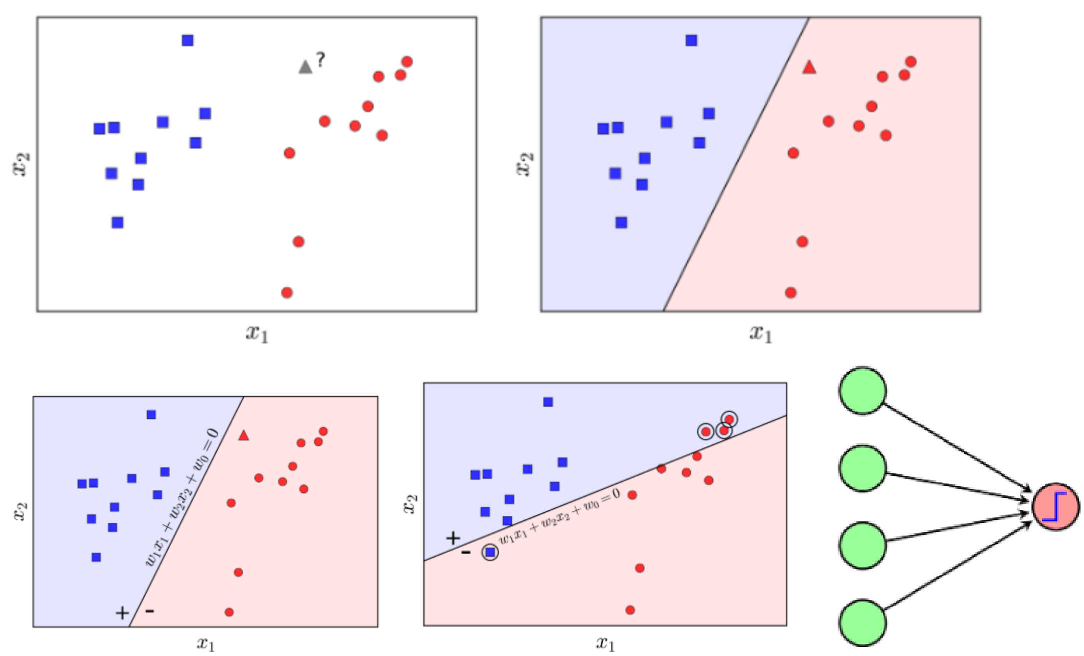
NGUYEN BA PHU 513203/20101

Report about Neural Networks with the mathematical formulation of algorithms Perceptron, Logistic Regression and Multilayer Perceptron.

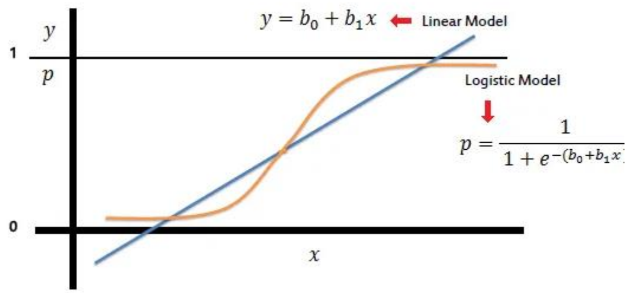
In this paper, I will briefly introduce the mathematical formulation of three algorithms: Perceptron, Logistic Regression and Multilayer Perceptron.

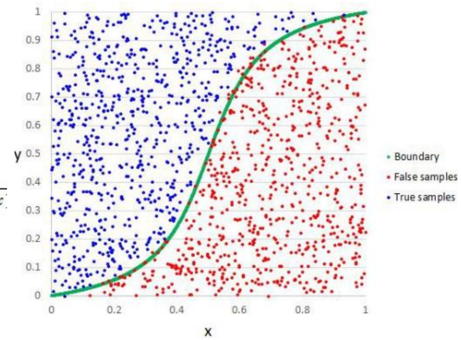
1. Perceptron algorithm.
   A perceptron is a simple neural model that mimics the way a biological neuron works. It takes in a number of inputs, calculates a weighted sum of those inputs, and then applies an activation function to decide what the output should be.
   Model architecture: A simple layer of neurons, with input x, weights w, bias b, and output y.
   Input vector (x): A vector x=[x1,x2,...,xn] contains features.
   Output (y): y∈{−1,1}.
   Linear combination: $z = w^T x + b$
   Step function (sign): $y_{\text{hat}} = \text{sign}(z)$
   Hinge loss function: $L = \max(0, -y \cdot z)$
   Predictive neural networks: $\hat{y} = \text{sign}(w^T x + b)$



2. Logistic Regression algorithm.
   Logistic Regression algorithm is a supervised learning method used to solve binary classification problem. Despite the name "Regression", Logistic Regression is actually a classification model, not a regression like Linear Regression.

Model architecture: A single layer with output processed through a sigmoid function to predict probabilities.
Input vector (x): A vector x=[x1,x2,...,xn]
Out put (y): y∈[0,1].





Linear combination: $$z = w^T x + b$$

Sigmoid function: $$y_{\text{hat}} = \frac{1}{1 + e^{-z}}$$

Cross-entropy function: $$L = -\left[y \log(y_{\text{hat}}) + (1 - y)\log(1 - y_{\text{hat}})\right]$$

Predictive neural networks: $$\hat{y} = \frac{1}{1 + e^{-(w^T x + b)}}$$

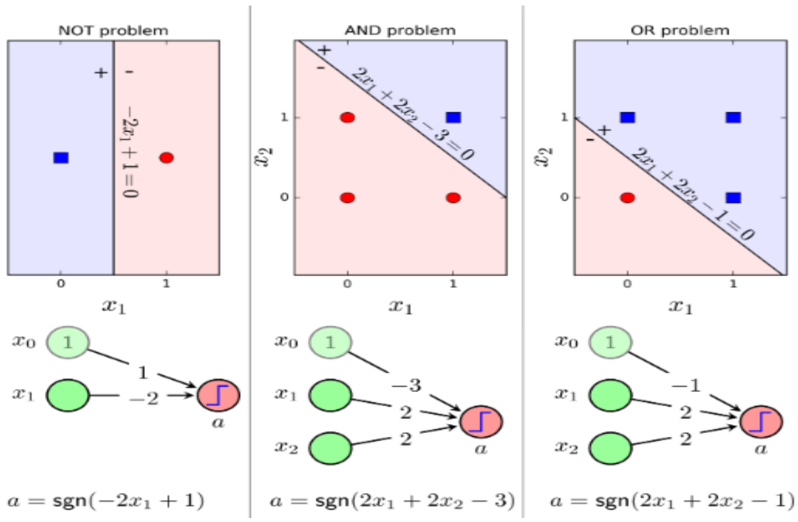
Weighted gradient:

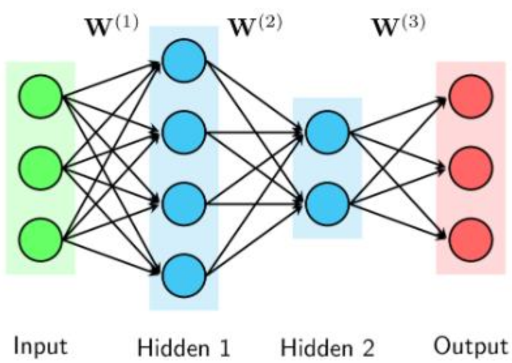
$$\frac{\partial L}{\partial w} = \frac{1}{n}\sum_{i=1}^{n}(y_{\text{hat}} - y)x$$

Gradient offset:

$$\frac{\partial L}{\partial b} = \frac{1}{n}\sum_{i=1}^{n}(y_{\text{hat}} - y)$$

3. Multilayer Perceptron algorithm.
   Multilayer Perceptron (MLP) is a type of artificial neural network consisting of multiple layers of perceptrons. It is capable of learning complex relationships and solving problems that a single Perceptron cannot.
   Model architecture: Input Layer, Hidden Layer with nonlinear activation functions such as ReLU, sigmoid, Output Layer, making predictions.
   Input (x): A vector x=[x1,x2,...,xn] containing features.
   Output (y): y can be a classification label or a regression value.





Linear combination: $z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}$

Common activation functions:

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

ReLU: $f(z) = \max(0, z)$.

Regression: Mean Squared Error (MSE):

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Predictive neural networks (with L layers): $z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}$, $a^{(l)} = \text{activation}(z^{(l)})$ => $\hat{y} = a^{(L)}$.

Gradients propagate back through layers: $$\delta^{(L)} = \frac{\partial L}{\partial a^{(L)}} \cdot \text{activation}'(z^{(L)})$$

$$\delta^{(l)} = (W^{(l+1)})^T \delta^{(l+1)} \cdot \text{activation}'(z^{(l)})$$ => $$W^{(l)} \leftarrow W^{(l)} - \eta \cdot \delta^{(l)}(a^{(l-1)})^T$$ $$b^{(l)} \leftarrow b^{(l)} - \eta \cdot \delta^{(l)}$$