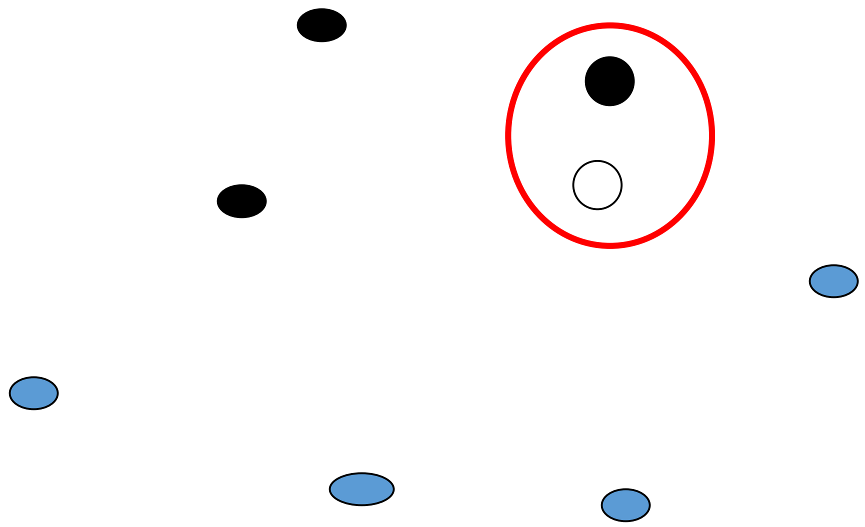
آکادمی یوتک

Machine Learning Course

Vahid Reza Khazaie

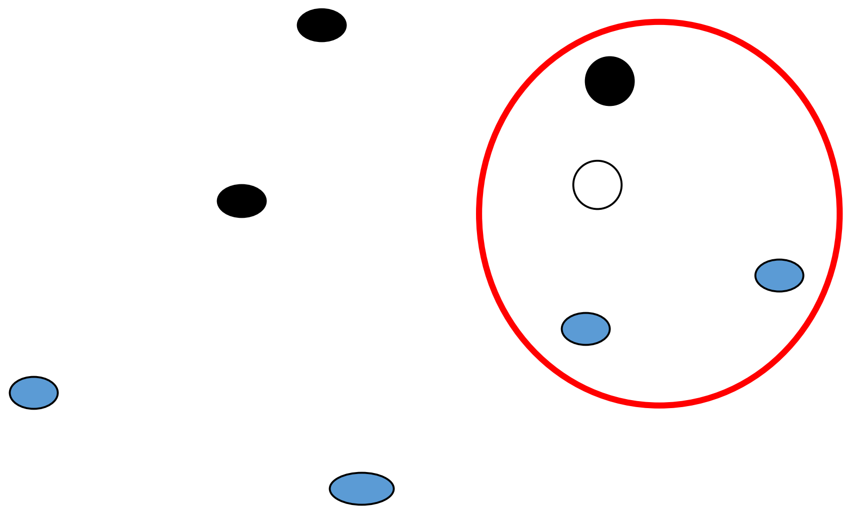
1-NN

- 1-Nearest Neighbor



3-NN

- 3-Nearest Neighbor



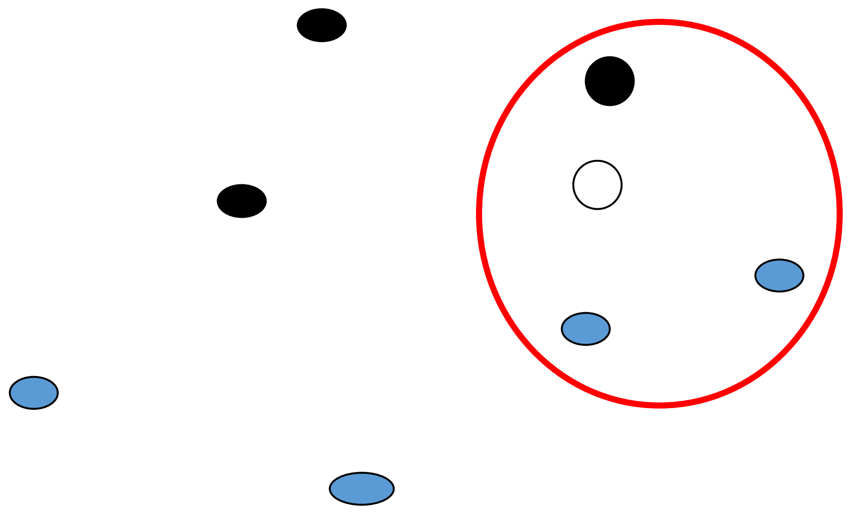
k-NN

- In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

- In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.
- In k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors.

3-NN

- 3-Nearest Neighbor



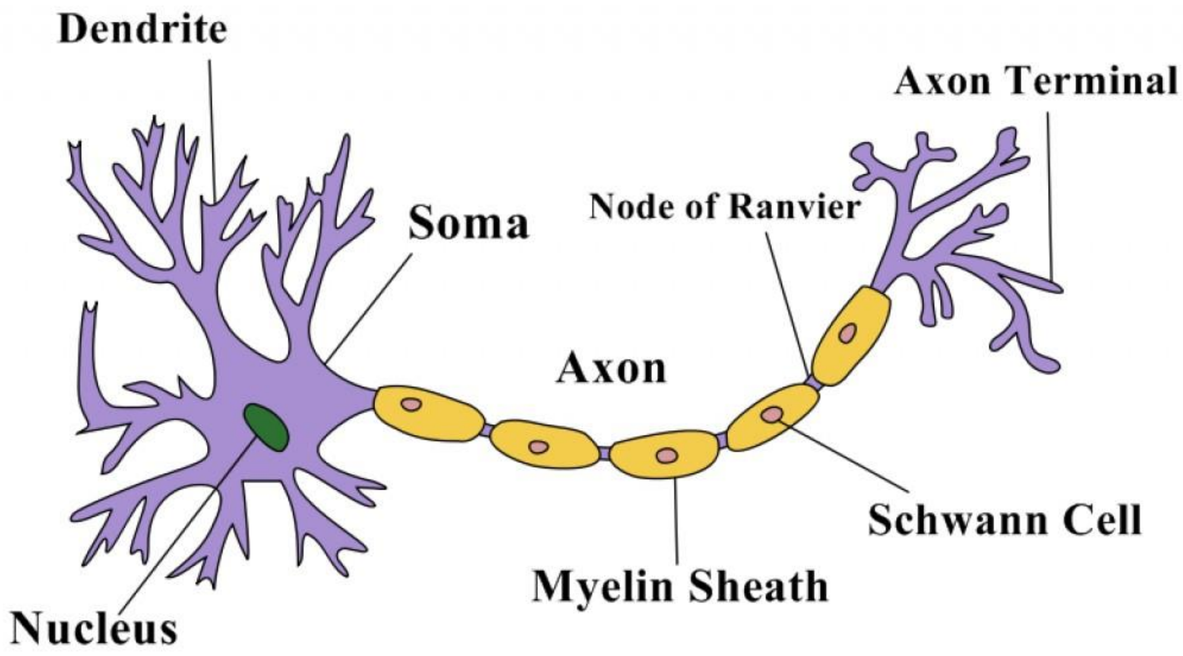
10_k-nearest_neighbor.ipynb

Introduction to the Perceptron

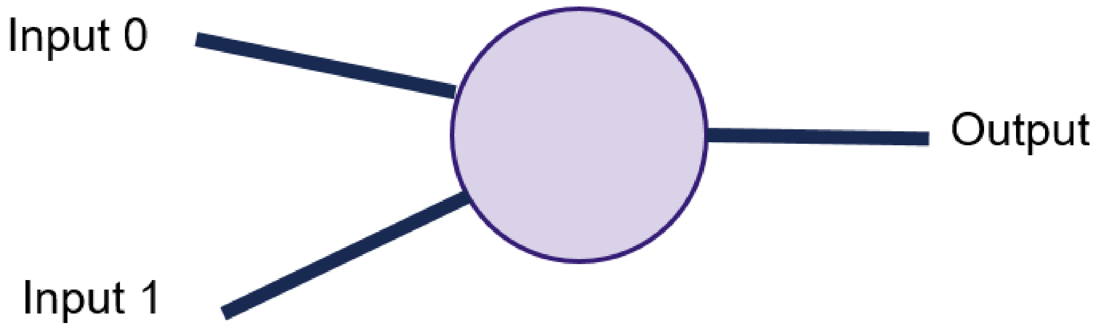
- In machine learning, the **perceptron** is an algorithm for supervised learning of binary classifiers. A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class.

Introduction to the Perceptron

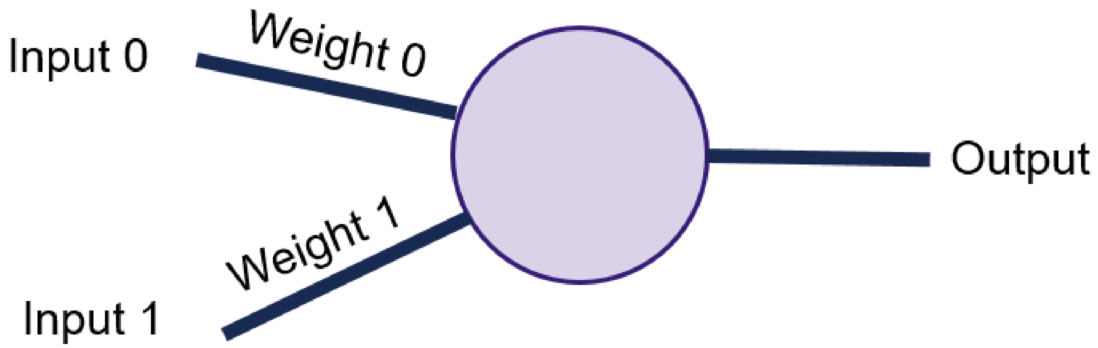
- **Biological Neuron**



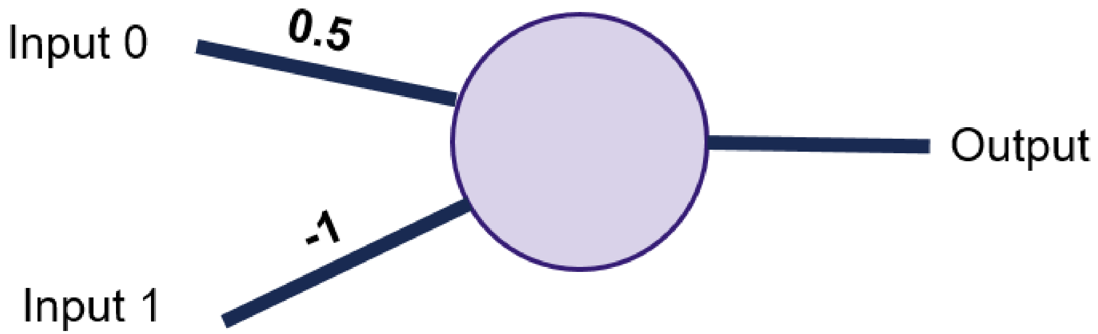
Perceptron



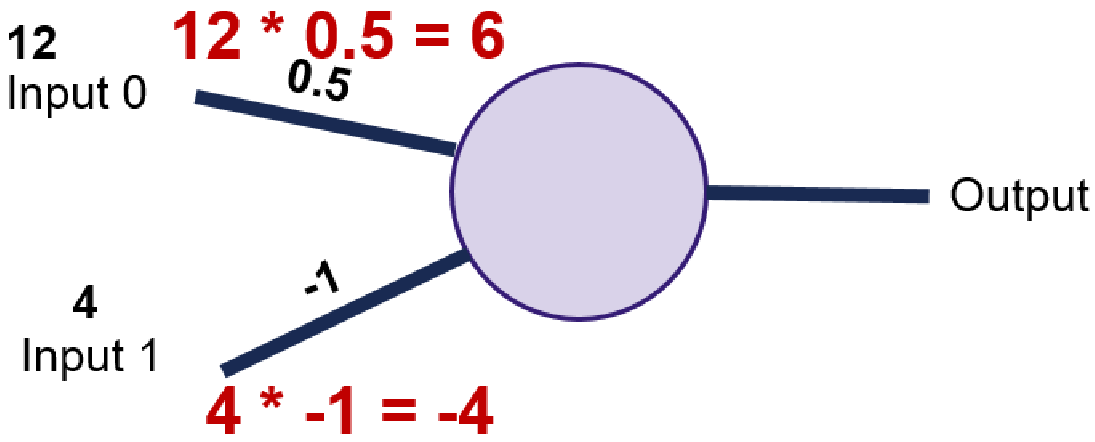
Perceptron



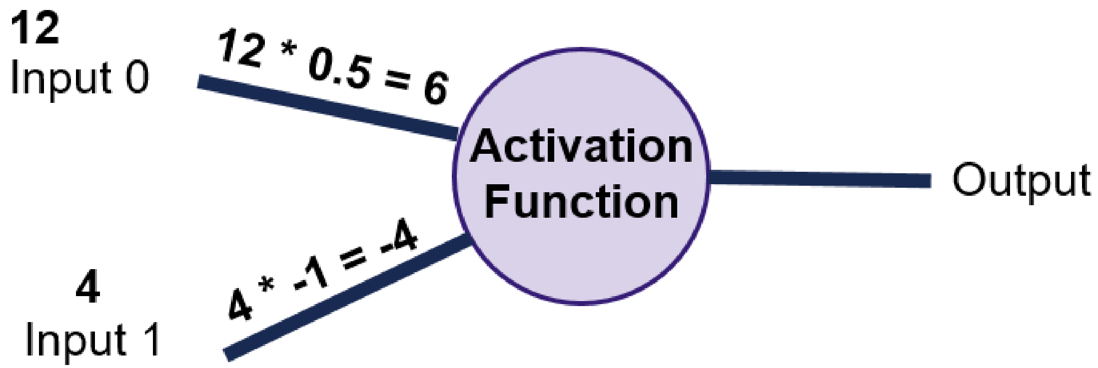
Perceptron



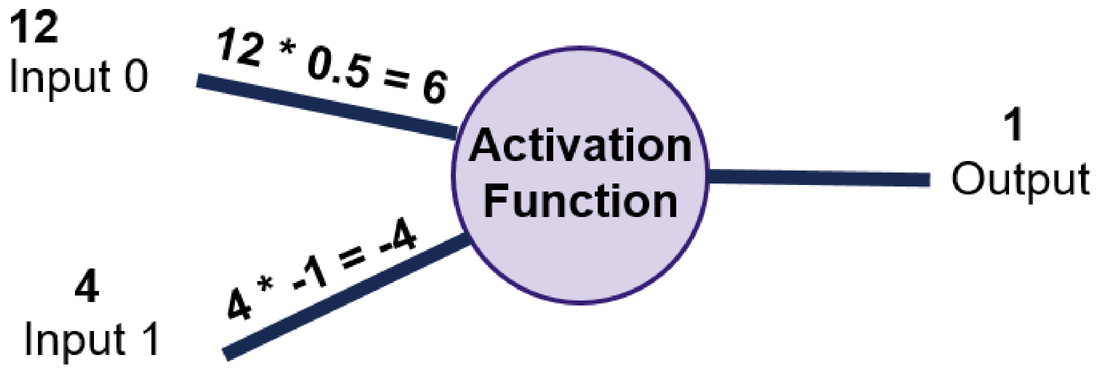
Perceptron



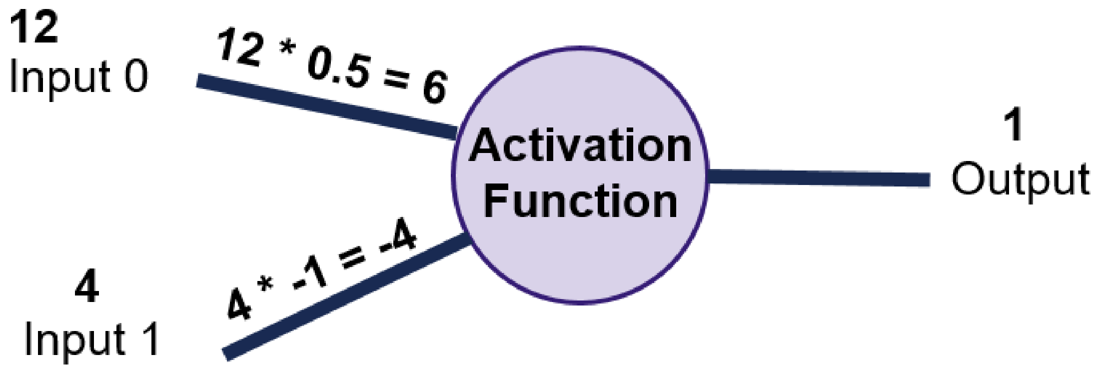
Perceptron



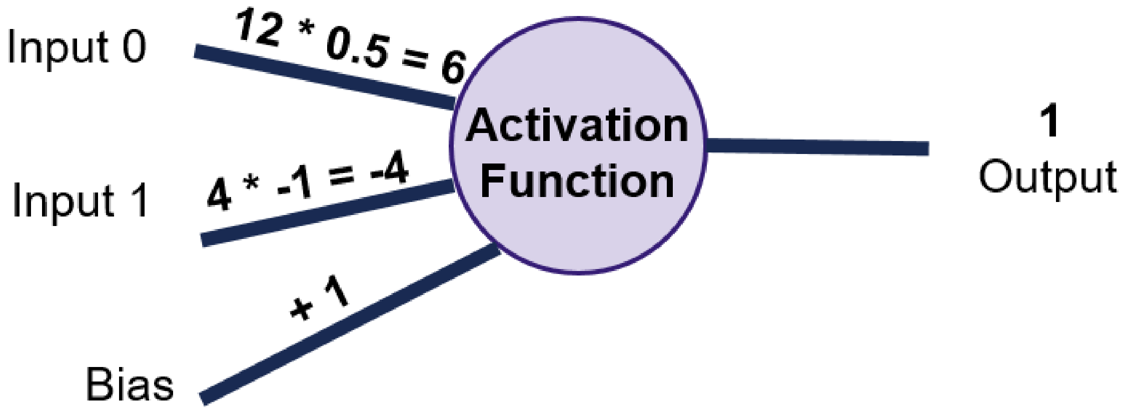
Perceptron



Perceptron

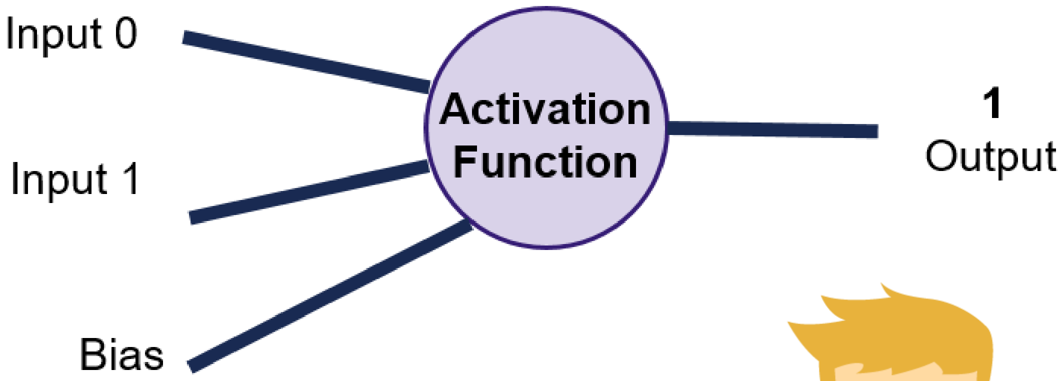


Perceptron



Input 0
12 * 0.5 = 6
Input 1
4 * -1 = -4
Bias
+ 1
Activation Function
1
Output

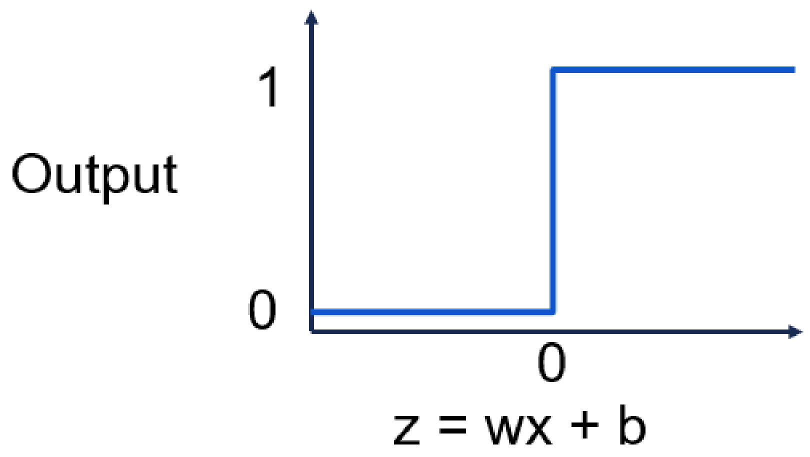
Perceptron



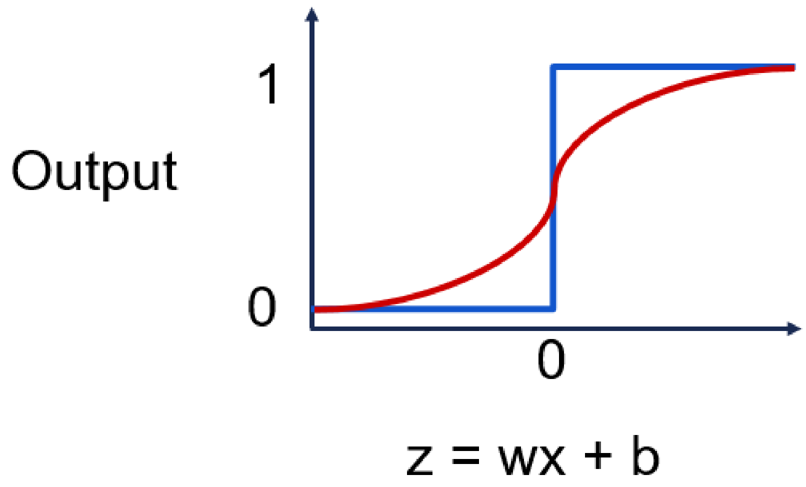
$$\sum_{i=0}^{n} w_i x_i + b$$

[14_perceptron.ipynb]

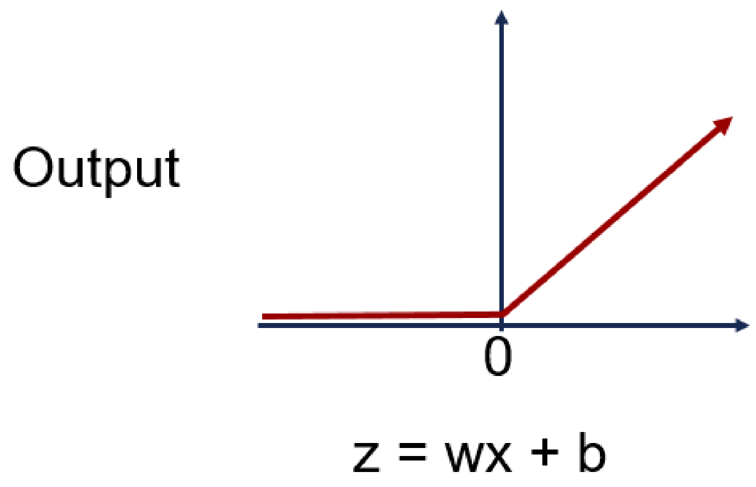
Activation Function



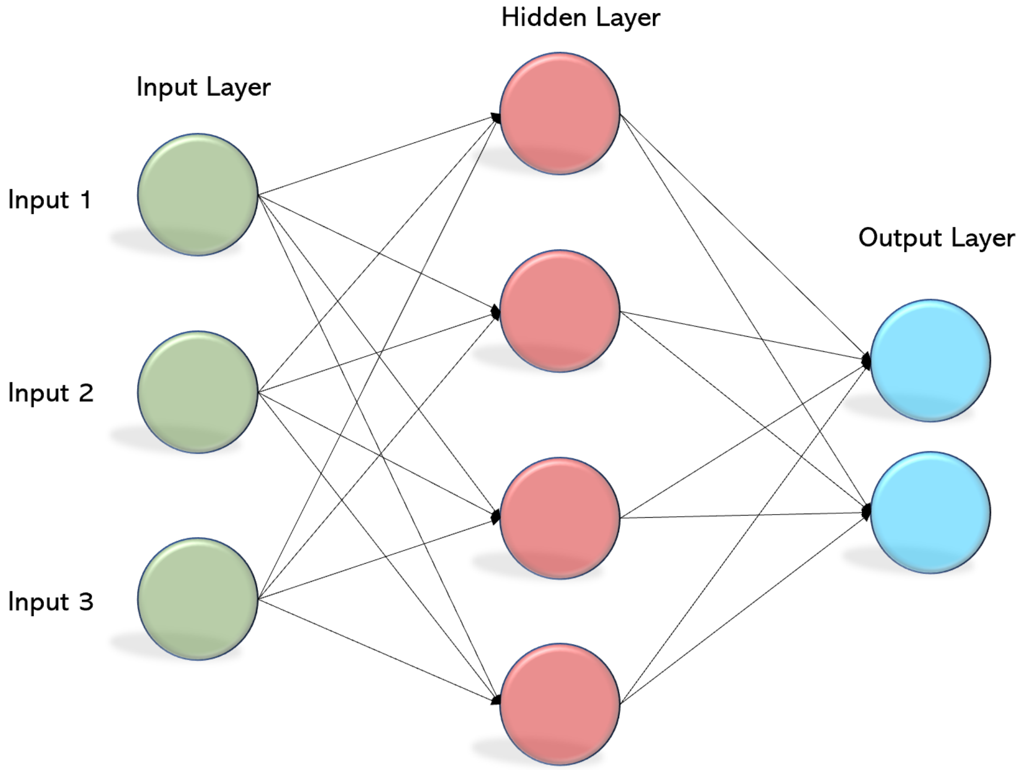
Activation Function



Activation Function



Introduction to Neural Networks



Cost Functions

- $C = \Sigma(y\text{-}yhat)2 / n$

- $C = (-1/n)\ \Sigma\ (y \cdot \ln(yhat) + (1-y) \cdot \ln(1-yhat)($

Learning

- Neurons
- Cost Function
- Gradient Decent and Backpropagation

Learning

- Gradient Decent

$$\text{Repeat until convergence } \{$$

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$$\}$$

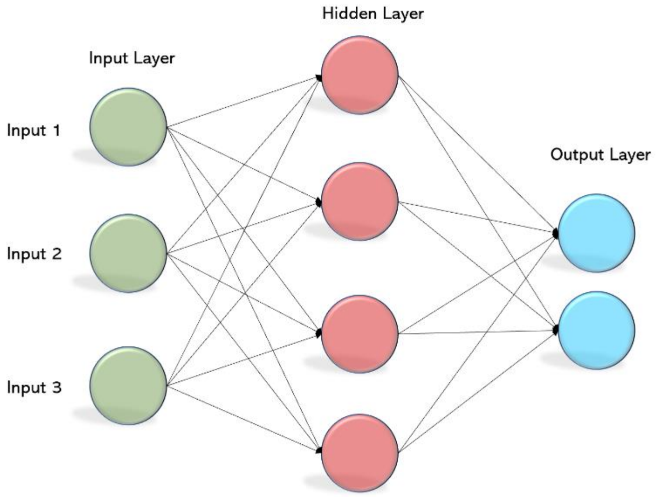
Learning

- Backpropagation

Repeat until convergence {

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

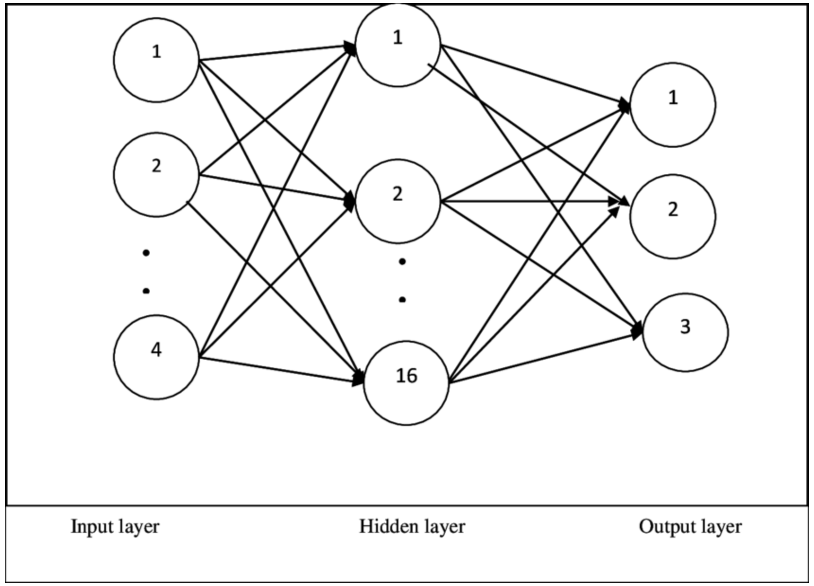
}



Learning

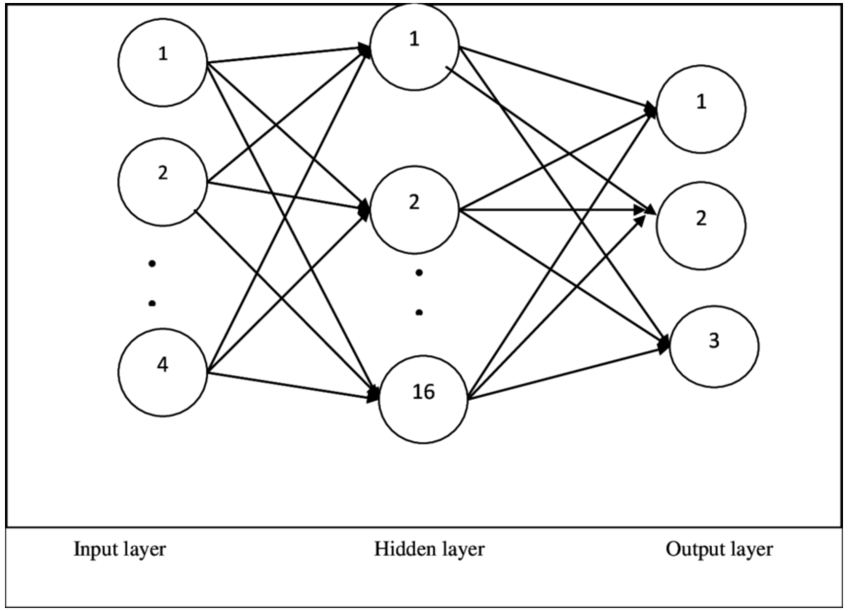
- Softmax

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$$

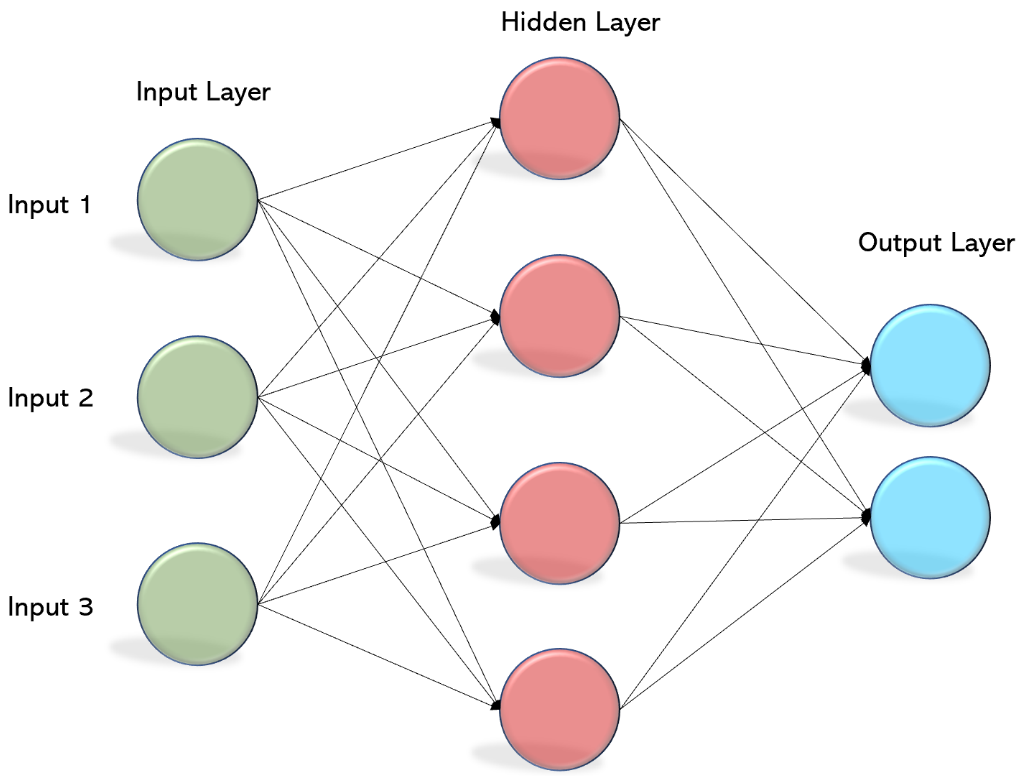


MLP

- Visualization   playground.tensorflow.org

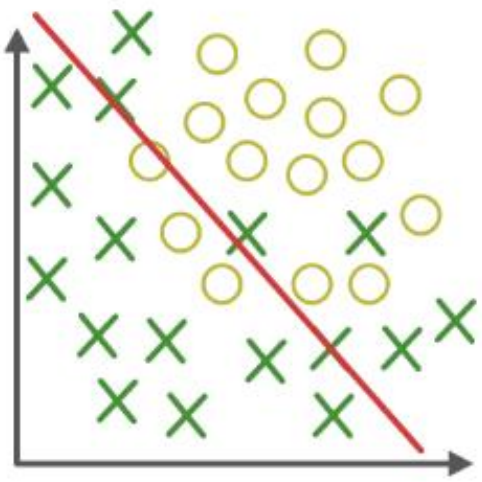


MLP

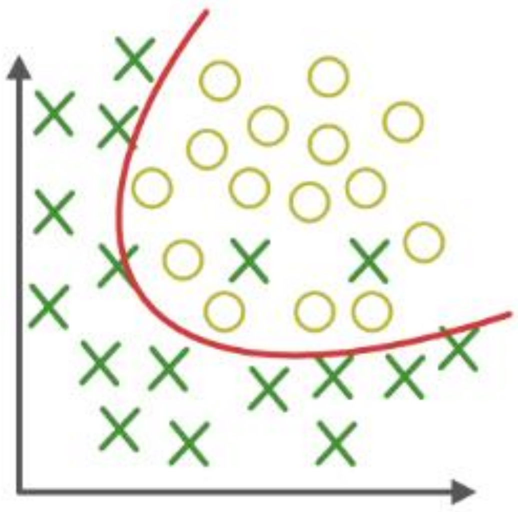


12_mlp.ipynb

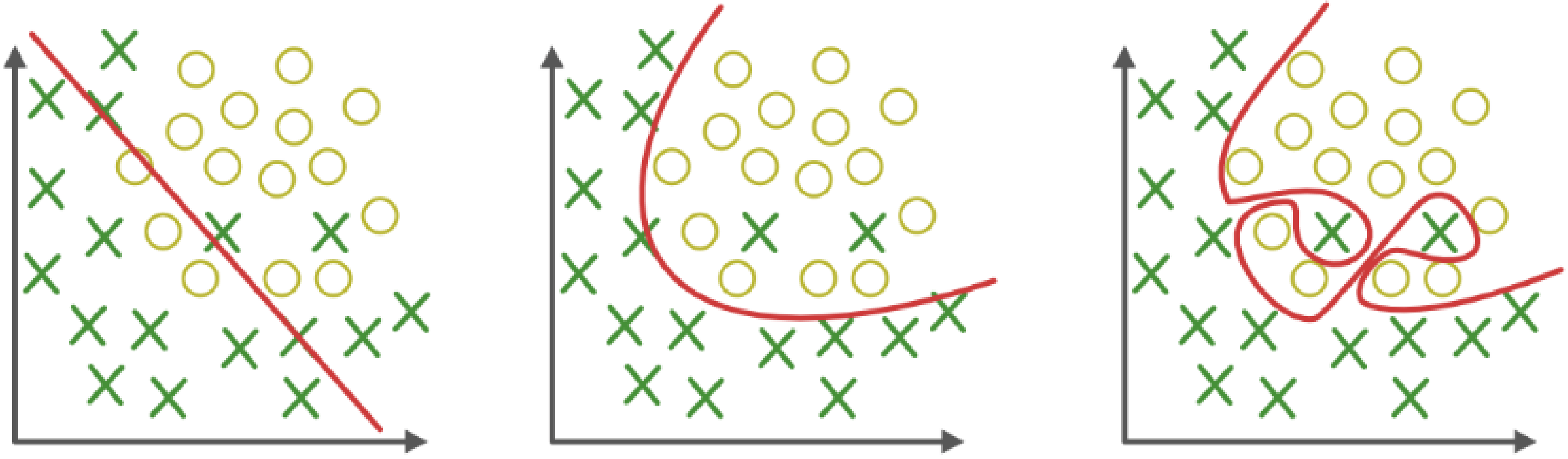
Overfitting, Underfitting



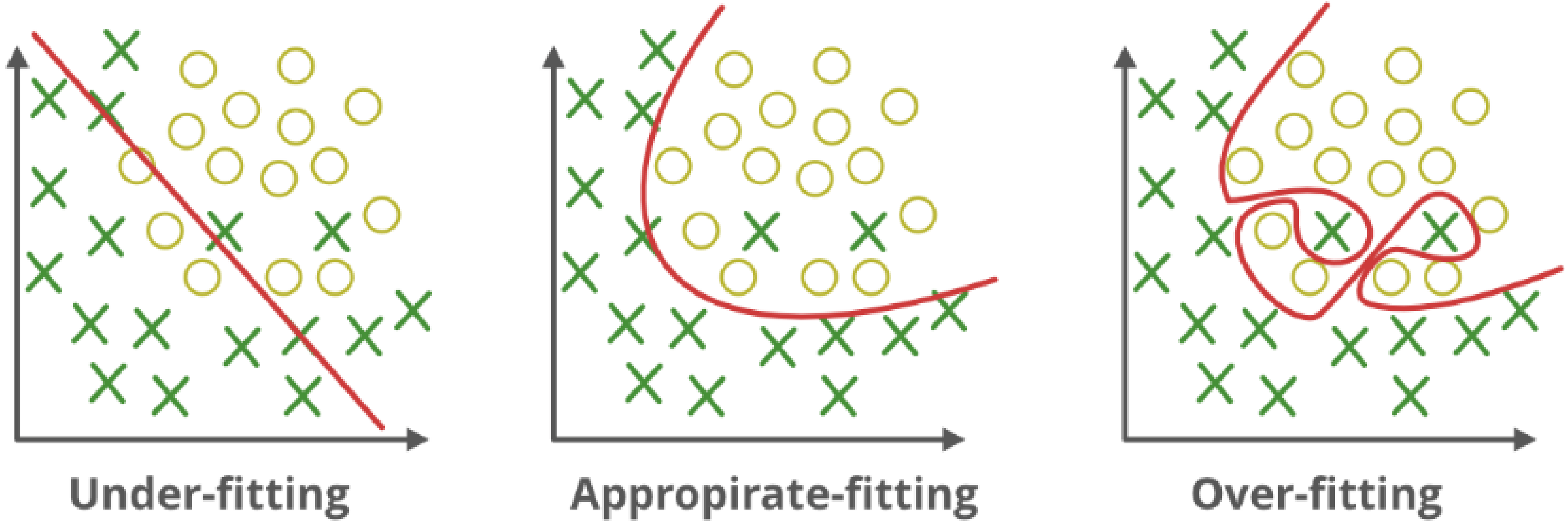
Overfitting, Underfitting



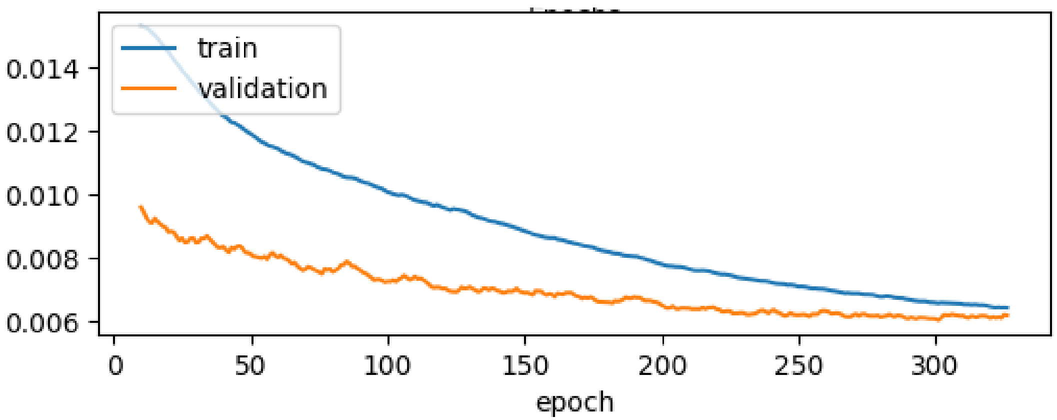
Overfitting, Underfitting



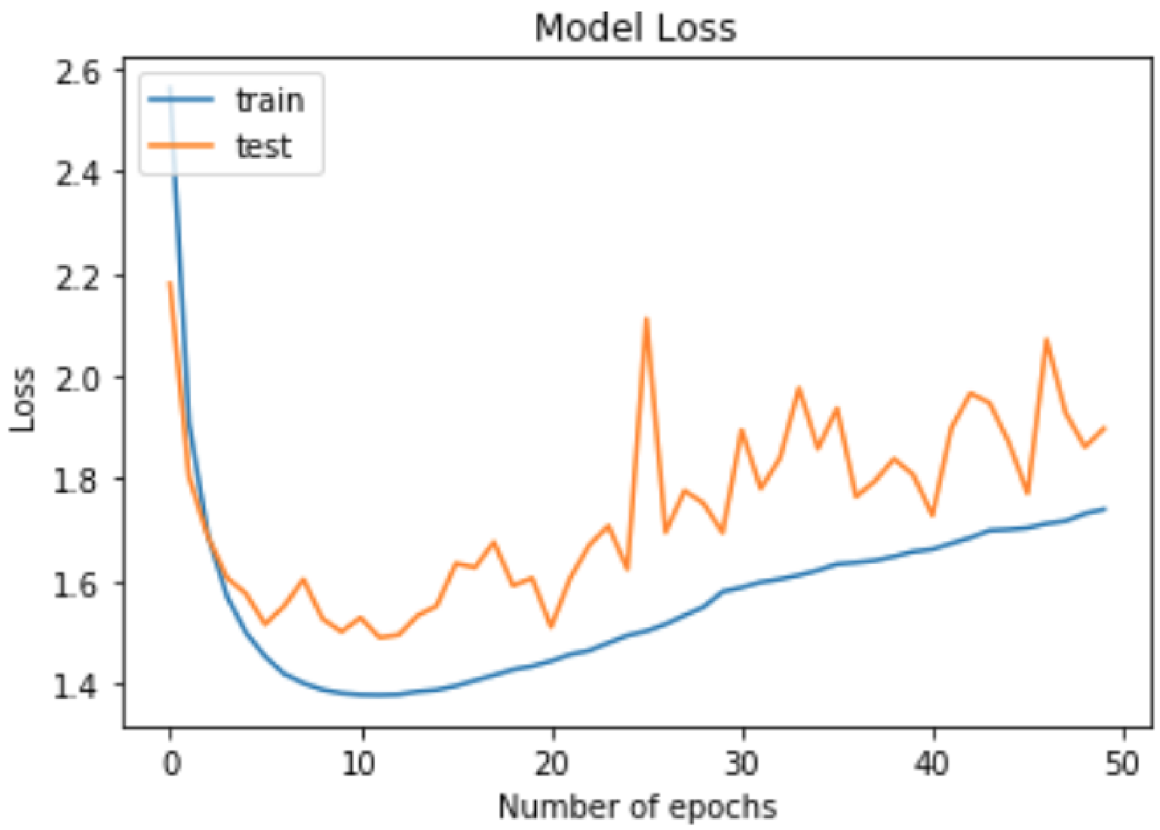
Overfitting, Underfitting



Overfitting, Underfitting



Overfitting, Underfitting



Avoid Overfitting

- Weight Regularization
- Dropout
- More Data
- Cross-validation

Performance Evaluation with Resampling

- Train/Validation/Test Sets
- K-fold Cross Validation
- Leave-one-out Cross Validation
- Bootstrapping

13_resampling.ipynb