Machine Learning

Neural Networks

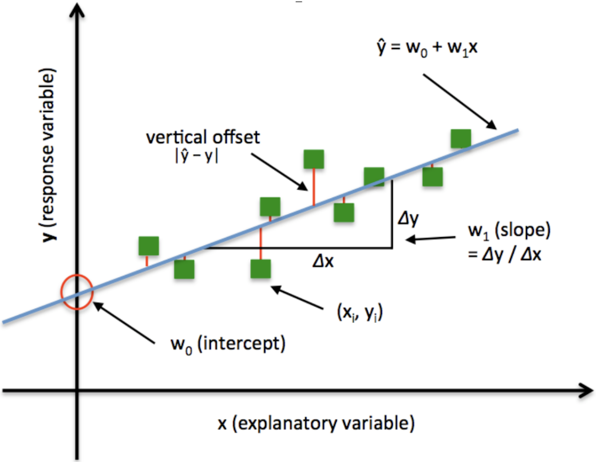
Kien C Nguyen

September 18, 2024

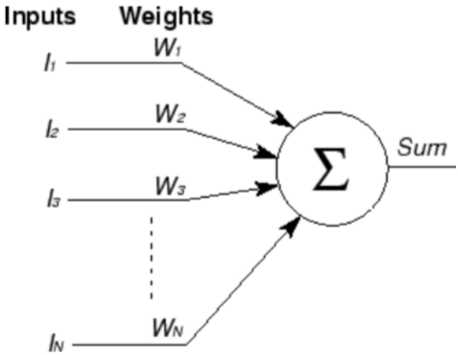
Contents

Linear Regression Revisit

- Recall that in Linear Regression, we fit a hyperplane through the input points
- $\hat{y} = \mathbf{w}^T \mathbf{x}$



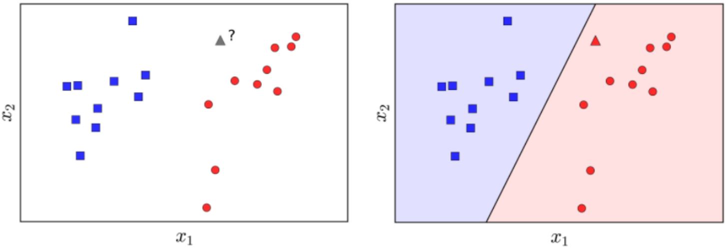
A Linear Regression Model can be represented by the following diagram



Perceptron Learning Algorithm

- Given: training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$, where $x_i \in \mathbf{R}^d$ and $y_i \in \{-1, +1\}$
- Goal: find a $d-1$ dimensional **hyperplane** (i.e decision boundary) H which separates the $+1's$ from the $-1's$

Figure: Find a $d-1$ dimensional **hyperplane** (i.e decision boundary) H which separates the $+1's$ from the $-1's$



Perceptron Learning Algorithm

1. PLA is a binary classification algorithm
2. Mathematically, the goal is to learn a weight $w \in \mathbf{R}^d$ that satisfies the linear separability constrains:

$$\begin{cases} w^T x_i \geq 0 & \text{if } y_i = 1 \\ w^T x_i \leq 0 & \text{if } y_i = -1 \end{cases} \tag{1}$$

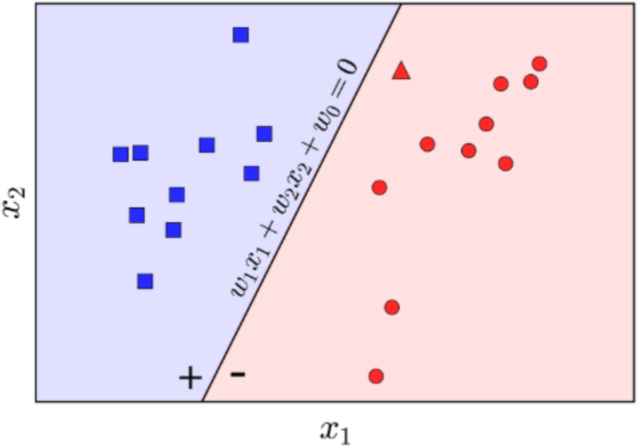
   Equivalently,

$$\forall i, y_i(w^T x_i) \geq 0$$

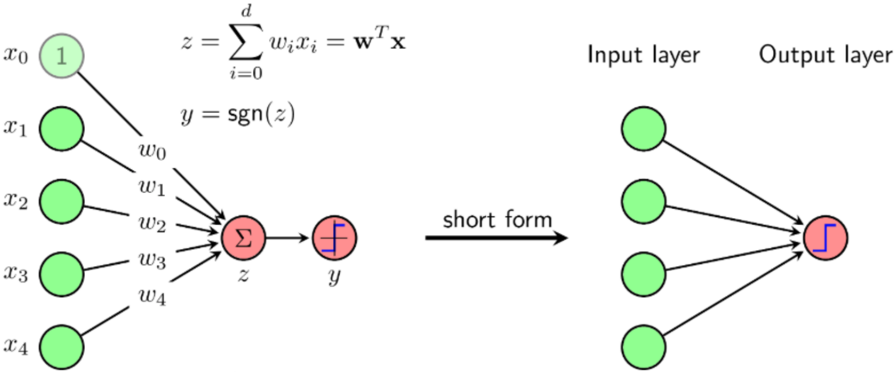
3. The resulting decision boundary is a hyperplane $H = \{x : w^T x = 0\}$

Peceptron Learning Algorithm

- We use a sign function to classify new datapoints
- $\hat{y} = sgn(\mathbf{w}^T\mathbf{x})$



Peceptron Learning Algorithm as a Neural Network

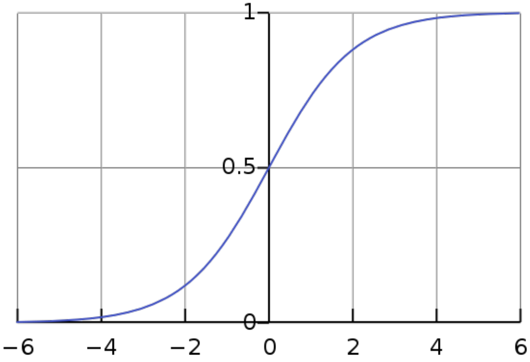


Hình 5: Biểu diễn của Perceptron dưới dạng Neural Network.

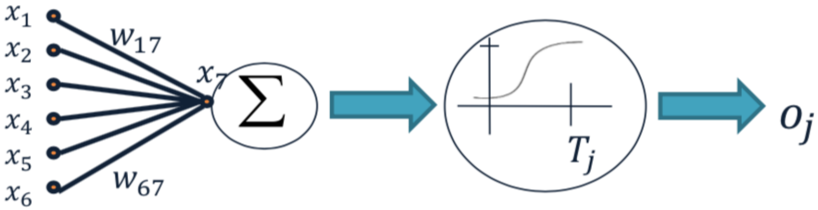
Logistic Regression Revisit

- Use a function $\sigma(\mathbf{w}^T\mathbf{x})$
- $0 \leq \sigma(\mathbf{w}^T\mathbf{x}) \leq 1$
- Sigmoid function (Logistic function)

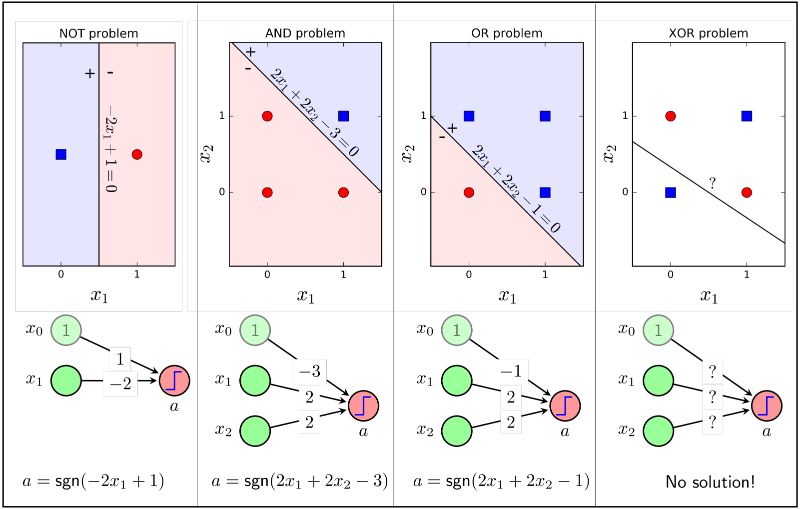
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

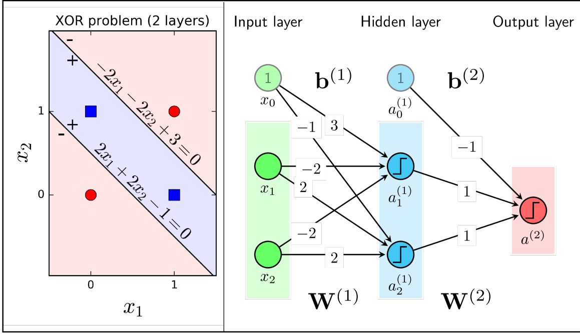


Logistic Regression as a Neural Network



PLA for some simple logical functions





$$\mathbf{W}^{(1)} = \begin{bmatrix} -2 & 2 \\ -2 & 2 \end{bmatrix}; \quad \mathbf{b}^{(1)} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$$

$$\mathbf{W}^{(2)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \quad \mathbf{b}^{(2)} = \begin{bmatrix} -1 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}; \mathbf{a}^{(1)} = \begin{bmatrix} a_1^{(1)} \\ a_2^{(1)} \end{bmatrix}$$

$$\mathbf{a}^{(1)} = f\left(\mathbf{W}^{(1)T}\mathbf{x} + \mathbf{b}^{(1)}\right)$$

$$a^{(2)} = f\left(\mathbf{W}^{(2)T}\mathbf{a}^{(1)} + \mathbf{b}^{(2)}\right)$$

$f(.) = \text{sgn}(.)$ (element-wise)

Notes

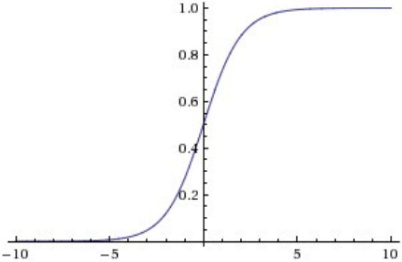
- PLA is an example of a single-layer neural network where the activation function is a sign function (sgn).
- Activation functions can be other nonlinear functions such as sigmoid function or tanh function.

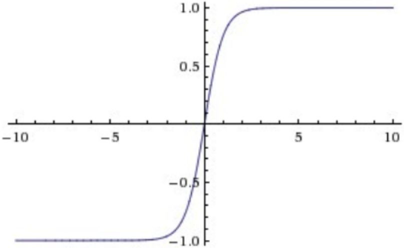
$$tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}} = 2\sigma(2s) - 1$$

- Activation functions must be nonlinear; otherwise we can collapse the multi-layer perceptron to have a linear network.

Sigmoid function and *tanh* function

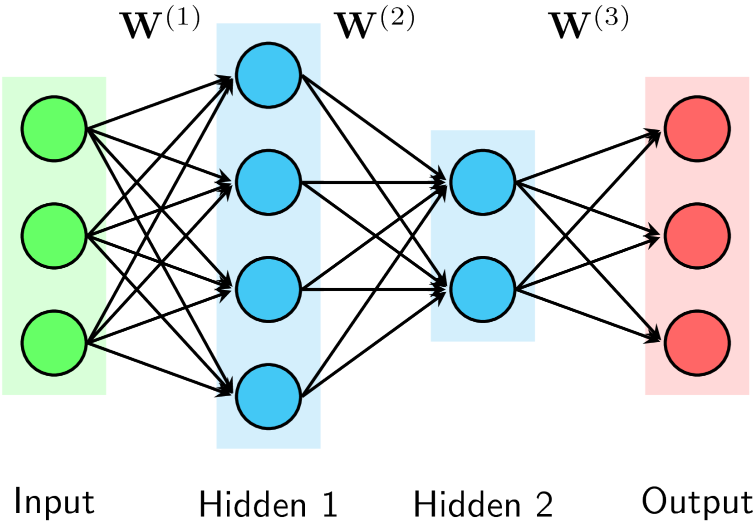
Figure: Left: *sigmoid* function, right: *tanh* function



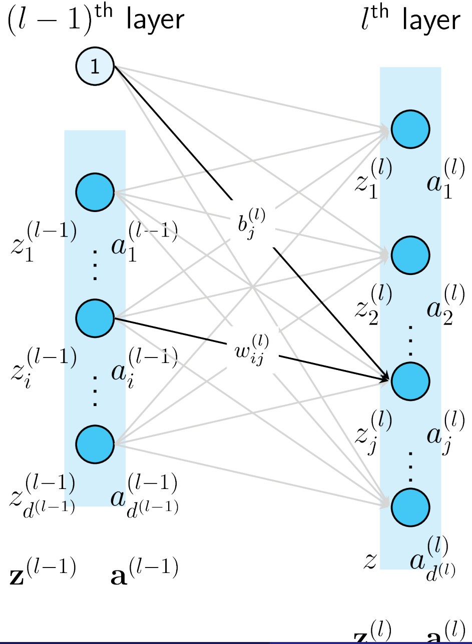


Contents

Multi-layer Perceptrons



Multi-layer Perceptrons – Notations



$$\mathbf{W}^{(l)} \in \mathbb{R}^{d^{(l-1)} \times d^{(l)}}$$

$$\mathbf{b}^{(l)} \in \mathbb{R}^{d^{(l)} \times 1}$$

$$z_j^{(l)} = \mathbf{w}_j^{(l)T} \mathbf{a}^{(l-1)} + b_j^{(l)}$$

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)T} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$

$$\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)})$$

- We use Gradient Descent algorithm to minimize the loss function L.
- We have to calculate Gradient of L with respect to $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$.

Feedfoward steps:

$$
\begin{aligned}
\mathbf{a}^{(0)} &= \mathbf{x} \\
z_i^{(l)} &= \mathbf{w}_i^{(l)T}\mathbf{a}^{(l-1)} + b_i^{(l)} \\
\mathbf{z}^{(l)} &= \mathbf{W}^{(l)T}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}, \quad l = 1, 2, \ldots, L \\
\mathbf{a}^{(l)} &= f(\mathbf{z}^{(l)}), \quad l = 1, 2, \ldots, L \\
\hat{\mathbf{y}} &= \mathbf{a}^{(L)}
\end{aligned}
$$

Backpropagation – Gradient Descent

- Let $J(\mathbf{W}, \mathbf{b}, \mathbf{X}, \mathbf{Y})$ be the loss function.
- We have to calculate $\frac{\partial J}{\partial \mathbf{W}^{(l)}}; \frac{\partial J}{\partial \mathbf{b}^{(l)}}, \quad l = 1, 2, \ldots, L.$

$$\begin{aligned} J(\mathbf{W}, \mathbf{b}, \mathbf{X}, \mathbf{Y}) &= \frac{1}{N} \sum_{n=1}^{N} ||\mathbf{y}_n - \hat{\mathbf{y}}_n||_2^2 \\ &= \frac{1}{N} \sum_{n=1}^{N} ||\mathbf{y}_n - \mathbf{a}_n^{(L)}||_2^2 \end{aligned}$$

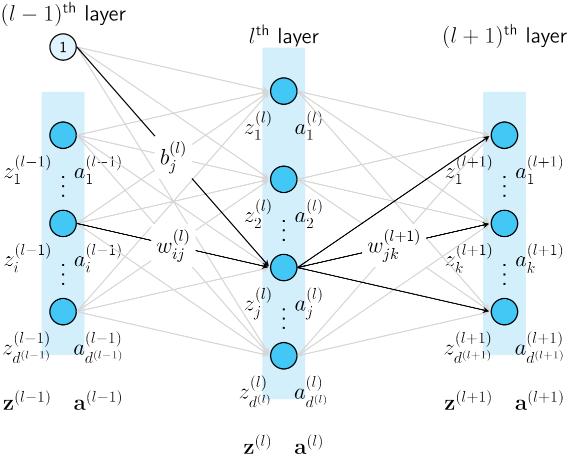
Backpropagation – Gradient Descent

- In backpropagation, we calculate the gradients from the last layer back to the first layer.
- Using chain rule

$$\begin{aligned}\frac{\partial J}{\partial w_{ij}^{(L)}} &= \frac{\partial J}{\partial z_j^{(L)}} \cdot \frac{\partial z_j^{(L)}}{\partial w_{ij}^{(L)}} \\ &= e_j^{(L)} a_i^{(L-1)}\end{aligned}$$

- Gradient of J w.r.t. the bias of the last layer

$$\frac{\partial J}{\partial b_j^{(L)}} = \frac{\partial J}{\partial z_j^{(L)}} \cdot \frac{\partial z_j^{(L)}}{\partial b_j^{(L)}} = e_j^{(L)}$$



$$\mathbf{W}^{(l)} \in \mathbb{R}^{d^{(l-1)} \times d^{(l)}}$$

$$\mathbf{b}^{(l)} \in \mathbb{R}^{d^{(l)} \times 1}$$

$$z_j^{(l)} = \mathbf{w}_j^{(l)T} \mathbf{a}^{(l-1)} + b_j^{(l)}$$

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)T} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$

$$\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)})$$

Similarly, for layer l, we have that

$$\begin{aligned}\frac{\partial J}{\partial w_{ij}^{(l)}} &= \frac{\partial J}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}} \\ &= e_j^{(l)} a_i^{(l-1)}\end{aligned}$$

where

$$
\begin{aligned}
e_j^{(l)} &= \frac{\partial J}{\partial z_j^{(l)}} = \frac{\partial J}{\partial a_j^{(l)}} \cdot \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \\
&= \left(\sum_{k=1}^{d^{(l+1)}} \frac{\partial J}{\partial z_k^{(l+1)}} \cdot \frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}} \right) f'(z_j^{(l)}) \\
&= \left(\sum_{k=1}^{d^{(l+1)}} e_k^{(l+1)} w_{jk}^{(l+1)} \right) f'(z_j^{(l)}) \\
&= \left(\mathbf{w}_{j:}^{(l+1)} \mathbf{e}^{(l+1)} \right) f'(z_j^{(l)})
\end{aligned}
$$

where $\mathbf{e}^{(l+1)} = [e_1^{(l+1)}, e_2^{(l+1)}, \ldots, e_{d^{(l+1)}}^{(l+1)}]^T \in \mathbb{R}^{d^{(l+1)} \times 1}$ and $\mathbf{w}_{j:}^{(l+1)}$ is the j^{th} row of $\mathbf{W}^{(l+1)}$.

Contents

References

[1] Vu Huu Tiep – Machine Learning co ban,
https://machinelearningcoban.com/2017/02/24/mlp/
[2] UIUC CS 446 Machine Learning
[3] Andrew Ng – Coursera Machine Learning