CSE463: Neural Networks

Activation and Loss Functions

+

Regression

by:

Hossam Abd El Munim

Computer & Systems Engineering Dept.,

Ain Shams University,

1 El-Sarayat Street, Abbassia, Cairo 11517

Many slides and images are taken from the internet.

Bipolar Perceptron Training
Recall

Batch Training

Input: $(\mathbf{X}_m, y_m)$, m = 1, 2, …,N

Set $\mathbf{W} = [0, 0,.., 0]^T$

Repeat

    $\mathbf{delta} = [0, 0,.., 0]^T$

    for m = 1 to N do

        if $y_m \mathbf{W}.\mathbf{X}_m <= 0$

            $\mathbf{delta} = \mathbf{delta} - y_m \mathbf{X}_m$

    $\mathbf{delta} = \mathbf{delta} / N$

    $\mathbf{W} = \mathbf{W} - \mathbf{delta}$

Until $||\mathbf{delta}|| < \epsilon$

Online Training

Input: $(\mathbf{X}_m, y_m)$, $m = 1, 2, \ldots, N$

Set $\mathbf{W} = [0, 0, .., 0]^T$

Repeat

  $\mathbf{delta} = [0, 0, .., 0]^T$

  for $m = 1$ to N do

    if $y_m \mathbf{W}.\mathbf{X}_m <= 0$

      $\mathbf{delta} = \mathbf{delta} - y_m \mathbf{X}_m$

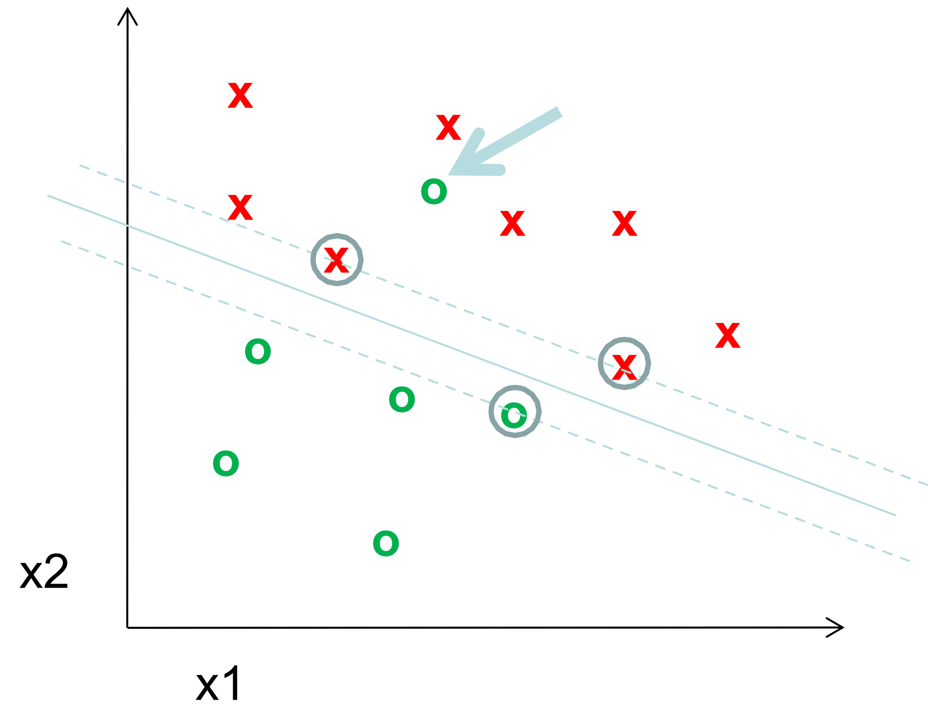
      $\mathbf{W} = \mathbf{W} - \mathbf{delta} \,/\, N$

Until $||\mathbf{delta}|| < \epsilon$

Bipolar Perceptron and SVM

Find a *linear function* to separate the classes:

$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$



What if my data are not linearly separable?

Introduce flexible ‘hinge’ loss (or ‘soft-margin’)

Relation to SVM: Modified Loss Function

$$J(\mathbf{w}) = \frac{1}{n}\sum_{m=1}^{n} \max(0,\ 1 - y_m \mathbf{w}^T \mathbf{x}_m)$$

Introduce flexible ‘hinge’ loss (or ‘soft-margin’)

Relation to SVM: Modified Loss Function

Does this affect the training algorithm?

Choice of Activation Function (1)

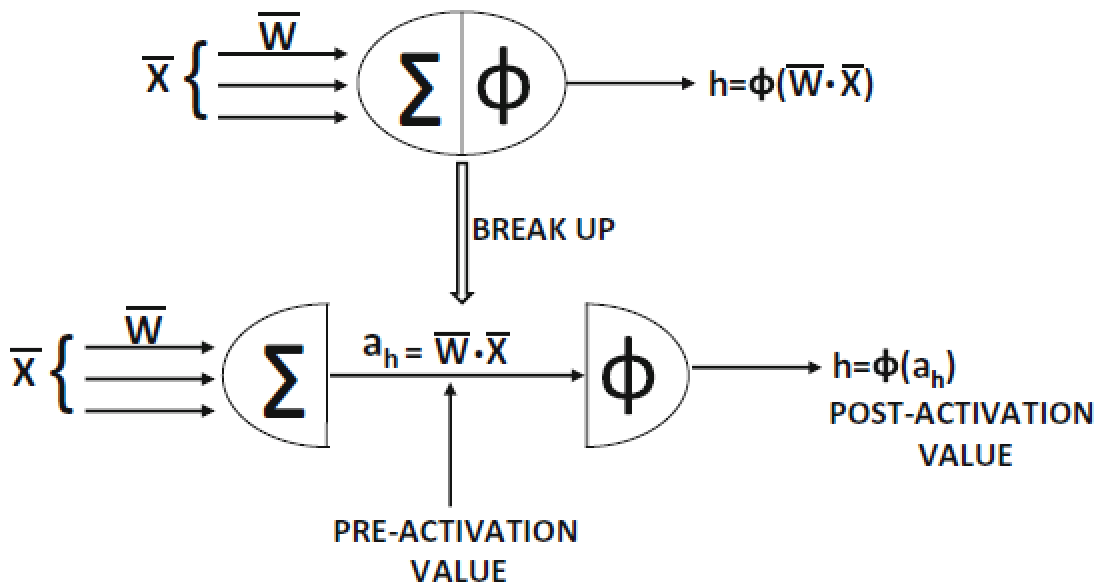


Figure 1.7: Pre-activation and post-activation values within a neuron

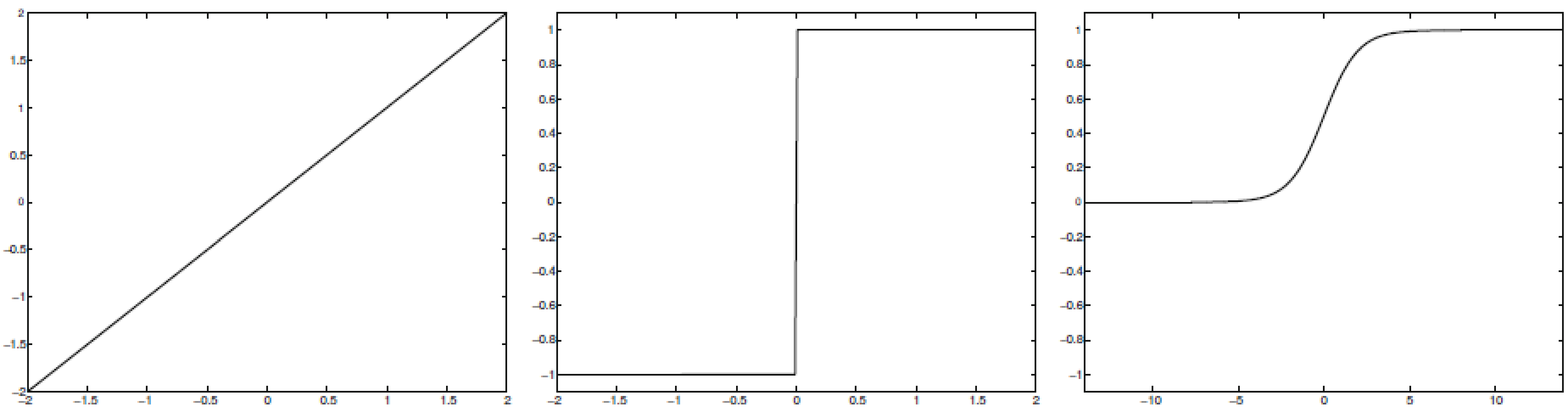
Choice of Activation Function (2)

The classical activation functions that were used early in the development of neural networks were the sign, sigmoid, and the hyperbolic tangent functions:

$$\Phi(v) = \text{sign}(v) \text{ (sign function)}$$

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

$$\Phi(v) = \frac{e^{2v}-1}{e^{2v}+1} \text{ (tanh function)}$$



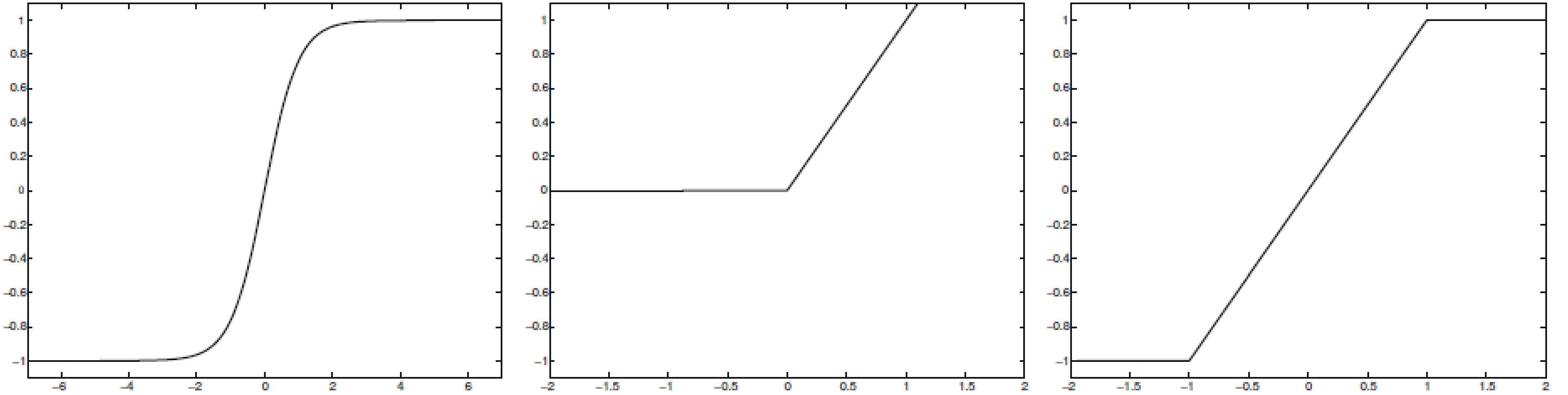
(a) Identity (b) Sign (c) Sigmoid

Choice of Activation Function (3)

The sigmoid and the tanh functions have been the historical tools of choice for incorporating nonlinearity in the neural network. In recent years, however, a number of piecewise linear activation functions have become more popular:

$$\Phi(v) = \max\{v, 0\} \text{ (Rectified Linear Unit [ReLU])}$$
$$\Phi(v) = \max\{\min[v, 1], -1\} \text{ (hard tanh)}$$

The ReLU and hard tanh activation functions have largely replaced the sigmoid and soft tanh activation functions in modern neural networks because of the ease in training multi-layered neural networks with these activation functions.



(d) Tanh (e) ReLU (f) Hard Tanh

Figure 1.8: Various activation functions

Sigmoid and tanh Derivatives

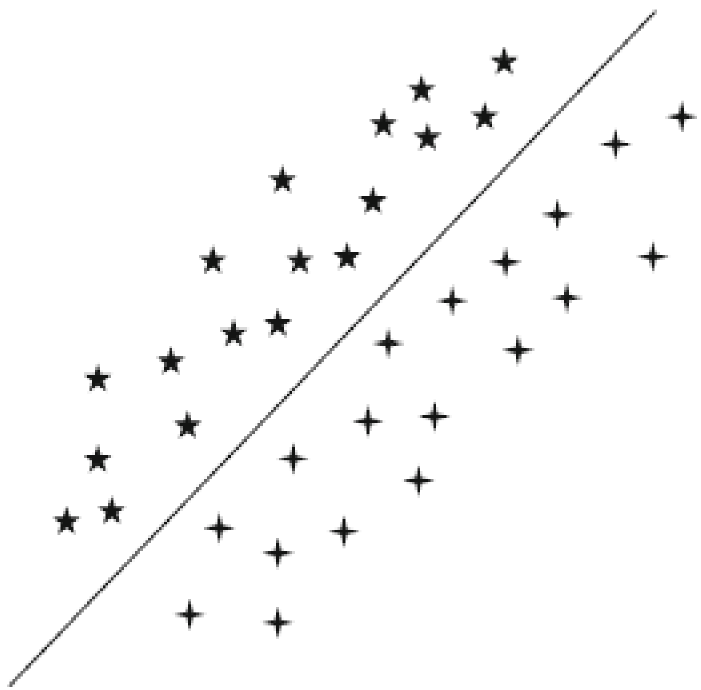
Sigmoid (u) = f(u)

$df/du = f(1-f)$

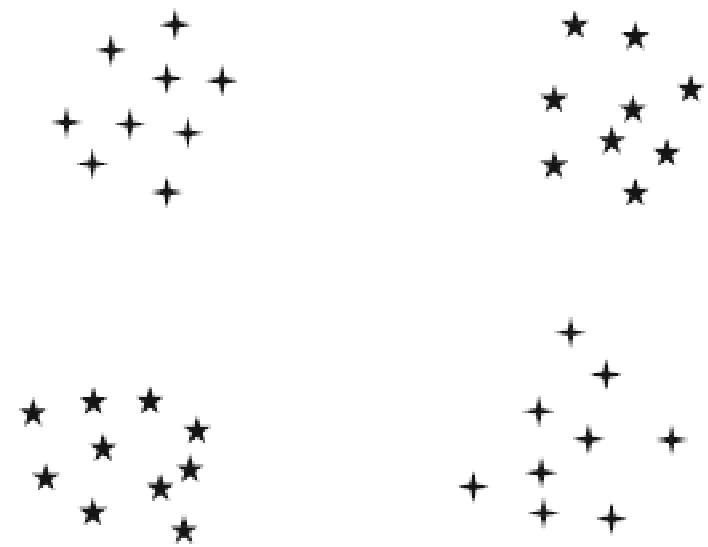
tanh (v) = g(v)

$dg/dv = (1-g^2)$

Will it work? Do need a multilayer network?

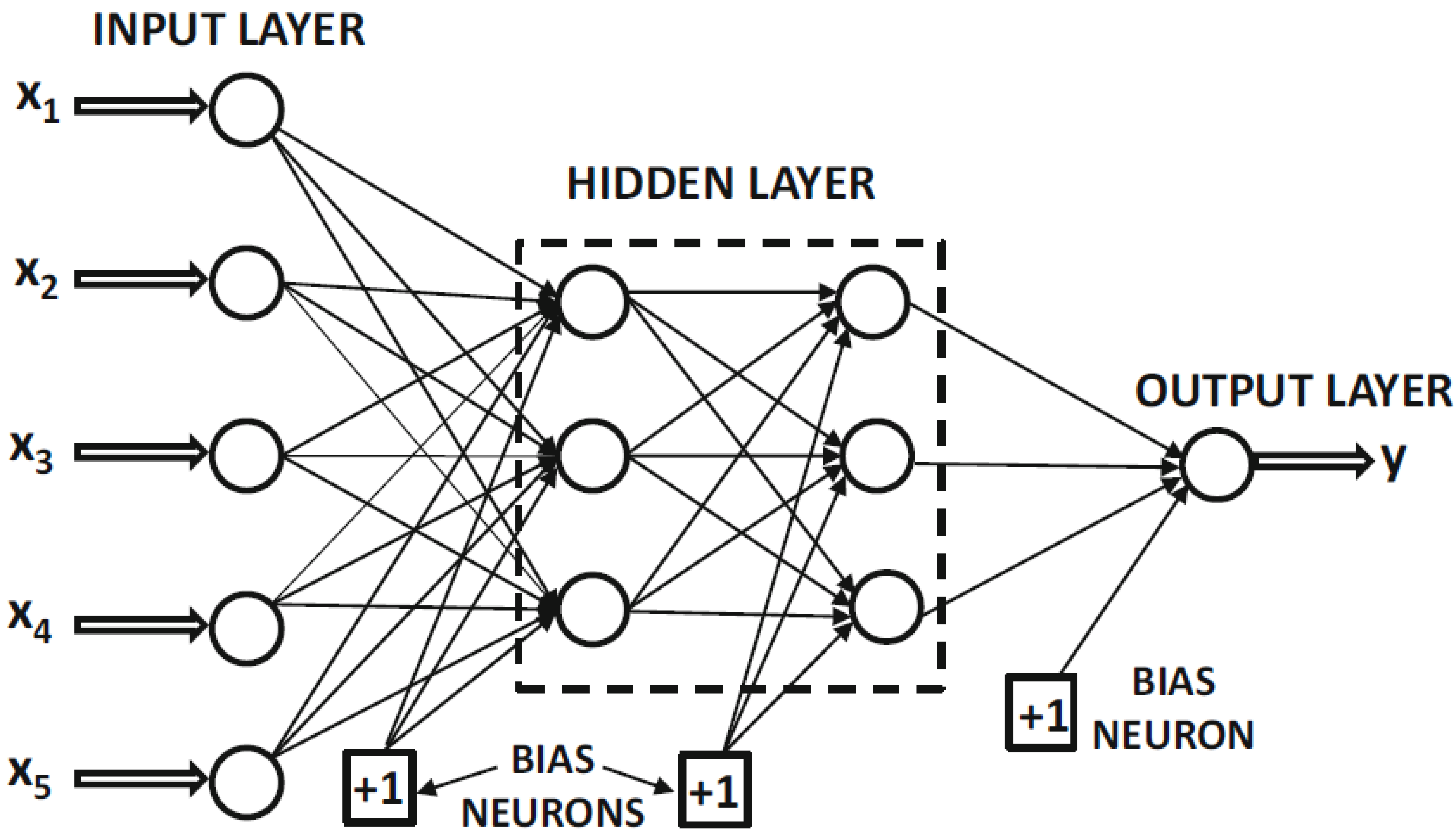


LINEARLY SEPARABLE



NOT LINEARLY SEPARABLE

A Multilayer Network



Multiple Outputs

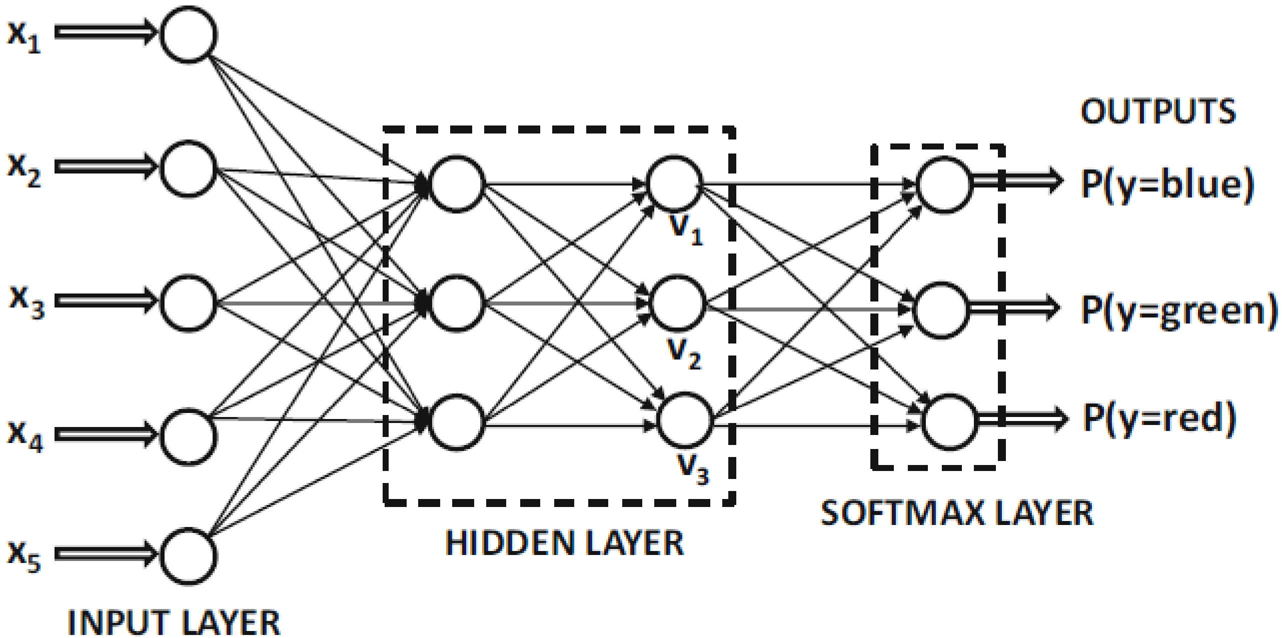
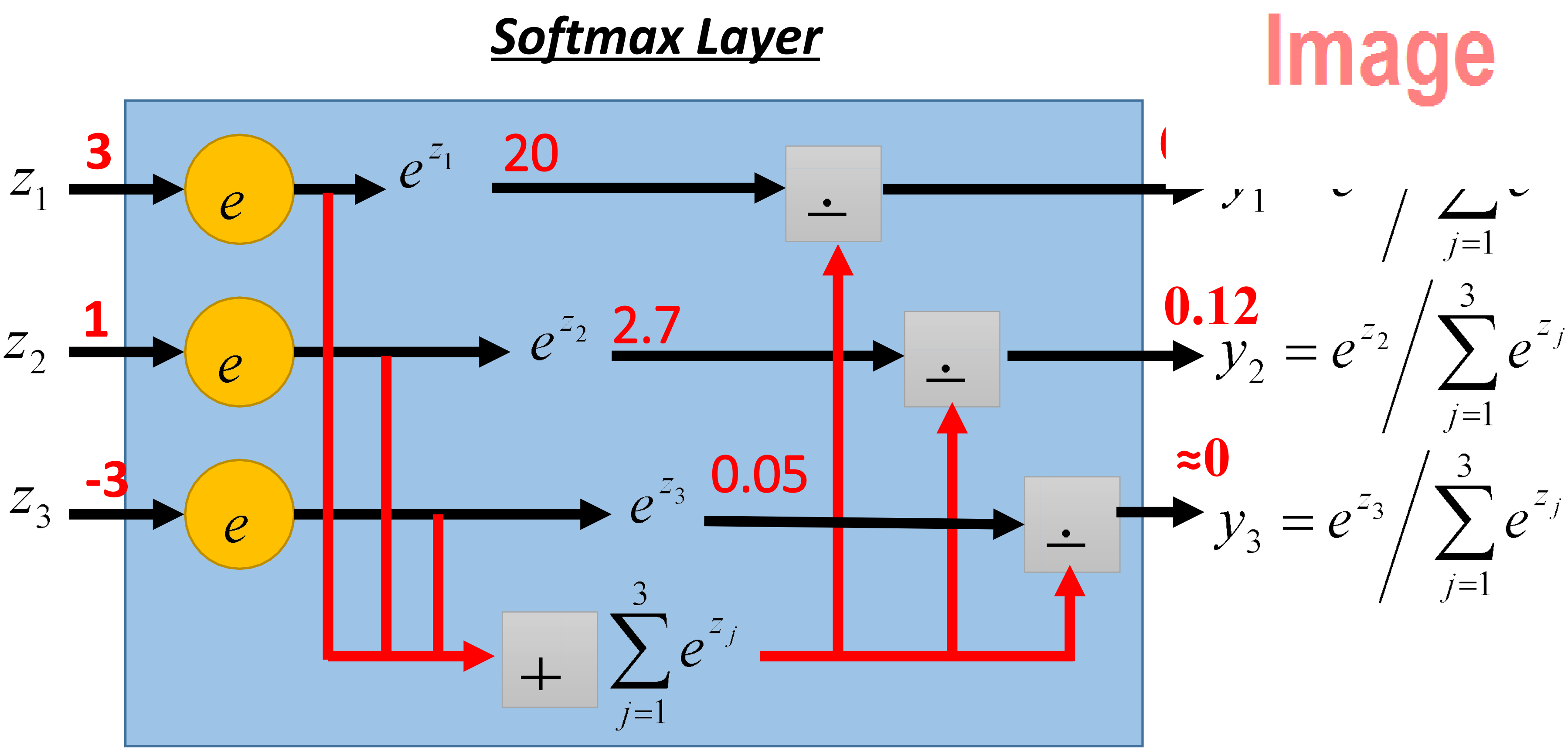


Figure 1.9: An example of multiple outputs for categorical classification with the use of a softmax layer

Probabilistic Prediction
Index of Maximum is the Prediction

- Softmax layer as the output layer



Choice of Loss Function for Probabilistic Prediction

Two Categories: Binary Classification Identity Activation Function

Binary targets (logistic regression): In this case, it is assumed that the observed value y is drawn from $\{-1, +1\}$, and the prediction $\hat{y}$ is a an arbitrary numerical value on using the identity activation function. In such a case, the loss function for a single instance with observed value y and real-valued prediction $\hat{y}$ (with identity activation) is defined as follows:

$$L = \log(1 + \exp(-y \cdot \hat{y})) \tag{1.14}$$

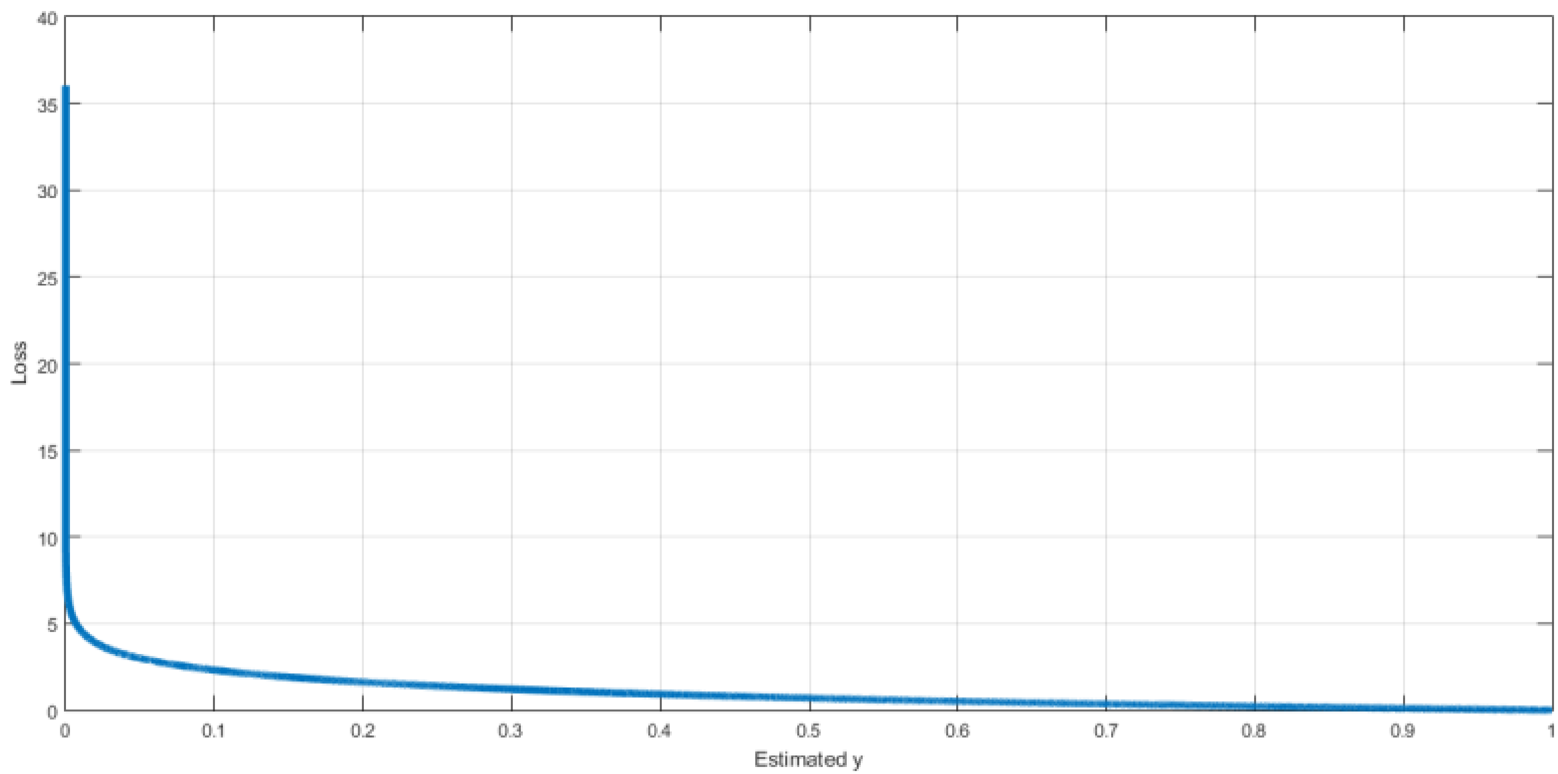
Two Categories: Binary Classification Sigmoid Activation Function

This type of loss function implements a fundamental machine learning method, referred to as *logistic regression*. Alternatively, one can use a sigmoid activation function to output $\hat{y} \in (0, 1)$, which indicates the probability that the observed value y is 1. Then, the negative logarithm of $|y/2 - 0.5 + \hat{y}|$ provides the loss, assuming that y is coded from $\{-1, 1\}$. This is because $|y/2 - 0.5 + \hat{y}|$ indicates the probability that the prediction is correct. This observation illustrates that one can use various combinations of activation and loss functions to achieve the same result.

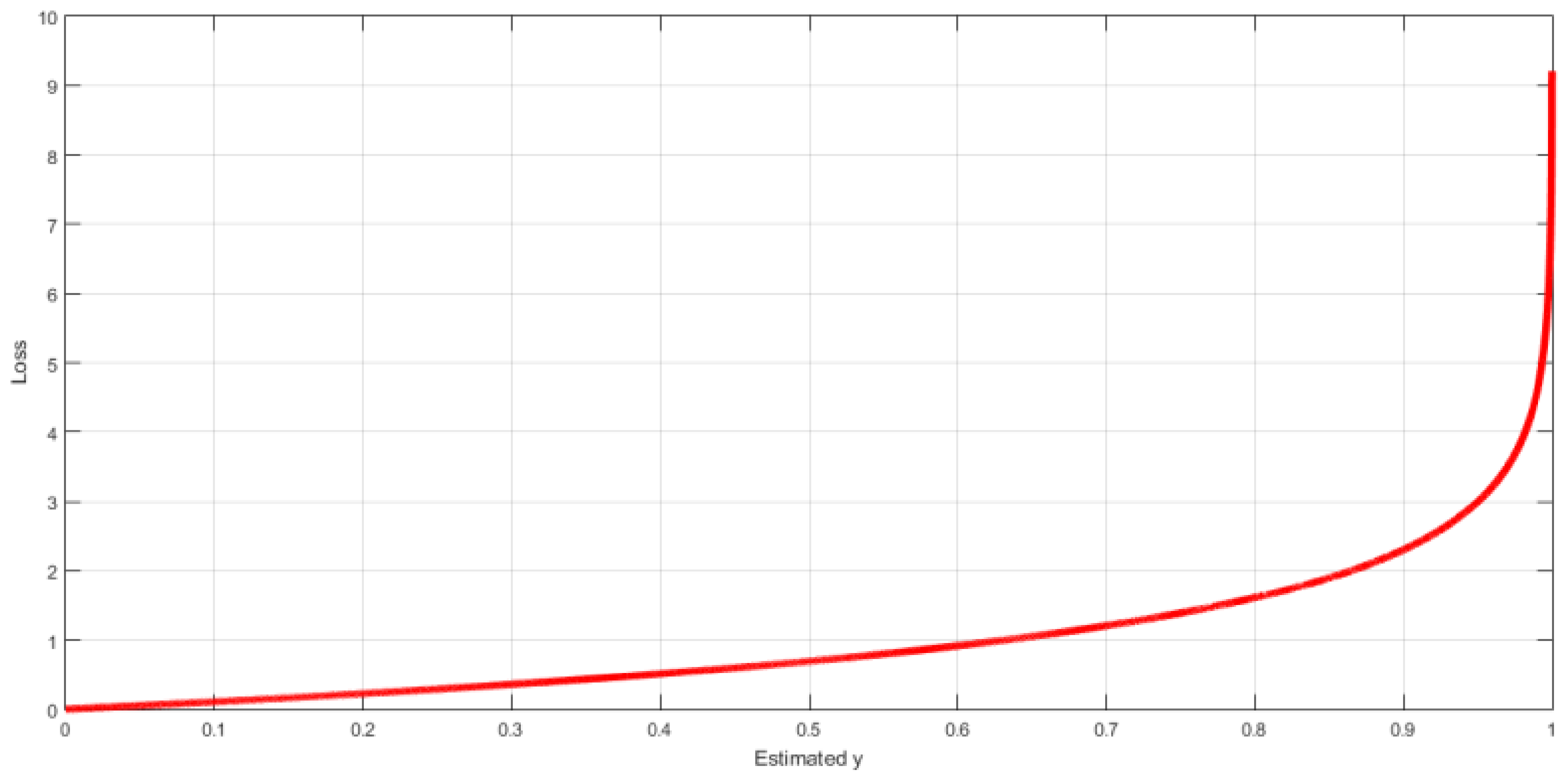
$$L = -\log(|\frac{y}{2} - \frac{1}{2} + \hat{y}|)$$

$$\hat{y} = \frac{1}{1 + e^{-(\mathbf{W}.\mathbf{X})}}$$

Two Categories: Binary Classification Sigmoid Activation Function (y=1)



Two Categories: Binary Classification Sigmoid Activation Function (y = -1)



Multi-Classifications

Categorical targets: In this case, if $\hat{y}_1 \ldots \hat{y}_k$ are the probabilities of the k classes (using the softmax activation of Equation 1.9), and the rth class is the ground-truth class, then the loss function for a single instance is defined as follows:

$$L = -\log(\hat{y}_r) \tag{1.15}$$

Machine Learning with Shallow Neural Networks

Neural Architectures for Binary Classification Models

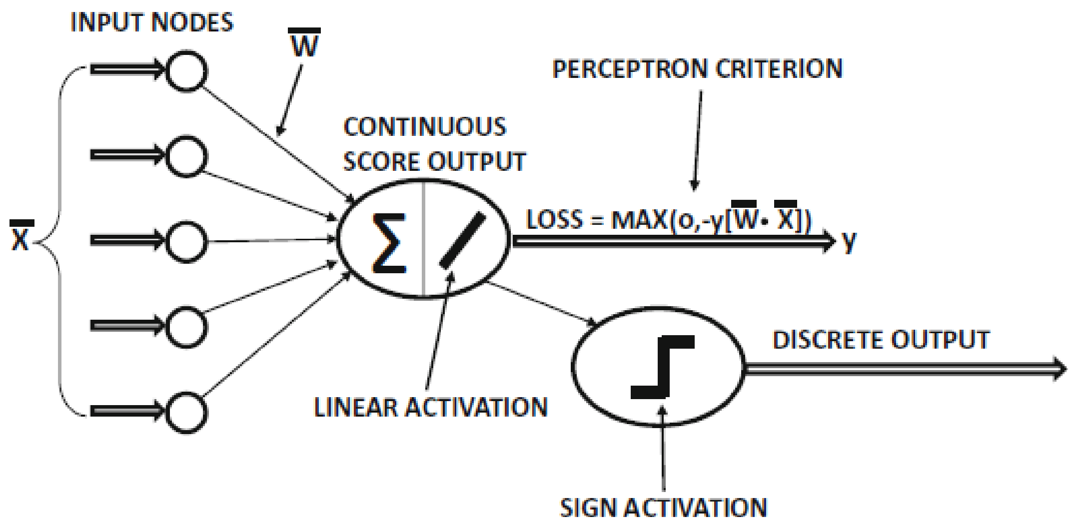
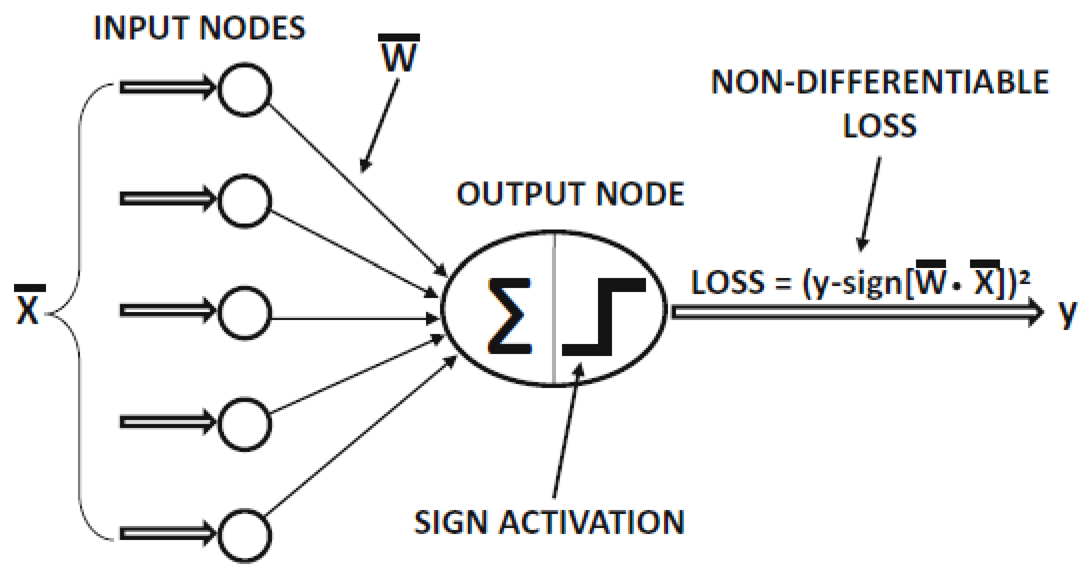
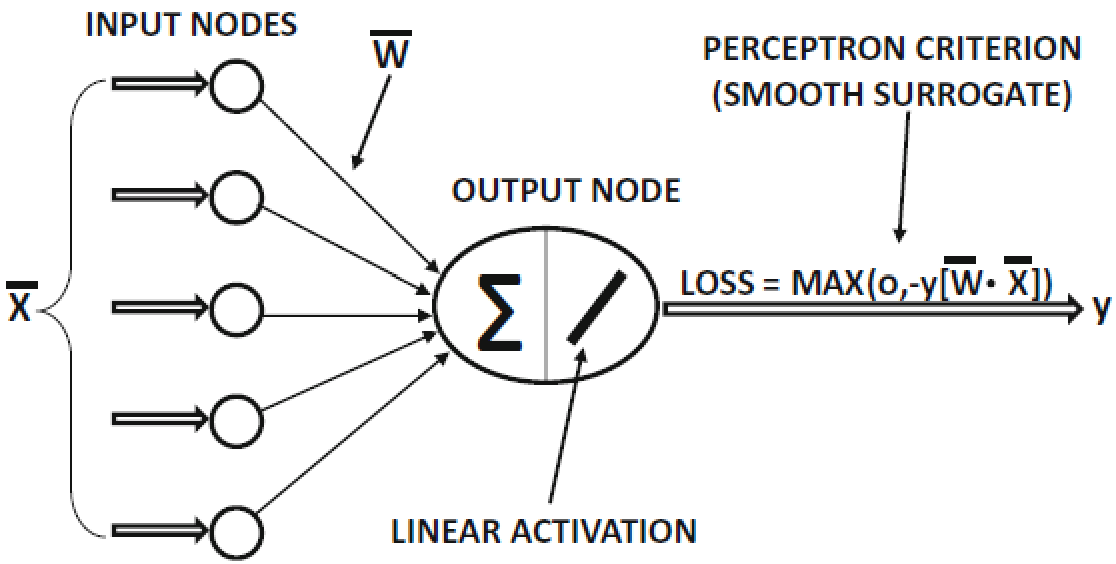


Figure 2.2: An extended architecture of the perceptron with both discrete and continuous predictions

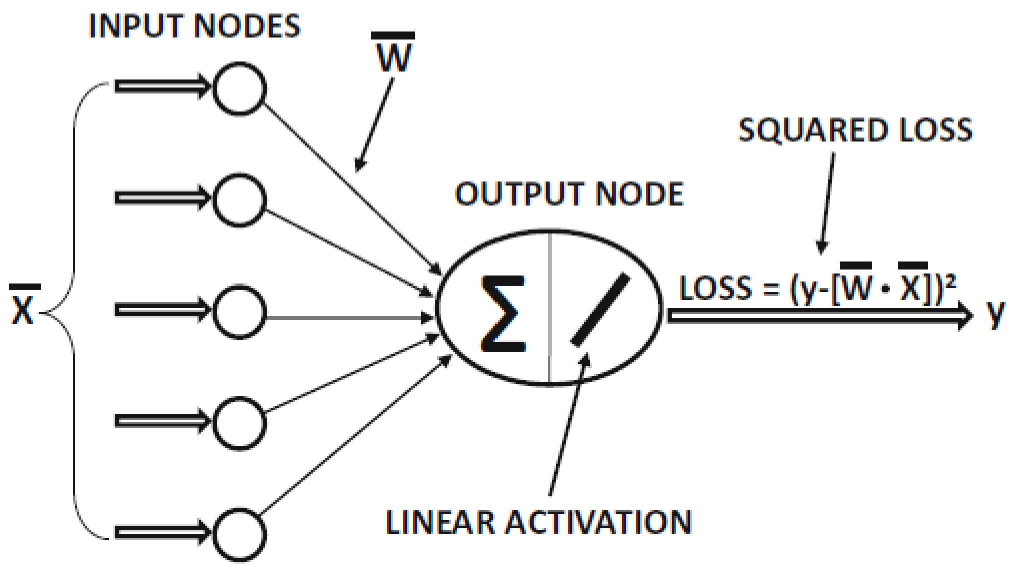
Different Variants of the Perceptron Binary Classifier – Bad Loss



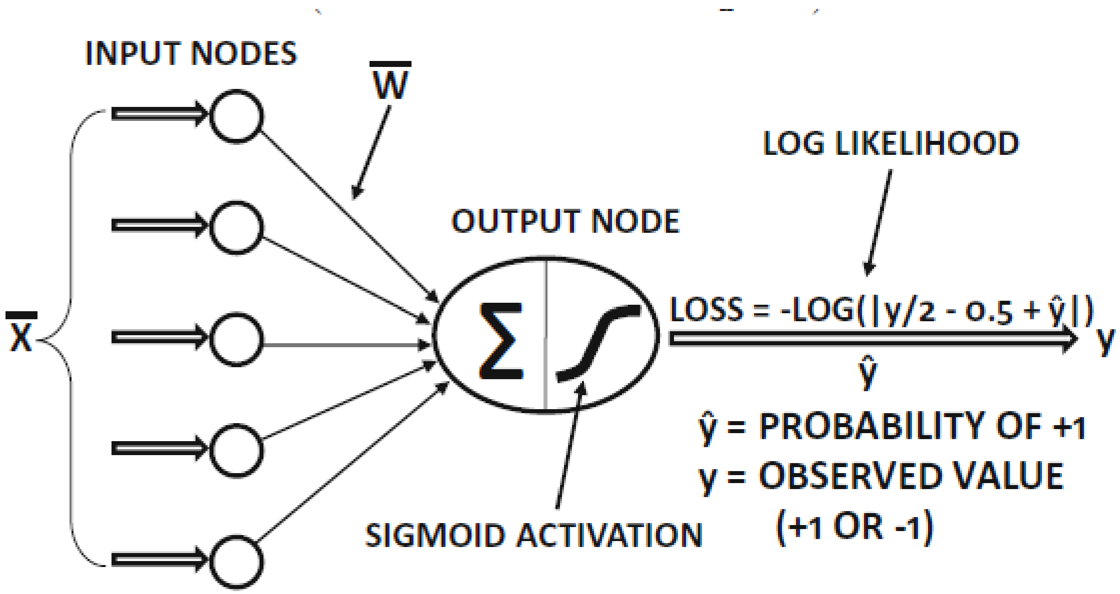
Different Variants of the Perceptron Binary Classifier – Smooth Loss Function



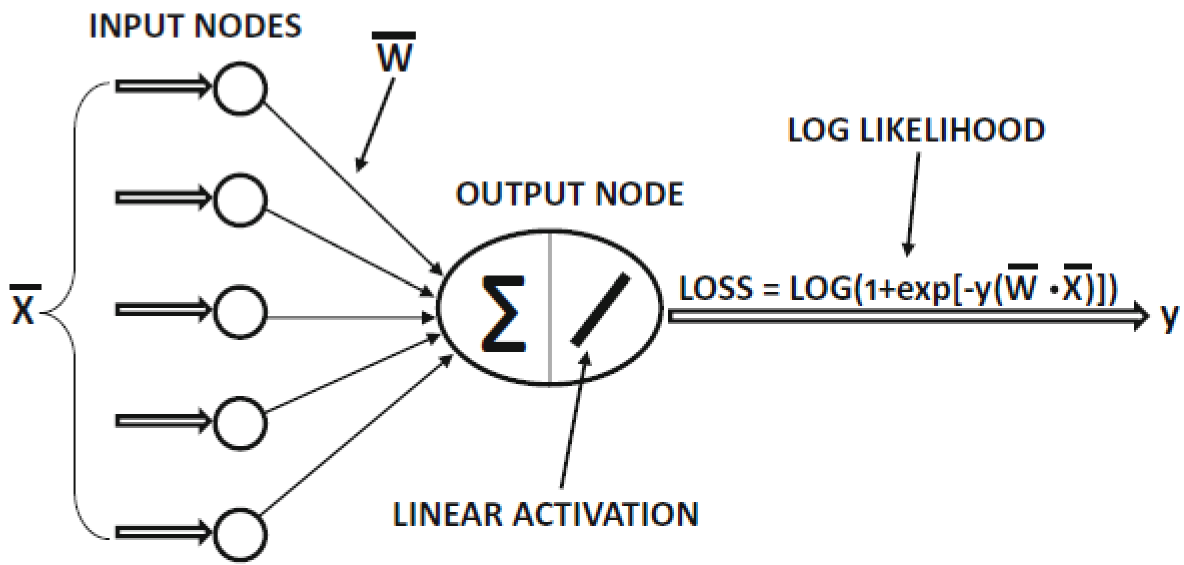
Different Variants of the Perceptron Linear Regression – SSD Loss Function



Different Variants of the Perceptron Logistic Regression – LOG Likelihood Loss Function



Different Variants of the Perceptron Logistic Regression – LOG Likelihood Loss Function (Alternative)



Different Variants of the Perceptron Binary Classifier – Hinge Loss (SVM)