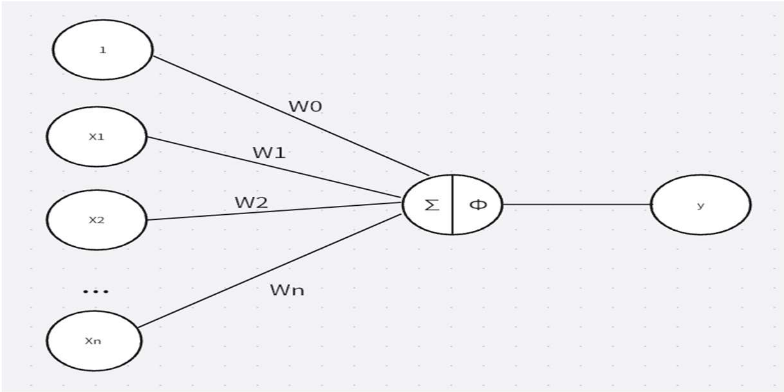
**Katsuba Danila 5130203/20101**

# Logistic Regression

1. Draw of model architecture:

   A single-layer network with sigmoid activation for binary classification.




2. Vector representation of data:

   Input vector X $= \begin{bmatrix} 1 \\ X_1 \\ \dots \\ X_n \end{bmatrix}$; Output $y \in [0, 1]$.

3. Mathematical formulation of linear combination, activation function and loss function:

$$\sum = W_1X_1 + W_2X_2 + \cdots + W_nX_n + W_0 = [W_0 \quad W_1 \quad \dots \quad W_n] \begin{bmatrix} 1 \\ X_1 \\ \dots \\ X_n \end{bmatrix} = W^TX = z$$

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

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

4. Calculating predictions:

$$\hat{y} = \Phi(\sum(\underbrace{X, W}_{z}))$$

5. Gradient descendent algorithm:

   Gradient Descent is an optimization algorithm for finding a local minimum of a differentiable function. Gradient descent in machine learning is simply used to find the values of a function's parameters (coefficients) that minimize a cost function as far as possible.

6. Formulas of gradients and weights/biases updates:

$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial W_1} \\ \dots \\ \frac{\partial L}{\partial W_n} \end{bmatrix}$$

$$Generical\ Update\ Rule: W_i^{(t+1)} = W_i^{(t)} - \mu * \frac{\partial L}{\partial W_i}$$

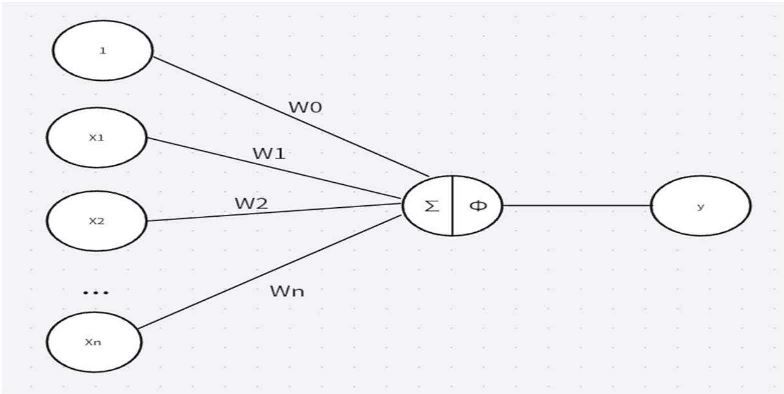
$$W_i^{(t+1)} = W_i^{(t)} + \mu(y - \hat{y})X_i$$

$$W_0^{(t+1)} = W_0^{(t)} + \mu(y - \hat{y})$$

**Perceptron**

1. Draw of model architecture:

   A single-layer neural network with one output node.




2. Vector representation of data:

   Input vector X = $\begin{bmatrix} 1 \\ X_1 \\ \dots \\ X_n \end{bmatrix}$; Output $y \in [0, 1]$.

3. Mathematical formulation of linear combination, activation function and loss function:

$$\sum = W_1X_1 + W_2X_2 + \cdots + W_nX_n + W_0 = [W_0 \quad W_1 \quad \dots \quad W_n] \begin{bmatrix} 1 \\ X_1 \\ \dots \\ X_n \end{bmatrix} = W^TX = z$$

$$\Phi(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

$$L = \max(0, -y * z)$$

4. Calculating predictions:

$$\hat{y} = \Phi(\underbrace{\sum(X, W)}_{z})$$

5. Gradient descendent algorithm:

   Not explicitly used in the classical Perceptron.

6. Formulas of gradients and weights/biases updates:

$$Generical\ Update\ Rule: W_i^{(t+1)} = W_i^{(t)} - \mu * \frac{\partial L}{\partial W_i}$$

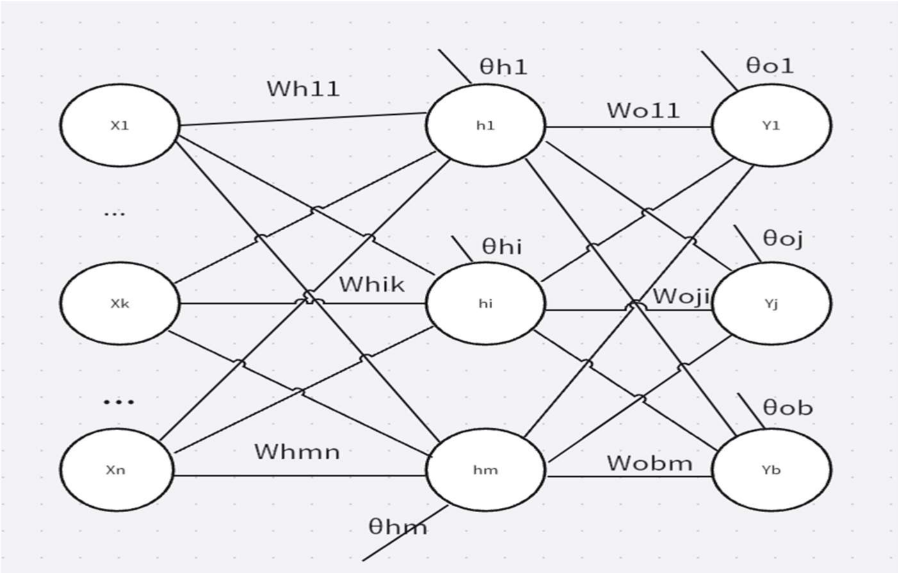
$$W_i^{(t+1)} = W_i^{(t)} + \mu(y - \hat{y})X_i$$

$$W_0^{(t+1)} = W_0^{(t)} + \mu(y - \hat{y})$$

**Multilayer perceptron**

1. Draw of model architecture:

   A network with one or more hidden layers.




2. Vector representation of data:

   Input vector: $X = \begin{bmatrix} X_1 \\ \dots \\ X_n \end{bmatrix}$; Hidden layer: $H = \begin{bmatrix} h_1 \\ \dots \\ h_m \end{bmatrix}$; Output vector: $Y = \begin{bmatrix} Y_1 \\ \dots \\ Y_b \end{bmatrix}$

   Weights: $W^h = \begin{bmatrix} W_{11}^h & \dots & W_{1n}^h \\ \dots & W_{ik}^h & \dots \\ W_{m1}^h & \dots & W_{mn}^h \end{bmatrix}$; $W^o = \begin{bmatrix} W_{11}^o & \dots & W_{1n}^o \\ \dots & W_{ik}^o & \dots \\ W_{m1}^o & \dots & W_{mn}^o \end{bmatrix}$

   Biases: $\theta^h = \begin{bmatrix} \theta_1^h \\ \dots \\ \theta_m^h \end{bmatrix}$; $\theta^o = \begin{bmatrix} \theta_1^o \\ \dots \\ \theta_b^o \end{bmatrix}$

3. Mathematical formulation of linear combination, activation function and loss function:

   Linear combinations: $net_i^h = \sum_{k=1}^{n}\left(X_k * W_{ik}^h\right) + \theta_i^h$; $net_j^o = \sum_{i=1}^{m}\left(fnet_i^h * W_{ji}^o\right) + \theta_j^o$

   Activation functions: $fnet_i^h = \Phi\left(net_i^h\right) = \frac{1}{1+e^{-ne_i^h}}$; $fnet_j^o = \Phi\left(net_j^o\right) = \frac{1}{1+e^{-net_j^o}}$

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

4. Calculating predictions:

$$\hat{y} = fnet_i^h = \Phi\left(net_i^h\right) = \frac{1}{1 + e^{-ne_i^h}}$$

5. Gradient descendent algorithm:

   Not explicitly used in the classical Perceptron.

6. Formulas of gradients and weights/biases updates:

$$Generalised\ Delta\ Rule: W_{ji}^{0(t+1)} = W_{ji}^{o(t)} - \mu \underbrace{\frac{\partial L}{\partial W_{ji}^o}}_{\delta_j^o fnet_j^h}; \theta_j^{o(t+1)} = \theta_j^{o(t)} - \mu \underbrace{\frac{\partial L}{\partial \theta_j^o}}_{\delta_j^o * 1};$$

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

$$Generalised\ Delta\ Rule: W_{ik}^{h(t+1)} = W_{ik}^{h(t)} - \underbrace{\mu \frac{\partial L}{\partial W_{ik}^{h}}}_{\delta_i^h x_k}; \theta_i^{h(t+1)} = \theta_i^{h(t)} - \underbrace{\mu \frac{\partial L}{\partial \theta_i^{h}}}_{\delta_i^h * 1};$$

$$\partial_i^h = dfnet_i^h * \sum_{j=1}^{b} \delta_j^o W_{ji}^o$$