial w_{11}} = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times w_{51} \times \sigma(Z_{21})[1-\sigma(Z_{21})] \times w_{31} \times \sigma(Z_{11})[1-\sigma(Z_{11})] \times X$$

$$= 0.0019$$

$$\frac{\partial MSE}{\partial w_{12}} = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times w_{51} \times \sigma(Z_{21})[1-\sigma(Z_{21})] \times w_{31} \times \sigma(Z_{11})[1-\sigma(Z_{11})] \times X$$

$$= 0.0019$$

Multi-Layer Perceptron – Example 2

$$\frac{\partial MSE}{\partial w_{21}} = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times w_{52} \times \sigma(Z_{22})[1-\sigma(Z_{22})] \times w_{42} \times \sigma(Z_{12})[1-\sigma(Z_{12})] \times X$$

$$= -0.0021$$

$$\frac{\partial MSE}{\partial w_{22}} = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times w_{52} \times \sigma(Z_{22})[1-\sigma(Z_{22})] \times w_{42} \times \sigma(Z_{12})[1-\sigma(Z_{12})] \times X$$

$$= -0.0021$$

$$\frac{\partial MSE}{\partial b_{1}} = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times w_{51} \times \sigma(Z_{21})[1-\sigma(Z_{21})] \times w_{31} \times \sigma(Z_{11})[1-\sigma(Z_{11})] \times 1$$

$$\frac{\partial MSE}{\partial b_{2}} = \frac{1}{n}\sum_{i=1}^{n}(Y_{pred,i} - Y_{true,i}) \times w_{52} \times \sigma(Z_{22})[1-\sigma(Z_{22})] \times w_{42} \times \sigma(Z_{12})[1-\sigma(Z_{12})] \times 1$$

Multi-Layer Perceptron – Example 2

Step 6: Updating the Parameters

$$w_{11} = w_{11} - \alpha \frac{\partial MSE}{\partial w_{11}} = -0.251$$

$$w_{31} = w_{31} - \alpha \frac{\partial MSE}{\partial w_{31}} = -0.688$$

$$w_{12} = w_{12} - \alpha \frac{\partial MSE}{\partial w_{12}} = 0.464$$

$$w_{32} = w_{32} - \alpha \frac{\partial MSE}{\partial w_{32}} = -0.883$$

$$w_{21} = w_{21} - \alpha \frac{\partial MSE}{\partial w_{21}} = 0.901$$

$$w_{41} = w_{41} - \alpha \frac{\partial MSE}{\partial w_{41}} = -0.687$$

$$w_{22} = w_{22} - \alpha \frac{\partial MSE}{\partial w_{22}} = 0.198$$

$$w_{42} = w_{42} - \alpha \frac{\partial MSE}{\partial w_{42}} = 0.734$$

$$w_{51} = w_{51} - \alpha \frac{\partial MSE}{\partial w_{51}} = 0.210$$

$$w_{52} = w_{52} - \alpha \frac{\partial MSE}{\partial w_{52}} = 0.425$$

$$b_1 = b_1 - \alpha \frac{\partial MSE}{\partial b_1} = -0.959$$

$$b_3 = b_3 - \alpha \frac{\partial MSE}{\partial b_3} = 0.666$$

$$b_2 = b_2 - \alpha \frac{\partial MSE}{\partial b_2} = 0.94$$

$$b_4 = b_4 - \alpha \frac{\partial MSE}{\partial b_4} = -0.573$$

Thank You