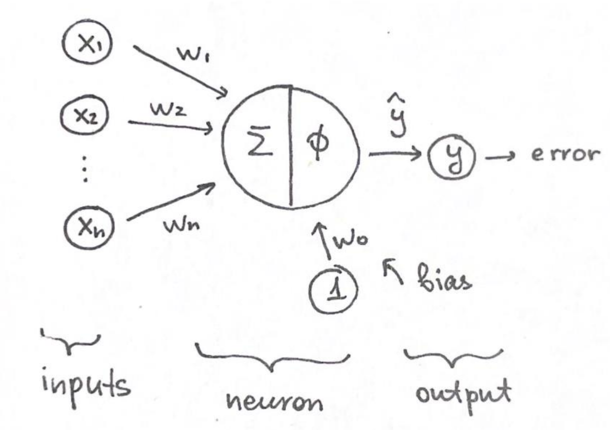
# REPORT ABOUT NEURAL NETWORKS

## 1. PERCEPTRON

### *1.1. Architecture*




- $\Sigma$ is a linear combination: $z = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$.
- ϕ is an activation function.
- There are $m$ input values.
- Output is binary.
- $w_1, \dots, w_m$ – weights.

### *1.2. Vector Representation.*

- Inputs are a vector $X$ – vector of features.

$$X = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}_{n+1\times 1}, x_i \in \mathbb{R}\ \forall i = 1, \dots, n.$$

- $W$ is a vector of weights.

$$W = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_n \end{bmatrix}_{n+1\times 1}$$

- $\hat{y}$ is predicted value (binary output) and $y$ is real value.

Using these vectors, we have:

$$z = X^T W.$$

### *1.3. Activation Function*

$$\phi(z) = \begin{cases} 1, z \geq 0 \\ -1, z < 0 \end{cases}.$$

### *1.4. Predictions*

$$\hat{y} = \phi(z), \qquad z = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

### *1.5. Loss Function*

We calculate the error by counting wrong predictions.

$$L = \sum_{i=1}^{n} \delta(y_i \neq \hat{y}_i)$$

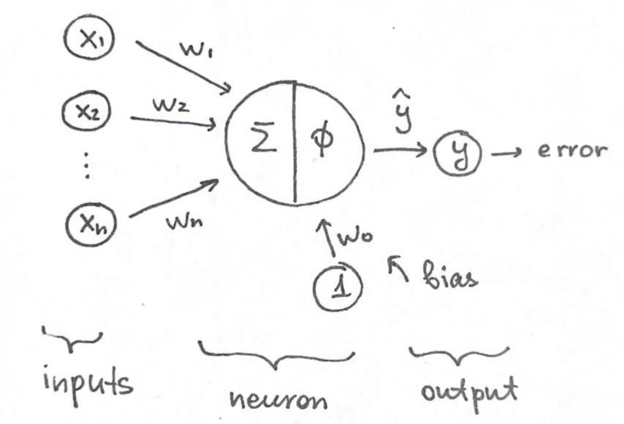
### *1.6. Weights Update*

$$w_i^{(t+1)} = w_i^{(t)} + \eta(y - \hat{y})x_i$$

$$w_o^{(t+1)} = w_0^{(t)} + \eta(y - \hat{y})$$

## 2. LOGISTIC REGRESSION

### *2.1. Architecture*




- $\Sigma$ is a linear combination: $z = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$.
- ϕ is an activation function.
- There are $m$ input values.
- Output is not always binary.
- $w_1, \ldots, w_n$ – weights.

### *2.2. Vector Representation.*

- Inputs are a vector $X$ – vector of features.

$$X = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix}_{n+1\times 1}, x_i \in \mathbb{R}\ \forall i = 1, \ldots, n.$$

- $W$ is a vector of weights.

$$W = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \cdots \\ w_n \end{bmatrix}_{n+1\times 1}$$

- $\hat{y}$ is predicted value (output) and $y$ is real value.

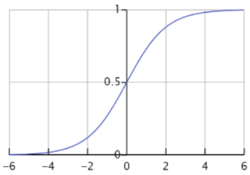
Using these vectors, we have:

$$z = X^T W.$$

### *2.3. Activation function*

$\phi(z) = \frac{1}{1+e^{-z}}$ – sigmoid function.

Graphic of sigmoid function:




### *2.4. Prediction*

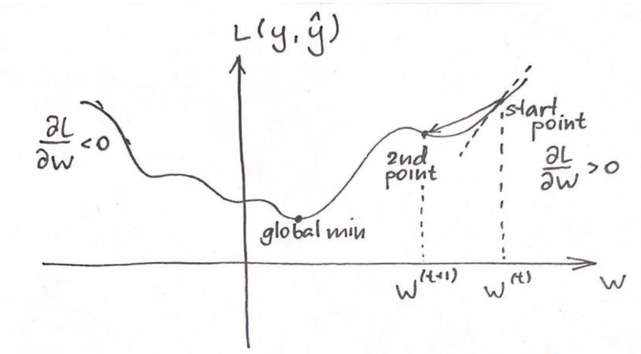
$$\hat{y} = \phi(\mathrm{z}).$$

### *2.5. Calculating Errors*

$error = L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y \log(\hat{y}) + (1-y)\log(1-\hat{y})]$ - cross entropy loss.

### *2.6. Gradient Descendant*

For example, we have this loss function:




Gradient is a vector of deviates:

$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \cdots \\ \frac{\partial L}{\partial w_n} \end{bmatrix}$$

Generical updating rule:

$$W^{(t+1)} = W^t - \eta \frac{\partial L}{\partial W}$$

Where $\eta$ is learning rate.

$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_i}$ – rule of chain.

$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y \log(\hat{y}) + (1-y)\log(1-\hat{y})]$.

$\hat{y} = \phi(z) = \frac{1}{1+e^{-z}}$.

$z = w_0 + x_1 w_1 + \cdots + x_n w_n$.

1) $\frac{\partial L}{\partial \hat{y}} = -\left[y\frac{1}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right] = -\left[\frac{y(1-\hat{y})-(1-y)\hat{y}}{\hat{y}(1-\hat{y})}\right] = -\frac{y-\hat{y}}{\hat{y}(1-\hat{y})}$

2) $\frac{\partial \hat{y}}{\partial z} = -(1+e^{-z})^{-2}(-e^{-z}) = \frac{e^{-z}}{(1+e^{-z})^2} = \left(\frac{1}{1+e^{-z}}\right)\left(\frac{e^{-z}}{1+e^{-z}}\right) = \hat{y}(1-\hat{y}).$

3) $\frac{\partial z}{\partial w_i} = x_i.$

$\frac{\partial L}{\partial w_i} = -\frac{y-\hat{y}}{\hat{y}(1-\hat{y})}\hat{y}(1-\hat{y})x_i = -(y-\hat{y})x_i.$

Generical update rule for components:

$$w_i^{(t+1)} = w_i^{(t)} - \eta \frac{\partial L}{\partial w_i}$$

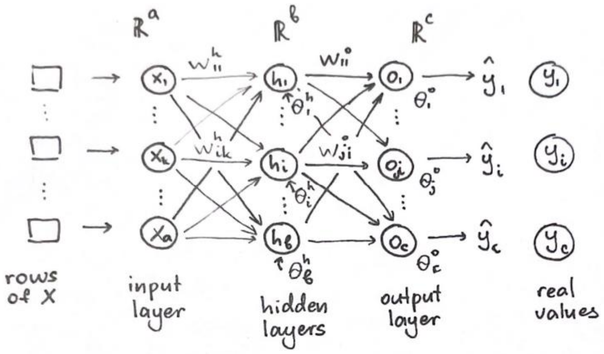
$$w_i^{(t+1)} = w_i^{(t)} - \eta[-(y-\hat{y})x_i] = w_i^{(t)} + \eta(y-\hat{y})x_i$$

If $i = 0$ (bias):

$$\frac{\partial z}{\partial w_i} = 1 \Rightarrow w_0^{(t+1)} = w_i^{(t)} + \eta(y-\hat{y})$$

## 3. MULTILAYER PERCEPTRON

### *3.1. Architecture*




$x_1, \dots, x_a$ – input values.

$h_1, \dots, h_b$ – hidden layer.

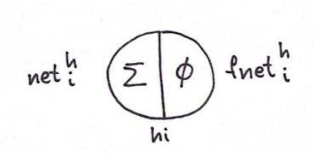
$o_1, \dots, o_c$ – output layer.

$\hat{y}_1, \dots, \hat{y}_c$ – predicted values.

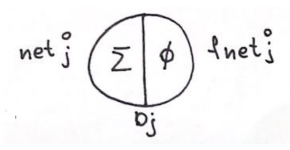
$\theta_1^h, \dots, \theta_b^h$ - biases for hidden layers.

$\theta_1^o, \dots, \theta_c^o$ - biases for output layer.

Hidden layer



Output layer



$net_i^h, net_j^o$ - results of linear combination.

$fnet_i^h, fnet_j^o$ - results of activation function to the linear combination.

### *3.2. Vector Representation*

- Input layer:

$$X = (x_1, \dots, x_k, \dots, x_a)_{1\times a}$$

- Hidden layer:

$$A = (h_1, \dots, h_i, \dots, h_b)_{1\times b}$$

Weights for hidden layer:

$$W^h = \begin{bmatrix} w_{11}^h & \cdots & w_{1k}^h & \cdots & w_{1a}^h \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ w_{i1}^h & \cdots & w_{ik}^h & \cdots & w_{ia}^h \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ w_{b1}^h & \cdots & w_{bk}^h & \cdots & w_{ba}^h \end{bmatrix}_{b\times a}$$

Biases for hidden layer:

$$\theta^h = \begin{bmatrix} \theta_1^h \\ \vdots \\ \theta_i^h \\ \vdots \\ \theta_b^h \end{bmatrix}_{b\times 1}$$

- Output layer:

$$O = \left(o_1, \dots, o_j, \dots, o_c\right)_{1\times c}$$

Weights for output layer:

$$W^o = \begin{bmatrix} w_{11}^o & \cdots & w_{1i}^o & \cdots & w_{1b}^o \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ w_{j1}^o & \cdots & w_{ji}^o & \cdots & w_{jb}^0 \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ w_{c1}^o & \cdots & w_{ci}^o & \cdots & w_{cb}^o \end{bmatrix}_{c\times b}$$

Biases for output layer:

$$\theta^o = \begin{bmatrix} \theta_1^o \\ \vdots \\ \theta_j^o \\ \vdots \\ \theta_c^o \end{bmatrix}_{c\times 1}$$

### *3.3. Activation Function*

$$\phi(z) = \frac{1}{1+e^{-z}} \text{ – sigmoid function.}$$

### *3.4. Predictions*

$$\hat{y} = \phi(\mathrm{z}).$$

### *3.5. Calculating Errors*

- Step forward
  - Hidden layers:

$$net_i^h = \sum_{k=1}^{a} x_k w_{ik}^h + \theta_i^h$$

$$fnet_i^h = \phi(net_i^h)$$

- Output layer:

$$net_i^o = \sum_{i=1}^{b} fnet_i^h w_{ji}^o + \theta_j^h$$

$$fnet_j^o = \phi(net_j^o)$$

As a result, we get probabilities of each class.

### *3.6. Gradient Descendant*

Back propagation

Loss function:

$$L = min\left\{\frac{1}{2}\sum_{j=1}^{c}\left(y_j - \hat{y}_j\right)^2\right\}$$

- Backward of outputs

Generalized delta rule:

$$w_{ji}^{o(t+1)} = w_{ji}^{o(t)} - \eta\frac{\partial L}{\partial W_{ji}}$$

$\eta\frac{\partial L}{\partial W_{ji}} = \delta_j^o * fnet_j^h$ - learning rate.

$$\theta_j^{o(t+1)} = \theta_j^{o(t)} - \eta\frac{\partial L}{\partial \theta_j^o}$$

$$\eta\frac{\partial L}{\partial \theta_j^o} = \delta_j^o * fnet_j^h = \delta_j^o$$

$$\delta_j^o = -(y - \hat{y})dfnet_j^o$$

- Backward of hiddens

Generalized delta rule:

$$w_{ik}^{h(t+1)} = w_{ik}^{h(t)} - \eta\frac{\partial L}{\partial W_{ik}}$$

$$\eta\frac{\partial L}{\partial W_{ik}} = \delta_j^h * x_k$$

$$\theta_i^{h(t+1)} = \theta_i^{o(t)} - \eta\frac{\partial L}{\partial \theta_i^o}$$

$$\eta\frac{\partial L}{\partial \theta_i^o} = \delta_j^h$$

$$\delta_i^h = dnet_i^h \sum_{j=1}^{c} \delta_j^o w_{ji}^o$$