DTE-2502: Neural Networks

Module02: Multi-layer Perceptron

Basic definitions

Kalyan Ram Ayyalasomayajula, email: kay001@uit.no

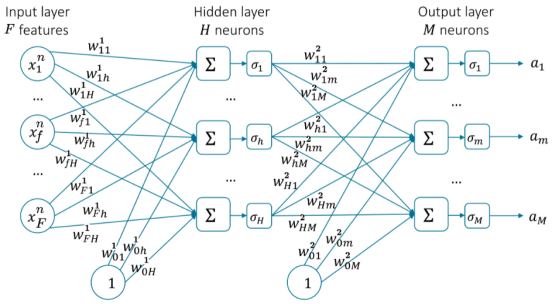
1. Multi-layer Perceptron

A Multi-Layer Perceptron (MLP) is a type of artificial neural network that consists of multiple layers of nodes, or neurons, which can learn to map input data to output classifications or predictions. Here's a breakdown of its components and how it works:

Layers : An MLP typically consists of

- Input Layer : This is where the network receives input data. Each node in this layer represents a feature in the input data.
- Hidden Layers : There can be one or more hidden layers, where the actual processing occurs through weighted connections. Each layer can contain multiple nodes that apply transformations to the input they receive.
- Output Layer : This layer provides the final output, which could be a class label for classification tasks or a value for regression tasks.

$$\text{Input: } \mathbf{x}_n \in X = \{\mathbf{x}_1, \cdots, \mathbf{x}_n, \cdots, \mathbf{x}_N\},$$
$$\mathbf{x}_n = (x_1^n, \cdots, x_f^n, \cdots, x_F^n)$$
$$\text{Output: } \mathbf{y}_n \in Y = \{\mathbf{y}_1, \cdots, \mathbf{y}_n, \cdots, \mathbf{y}_N\},$$
$$\mathbf{y}_n = (y_1^n, \cdots, y_m^n, \cdots, y_M^n)$$
$$\text{Parameters: } W = \Big(\mathbf{w}^1 = \{w_{11}^1, \cdots, w_{fh}^1, \cdots, w_{FH}^1\},$$
$$\mathbf{w}^2 = \{w_{11}^2, \cdots, w_{hm}^2, \cdots, w_{HM}^2\}\Big)$$



2. Working of MLP

- Forward Propagation : When input data is passed through the network, the data flows from the input layer to the hidden layers and finally to the output layer. Each neuron computes a weighted sum of its inputs, applies the activation function, and passes the result to the next layer.
- Loss Calculation : After the forward pass, the output is compared to the actual target values using a loss function (e.g., Mean Squared Error for regression tasks, Cross-Entropy Loss for classification).
- Backpropagation : The loss is then propagated back through the network to update the weights. This involves computing the gradient of the loss with respect to each weight using the chain rule, which helps in optimizing the weights to minimize the loss function. Gradient descent or its variations (like Adam) are usually applied to update the weights.
- Training : This process of forward propagation, loss calculation, and backpropagation is repeated over multiple iterations (epochs) using a training dataset until the model learns the relationship between inputs and outputs sufficiently well.

Backpropagation. Forward phase

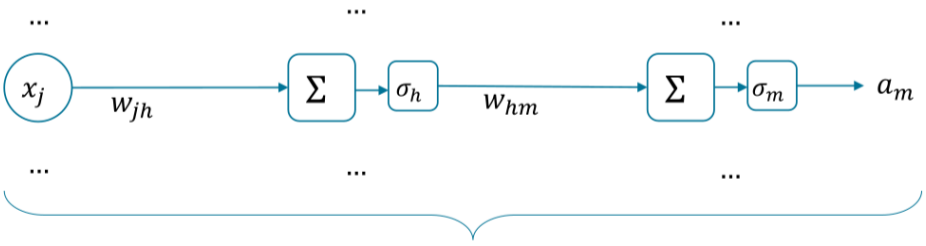
Given inputs $x_i = (x_1, \dots, x_j, \dots x_n)$

and outputs $y_i = y(x_i) = (y_1, \dots, y_m, \dots y_M)$

Let a loss function be

$$\mathcal{L}_i(w) = \frac{1}{2}\sum_{m=1}^{M}\left(a_m(x_i) - y_m(x_i)\right)^2$$

$$u_h(x_i) = \sigma_h\left(\sum_{j=0}^{n} w_{jh}x_j\right)$$



$$a_m(x_i) = \sigma_m\left(\sum_{h=0}^{H} w_{hm}u_h(x_i)\right)$$

Backward phase. Gradient computation

$$\frac{\partial \mathcal{L}_i(w)}{\partial w_{hm}} = \frac{\partial \mathcal{L}_i(w)}{\partial a_m}\frac{\partial a_m}{\partial w_{hm}}$$

$a_m(x_i) - y_m(x_i)$ $\quad \sigma'_m u_h(x_i)$

output layer error

$\varepsilon_m(x_i)$

$$\frac{\partial \mathcal{L}_i(w)}{\partial w_{hm}} = \varepsilon_m(x_i)\ \sigma'_m u_h(x_i),$$
$$m = 1, \dots, M, h = 0, \dots, H$$

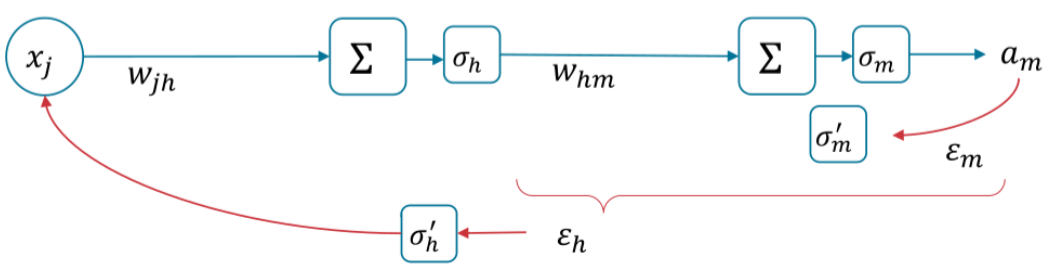
$$\frac{\partial \mathcal{L}_i(w)}{\partial w_{jh}} = \frac{\partial \mathcal{L}_i(w)}{\partial u_h}\frac{\partial u_h}{\partial w_{jh}}$$

$\sum_{m=1}^{M} \varepsilon_m(x_i)\sigma'_m w_{hm}$ $\quad \sigma'_h x_j$

hidden layer error

$\varepsilon_h(x_i)$

$$\frac{\partial \mathcal{L}_i(w)}{\partial w_{jh}} = \varepsilon_h(x_i)\ \sigma'_h x_j,$$
$$h = 1, \dots, H, j = 0, \dots, n$$



Backpropagation algorithm

Input: $X^l = (x_i, y_i)_{i=1}^l \subset \mathbb{R}^n \times \mathbb{R}^M$, parameters H, μ, λ
Output: weights w_{jh}, w_{hm}

Initialize w_{jh}, w_{hm}
do
select x_i from X^l
forward phase:

$$u_h(x_i) = \sigma_h\left(\sum_{j=0}^{n} w_{jh} x_j\right), \quad h = 1, \dots, H$$

$$a_m(x_i) = \sigma_m\left(\sum_{h=0}^{H} w_{hm} u_h(x_i)\right), \quad m = 1, \dots, M$$

$$\varepsilon_m(x_i) = \frac{\partial \mathcal{L}_i(w)}{\partial a_m} = a_m(x_i) - y_m(x_i), \quad m = 1, \dots, M$$

$$Q := \lambda \mathcal{L}_i + (1 - \lambda) Q$$

backward phase:

$$\varepsilon_h(x_i) = \sum_{m=1}^{M} \varepsilon_m(x_i) \sigma_m' w_{hm}, \quad h = 1, \dots, H$$

gradient step:

$$w_{hm} = w_{hm} - \mu \varepsilon_m(x_i) \sigma_m' u_h(x_i), \quad h = 0, \dots, H, \ m = 1, \dots, M$$

$$w_{jh} = w_{jh} - \mu \varepsilon_h(x_i) \sigma_h' x_j, \quad j = 0, \dots, n, \ h = 1, \dots, H$$

until Q converges

3. Loss functions

Regression loss functions

- Mean squared error (MSE) loss

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(a_i - y_i)^2$$

- Mean squared logarithmic error (MSLE) loss

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(\log(y_i + 1) - \log(a_i + 1))^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\log\left(\frac{y_i + 1}{a_i + 1}\right)\right)^2$$

- Mean absolute error (MAE) loss

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}|a_i - y_i|$$

Classification loss functions

- Categorical crossentropy loss

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \log a_i$$

- Binary crossentropy loss

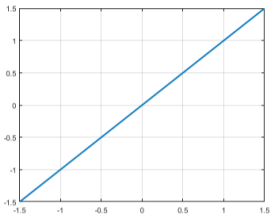
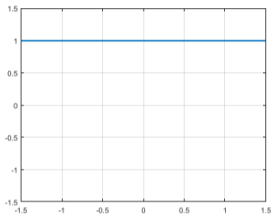
$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log a_i + (1 - y_i)\log(1 - a_i)$$

- Squared hinge loss

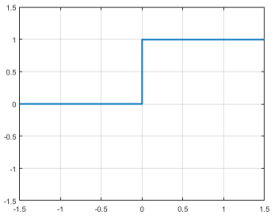
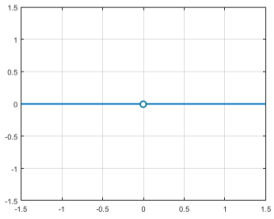
$$\mathcal{L} = \sum_{i=1}^{N}(\max\{0, 1 - y_i a_i\}^2)$$

4. Activation functions

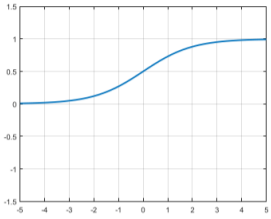
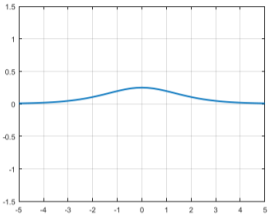
Identity $\sigma(x) = x$ $\sigma'(x) = 1$

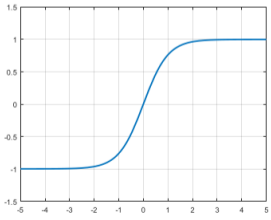
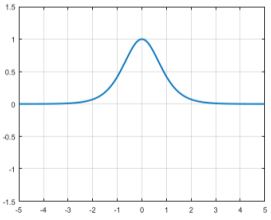
Sign $\sigma(x) = \text{sign}(x)$ $\sigma'(x) = 0$ everywhere except $x = 0$

Sigmoid $\sigma(x) = \dfrac{1}{1+e^{-x}}$ $\sigma'(x) = \dfrac{e^{-x}}{(1+e^{-x})^2} = \sigma(1-\sigma)$

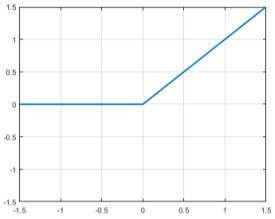



tanh $\sigma(x) = \dfrac{e^{2x}-1}{e^{2x}+1}$ $\sigma'(x) = \dfrac{4e^{2x}}{(e^{2x}+1)^2} = 1-\sigma^2$

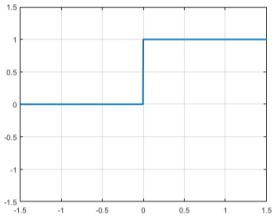



ReLU
Rectified Linear Unit

$$\sigma(x) = \max\{x, 0\}$$

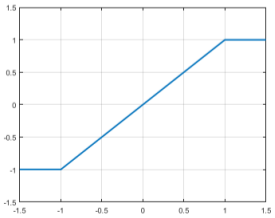


$$\sigma'(x) = \text{sign}(x)$$



Hard tanh

$$\sigma(x) = \max\{\min\{x, 1\}, -1\}$$



$$\sigma'(x) = \begin{cases} 1, & x \in [-1,1] \\ 0, & \text{otherwise} \end{cases}$$