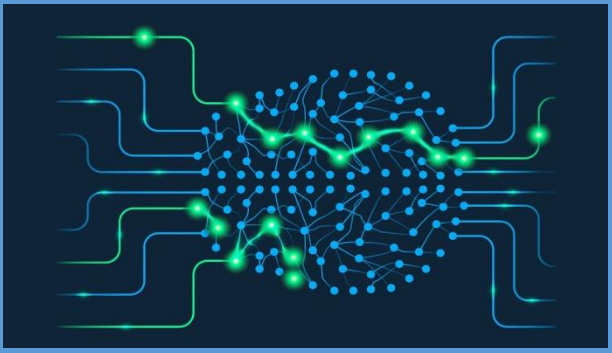
EMS702P Statistical Thinking and Applied Machine Learning

Week 8.1 – Neural Networks

Yunpeng Zhu



Neural Networks

Edition: v1.1

Table of Contents

1 Introduction of Neural Networks

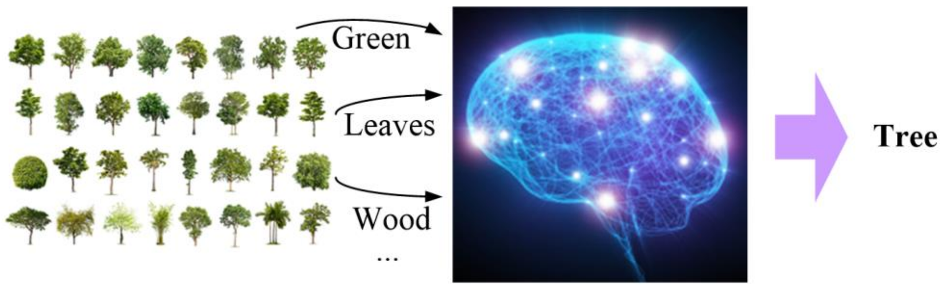
1.1 Introduction

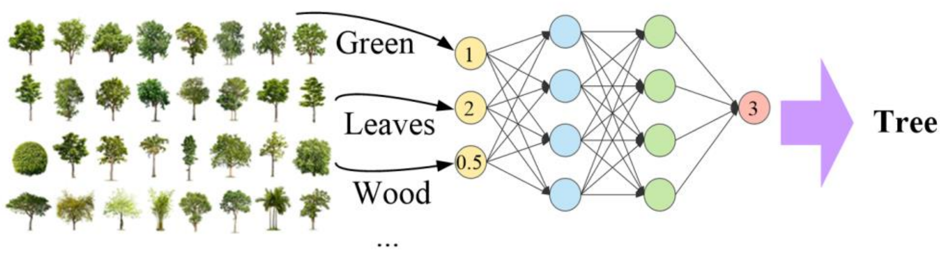
These are **TREES**.

What are these?

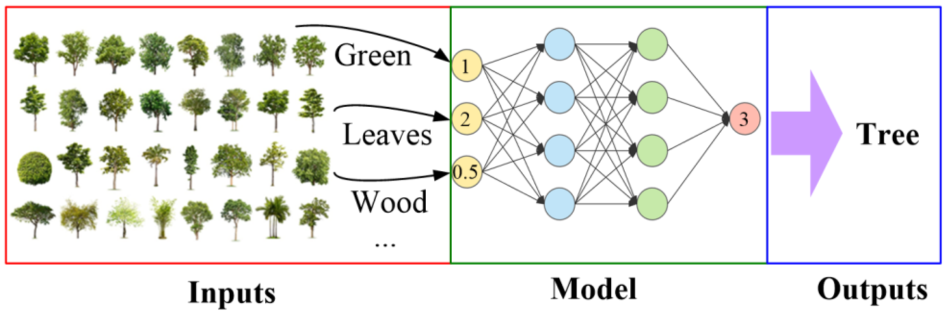
How do we learn these pictures?

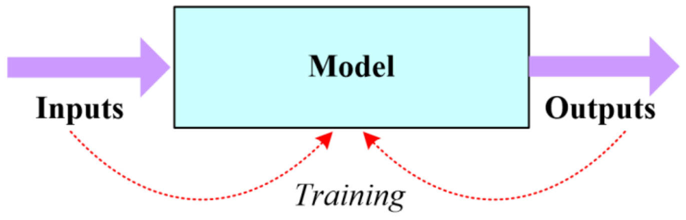
How to teach a computer to learn these pictures?

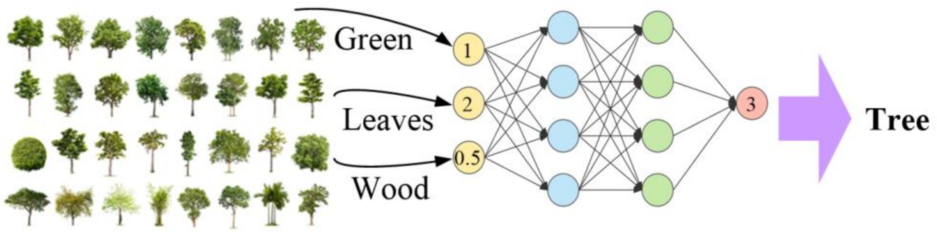


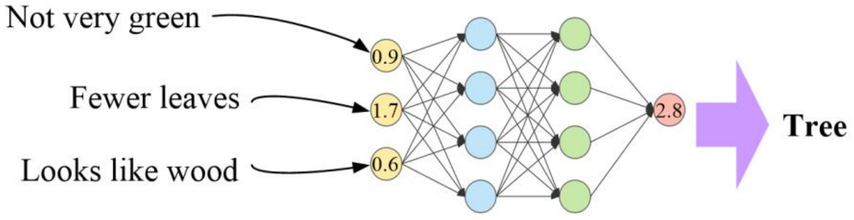


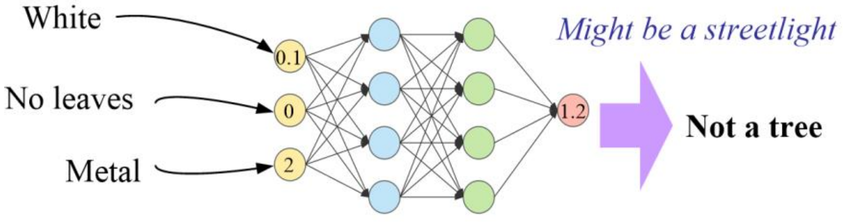
The capacity of a neural network is guaranteed by those inter-connections. We need to train the neural network to learn those features.



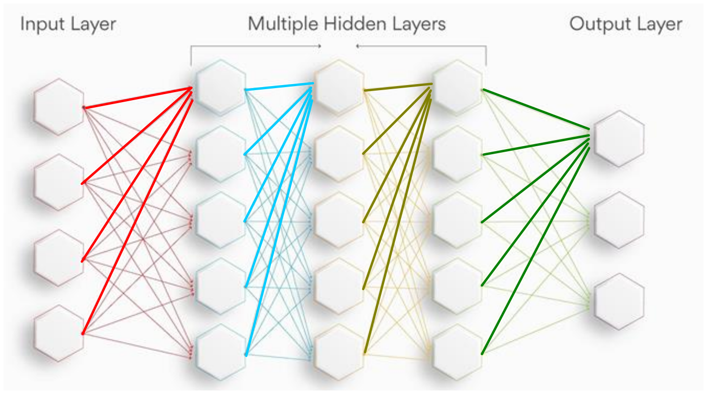




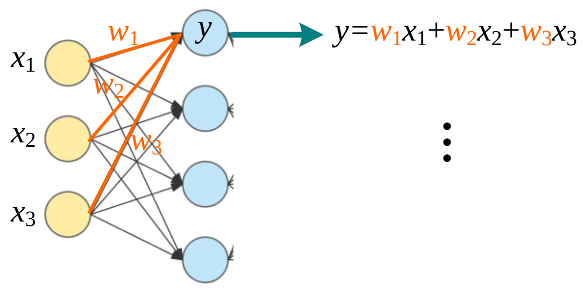




1.2 Structure of neural networks

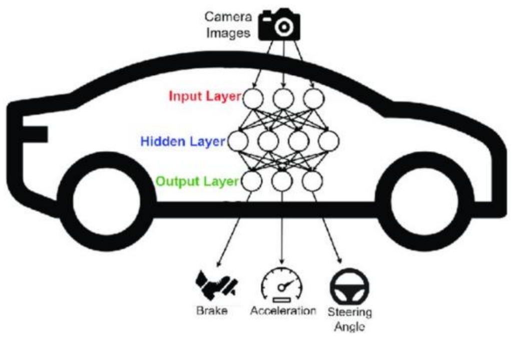


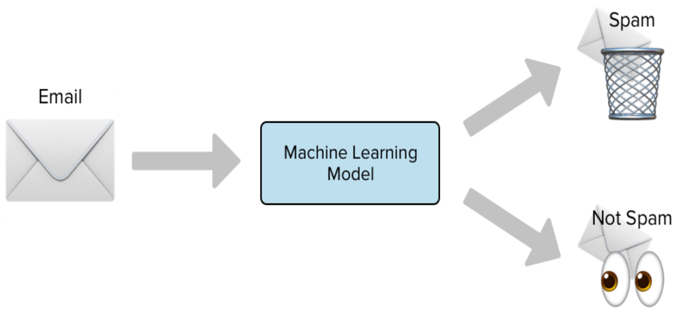
The first layer is the input layer. The middle layers are hidden layers. The final layer is the output layer. In neural networks, the circles are nodes or neurons. Neurons are connected by weights (numbers). We don't know what happens between the input and output layers (no physical meanings), so a neural network is a **black box** model.



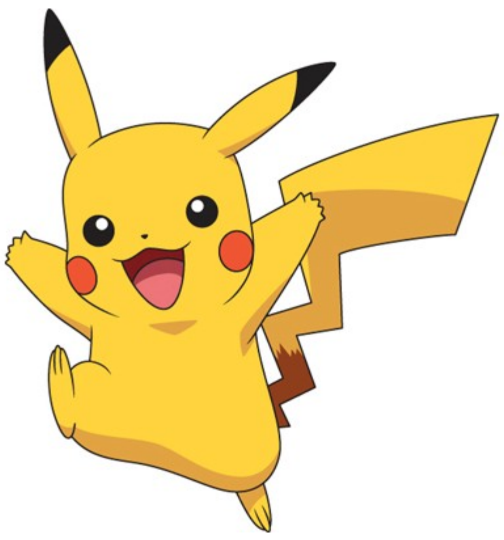
You can mathematically describe a neural network, but it is still *black* to people: why these functions work?

Actually, for deep neural networks, there can be **many** hidden layers. There can also be **thousands** of neurons in each layer. In many cases, people don't care about what happens inside a model. They only care about input and output.



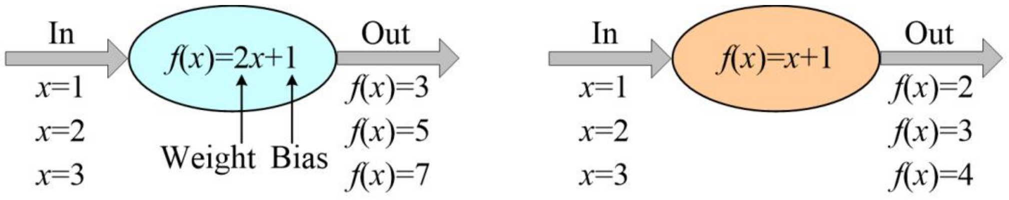


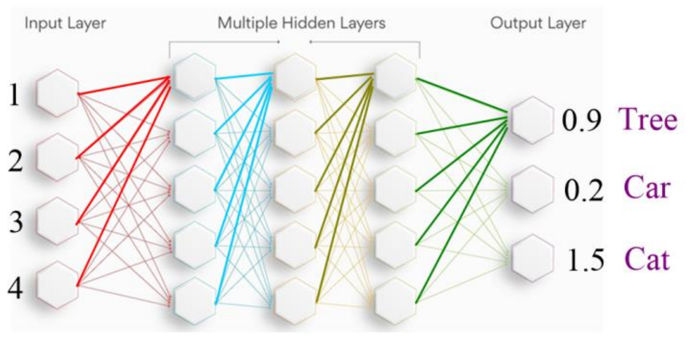
Open discussion: Can we build a very big neural network with trillions of neuron and layers, so that it can do anything?



How neural network works? – Functions

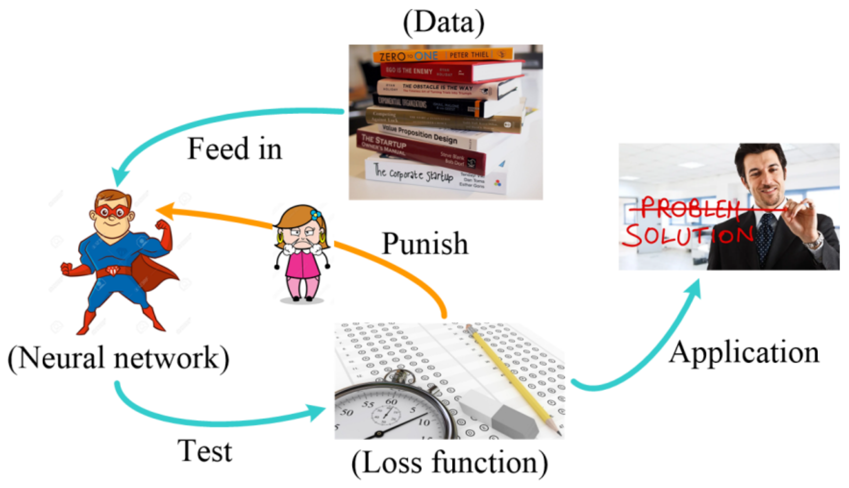
Unfortunately, we need math.





How do we train a neural network? – Similar to the way you learn.

Training a neural network is to determine all **weights** in the network.



The simplest neural network is called the **Perceptron**. It has multiple inputs, one neuron, one output.

2 Simple perceptron

2.1 Neuron and perceptron

Perceptron (Neuron) 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.

The basic structure of a perceptron is illustrated in Fig.1.

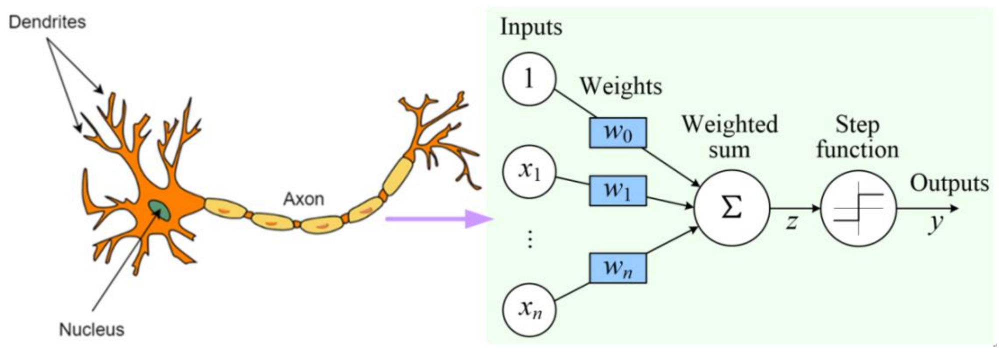


Fig.1 Neuron and perceptron

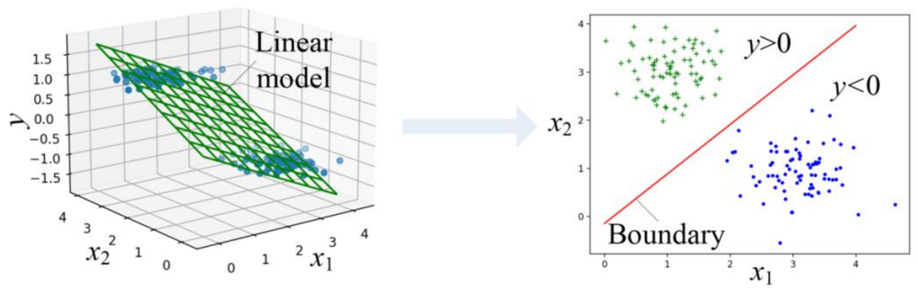
A perceptron can be written as

$$\begin{cases} z = w_0 + \sum_{i=1}^{n} w_i x_i = w_0 + \bar{\mathbf{x}}^{\mathrm{T}} \bar{\mathbf{w}} \\ y = f(z) \end{cases}$$

where $\bar{\mathbf{x}} = [x_1, \ldots, x_n]^{\mathrm{T}}$ and $\bar{\mathbf{w}} = [w_1, \ldots, w_n]^{\mathrm{T}}$, $f(z)$ is a step function

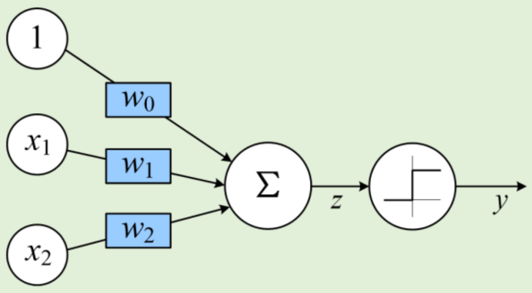
$$f(z) = \begin{cases} 1 & z > 0 \\ -1 & z \le 0 \end{cases}$$

In this case, the perceptron is exactly the same as a **linear binary classifier**, i.e.



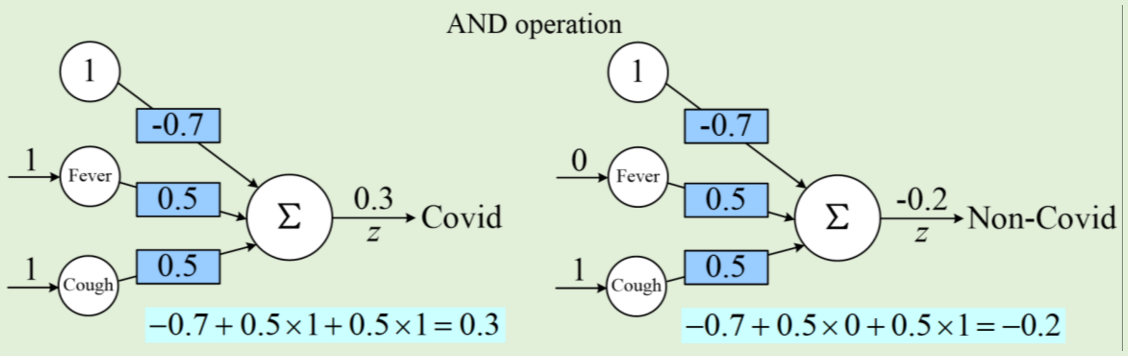
$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

Example: Application of perceptron

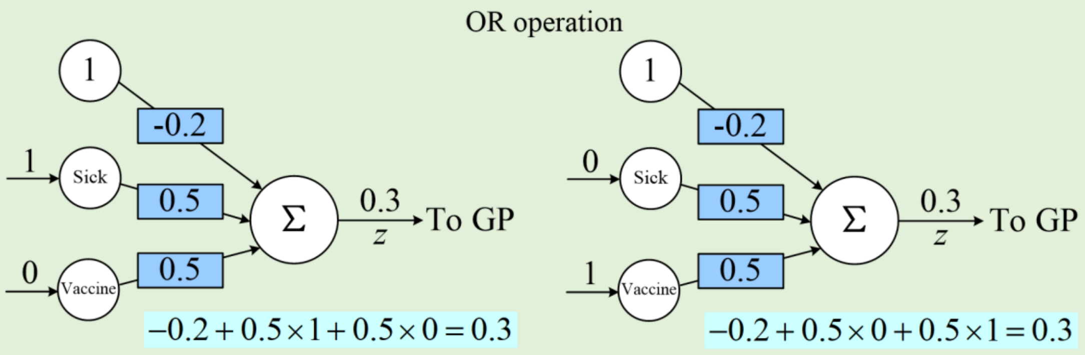


$$y = \begin{cases} 1 & w_0 + w_1 x_1 + w_2 x_2 > 0 \\ -1 & w_0 + w_1 x_1 + w_2 x_2 \le 0 \end{cases}$$

Large bias:



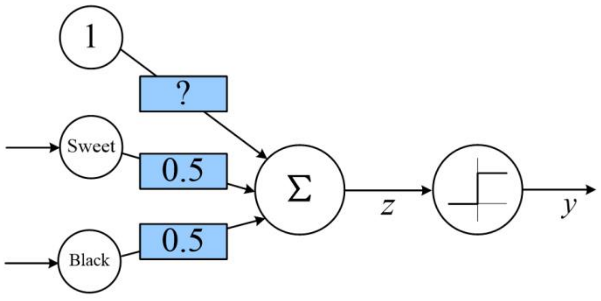
small bias:



Bias allows you to **control** the activation of the perceptron by adding a constant to the input.

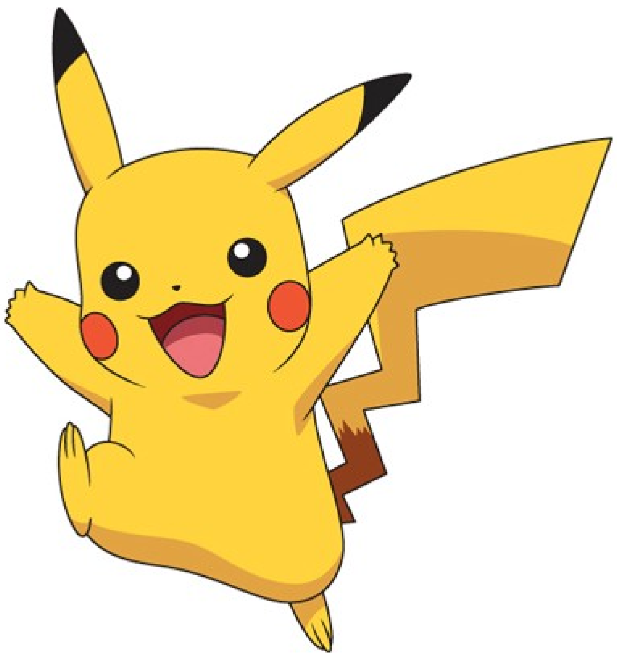
Quiz 2.1:

There is a white and a black chocolate. How does a robot separate them?



Optional bias: -0.1, -0.2, -0.6, 0.3.

Please determine the inputs and bias, and verify the results.

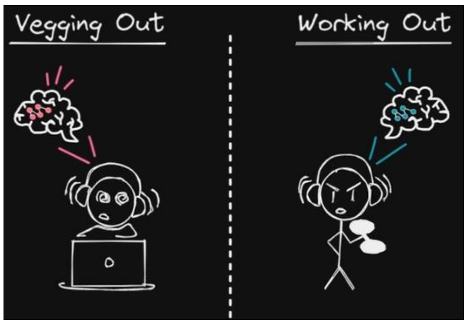


2.2 Activation functions

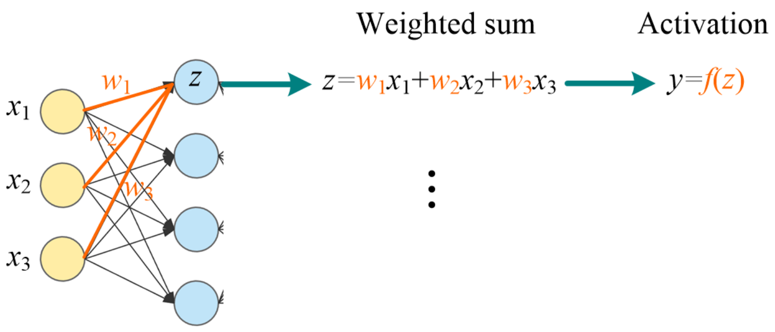
An Activation Function $f(z)$ decides whether a neuron/perceptron should be activated or not. The role of the Activation Function is to derive output from a set of input values. For example:

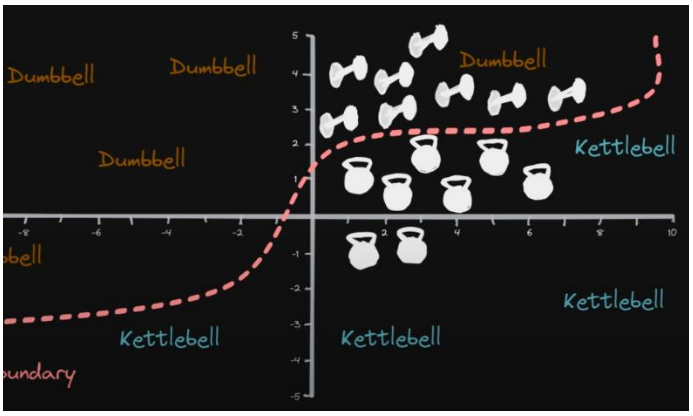
$$f(z)=\begin{cases} 1 & z>0 \\ -1 & z\le 0 \end{cases}$$

When you work on different missions, your brain **activates** different neurons.

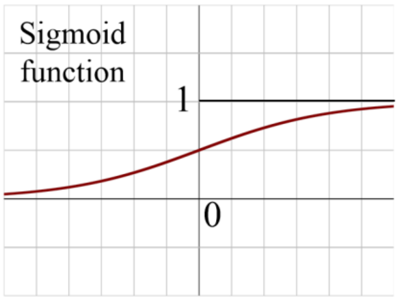


Activation Functions are often nonlinear, and there are various types of activation functions applied in practice [1,2]. If we don't use activation function, the relationships between inputs and outputs of a neural network are **linear**.

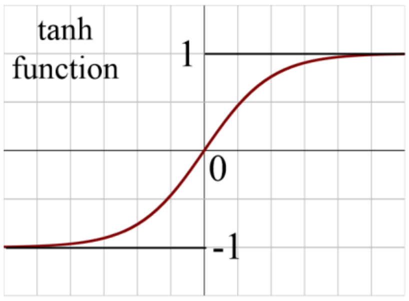




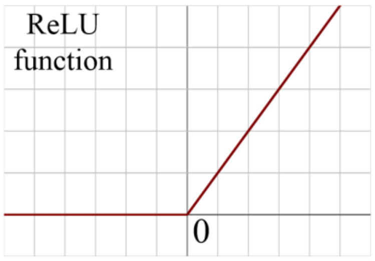
The step function we introduced above is an activation function. Besides, the sigmoid function, hyperbolic tangent (tanh) function, and Rectified linear unit (ReLU) function are also commonly used activation functions.



$$\begin{cases} f(z) = \dfrac{1}{1+\exp(-z)} \\ \dfrac{\mathrm{d}f(z)}{\mathrm{d}z} = f(z)\left[1 - f(z)\right] \end{cases}$$



$$\begin{cases} f(z) = \dfrac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} \\ \dfrac{\mathrm{d}f(z)}{\mathrm{d}z} = 1 - f(z)^2 \end{cases}$$



$$\begin{cases} f(z) = \begin{cases} 0 & z \le 0 \\ z & z > 0 \end{cases} = \max(0, z) \\ \dfrac{\mathrm{d}f(z)}{\mathrm{d}z} = \begin{cases} 0 & z \le 0 \\ 1 & z > 0 \end{cases} \end{cases}$$

We prefer activation function being differentiable. This is because we will use their derivations (gradients) in network training.

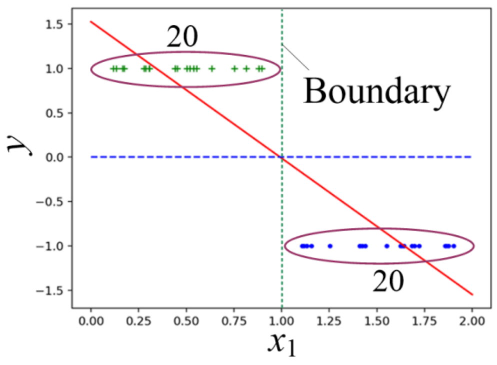
2.3 Determination of perceptron

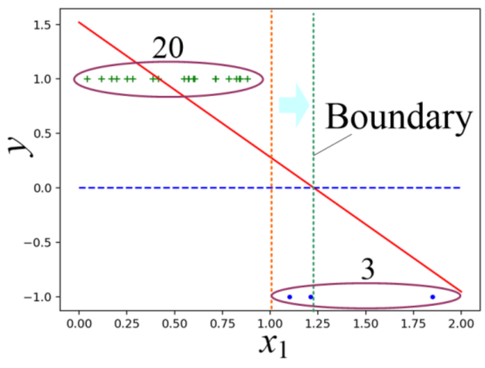
(1) Linear regression method

Considering the activation function is the **step function**, the perceptron is a binary linear classifier, which can be determined by using the linear regression approach, where the samples are labeled either true (1) or false (-1):

$$y = \begin{cases} 1 & \text{True} \\ -1 & \text{False} \end{cases}$$

where y represents the label.



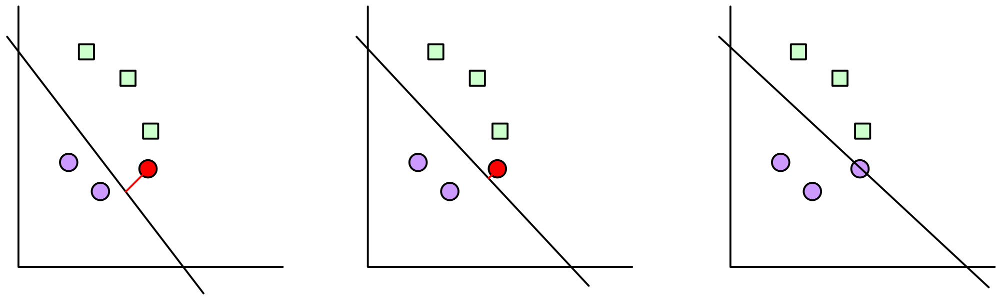


However, linear regression based classification may not always find the classification boundary.

(2) Gradient descent method

A new algorithm can be applied to solve the above issue for a **linearly separable** classification problem.

The aim of determining a perceptron is to minimize the **distance** between the **wrongly classified points** and the boundary.

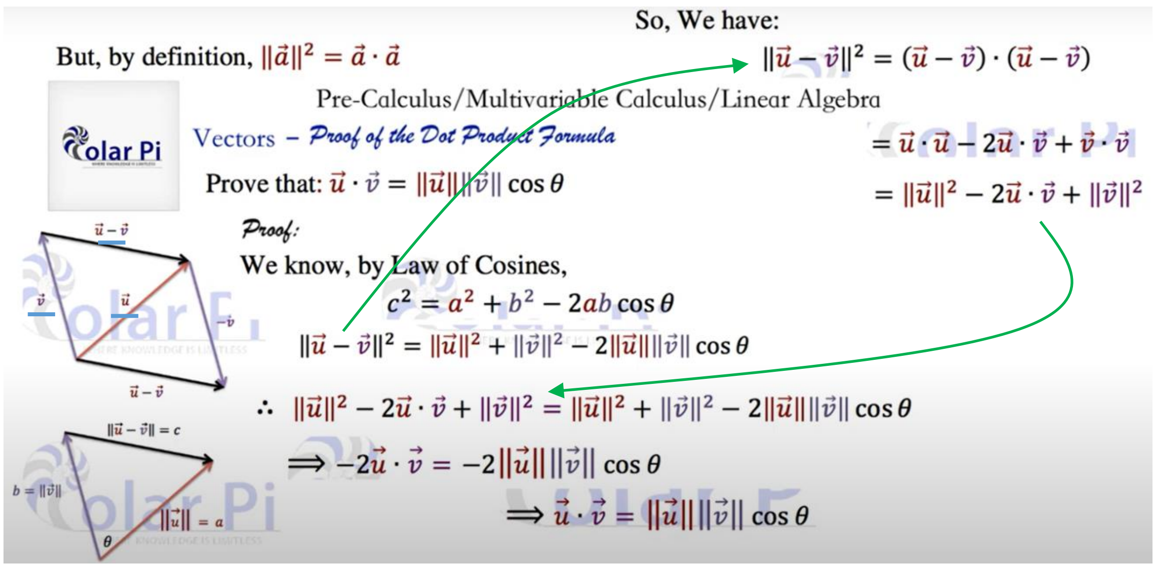


A) The distance from a point to the boundary [3]

Math: Considering two column vectors: $\mathbf{a}$ and $\mathbf{b}$. There are

✓ $\mathbf{a}\cdot\mathbf{b}=\mathbf{a}^{\mathrm{T}}\mathbf{b}=a_1b_1+a_2b_2+\cdots+a_nb_n$

✓ $\mathbf{a}\cdot\mathbf{b}=\mathbf{a}^{\mathrm{T}}\mathbf{b}=\|\mathbf{a}\|_2\|\mathbf{b}\|_2\cos\theta$



So, We have:

But, by definition, $\|\vec{a}\|^2=\vec{a}\cdot\vec{a}$

$\|\vec{u}-\vec{v}\|^2=(\vec{u}-\vec{v})\cdot(\vec{u}-\vec{v})$

Pre-Calculus/Multivariable Calculus/Linear Algebra

Colar Pi

Vectors – *Proof of the Dot Product Formula*

$=\vec{u}\cdot\vec{u}-2\vec{u}\cdot\vec{v}+\vec{v}\cdot\vec{v}$

Prove that: $\vec{u}\cdot\vec{v}=\|\vec{u}\|\|\vec{v}\|\cos\theta$

$=\|\vec{u}\|^2-2\vec{u}\cdot\vec{v}+\|\vec{v}\|^2$

Proof:

We know, by Law of Cosines,

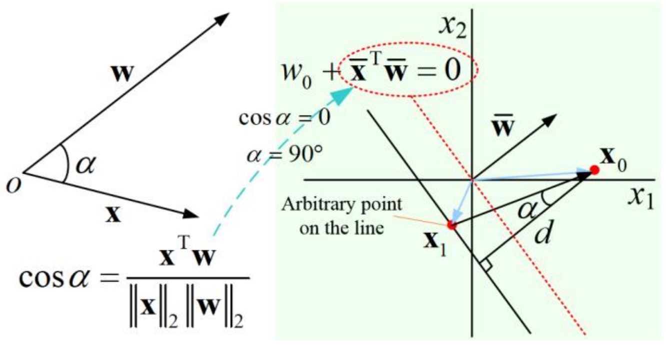
$$c^2=a^2+b^2-2ab\cos\theta$$

$$\|\vec{u}-\vec{v}\|^2=\|\vec{u}\|^2+\|\vec{v}\|^2-2\|\vec{u}\|\|\vec{v}\|\cos\theta$$

$$\therefore\ \|\vec{u}\|^2-2\vec{u}\cdot\vec{v}+\|\vec{v}\|^2=\|\vec{u}\|^2+\|\vec{v}\|^2-2\|\vec{u}\|\|\vec{v}\|\cos\theta$$

$$\Longrightarrow -2\vec{u}\cdot\vec{v}=-2\|\vec{u}\|\|\vec{v}\|\cos\theta$$

$$\Longrightarrow \vec{u}\cdot\vec{v}=\|\vec{u}\|\|\vec{v}\|\cos\theta$$



$$\cos\alpha=\frac{\mathbf{x}^{\mathrm{T}}\mathbf{w}}{\|\mathbf{x}\|_2\|\mathbf{w}\|_2}$$

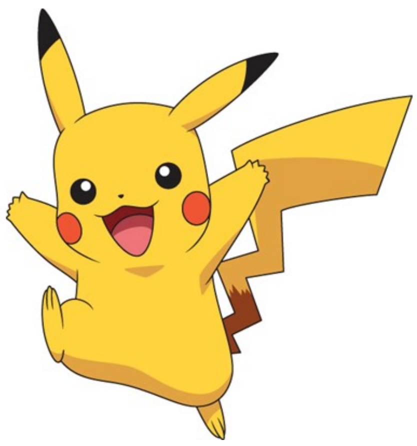
$$d=\|-\mathbf{x}_1+\mathbf{x}_0\|_2\cos\alpha$$

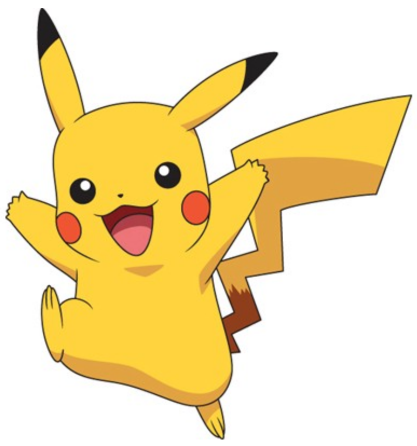
$$=\|-\mathbf{x}_1+\mathbf{x}_0\|_2\frac{(-\mathbf{x}_1+\mathbf{x}_0)^{\mathrm{T}}\overline{\mathbf{w}}}{\|-\mathbf{x}_1+\mathbf{x}_0\|_2\|\overline{\mathbf{w}}\|_2}$$

$$=\frac{w_0+\mathbf{x}_0{}^{\mathrm{T}}\overline{\mathbf{w}}}{\|\overline{\mathbf{w}}\|_2}$$

Quiz 2.2:

Find the distance from the point $(x_1,x_2)=(1,2)$ to the line $x_1+2x_2-3=0$





B) The wrongly classified points

The output label is

$$y = \begin{cases} 1 & w_0 + \overline{\mathbf{x}}^{\mathrm{T}}\overline{\mathbf{w}} > 0 \\ -1 & w_0 + \overline{\mathbf{x}}^{\mathrm{T}}\overline{\mathbf{w}} \le 0 \end{cases}$$

If a classification result is wrong, there is

$$\begin{cases} d \le 0,\ y = 1 \\ d > 0,\ y = -1 \end{cases} \Rightarrow -yd = |d|$$ (*Applied to find the wrongly classified points*)

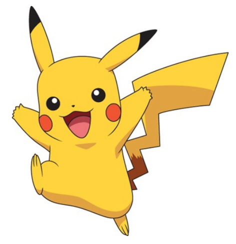
Quiz 2.3:

There are some points from 2 classes:

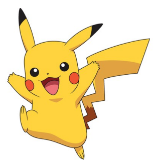
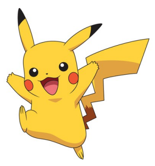
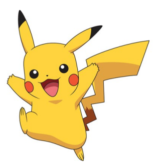
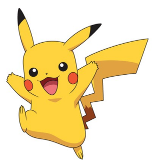
$y = 1: (x_1, x_2) = (1,1), (0.6, 0.8), (0.2, 0.7)$

$y = -1: (x_1, x_2) = (0,0), (0.5, 0.2), (0, 0.1)$

Find out the wrongly classified points by a classifier



$$y = \begin{cases} 1 & x_1 + x_2 - 0.5 > 0 \\ -1 & x_1 + x_2 - 0.5 \le 0 \end{cases}$$

C) The cost function

The total distance of the wrongly classified points can be obtained as

$$D = -\frac{1}{\|\overline{\mathbf{w}}\|_2} \sum_{\mathbf{x}_{(j)} \in \mathbf{M}} y_{(j)} \left[w_0 + \overline{\mathbf{x}}_{(j)}^{\mathrm{T}} \overline{\mathbf{w}} \right]$$

where $\mathbf{M}$ is a set of all wrongly classified points.

We want to minimize the distance D, thus the cost function can be formulated as

$$J(\overline{\mathbf{w}}, w_0) = -\sum_{\mathbf{x}_{(j)} \in \mathbf{M}} y_{(j)} \left(w_0 + \overline{\mathbf{x}}_{(j)}^{\mathrm{T}} \overline{\mathbf{w}} \right)$$

Quiz 2.4:

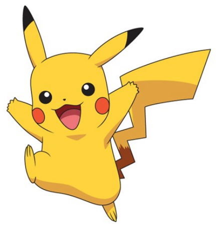
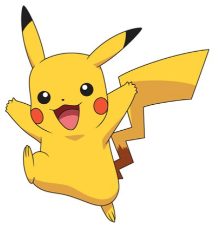
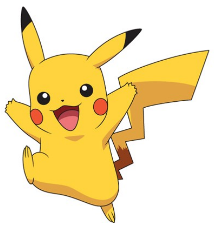
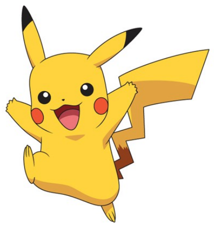
Considering a boundary function:

$$y = 2x_1 + 3x_2 - 1$$

Find out the wrongly classified points:

Negative: $(x_1, x_2) = (0, -1), (-2, 1), (0, 0.5)$

Positive: $(x_1, x_2) = (0, 1), (-1, 0), (0.3, 0.6)$

D) The gradient descent method

The gradient of the cost function can be obtained as

$$\frac{\partial J\left(\overline{\mathbf{w}}, w_0\right)}{\partial \overline{\mathbf{w}}} = -\sum_{\mathbf{x}_{(j)} \in \mathbf{M}} y_{(j)} \overline{\mathbf{x}}_{(j)}$$

$$\frac{\partial J\left(\overline{\mathbf{w}}, w_0\right)}{\partial w_0} = -\sum_{\mathbf{x}_{(j)} \in \mathbf{M}} y_{(j)}$$

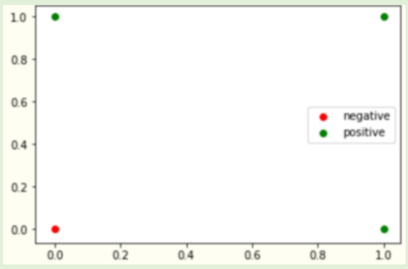
- Update weights by using the **gradient descent method**

$$\begin{cases} \overline{\mathbf{w}}_{\text{new}} = \overline{\mathbf{w}}_{\text{old}} - \lambda\left(-\sum_{\mathbf{x}_{(j)} \in \mathbf{M}} y_{(j)} \overline{\mathbf{x}}_{(j)}\right) = \overline{\mathbf{w}}_{\text{old}} + \lambda \sum_{\mathbf{x}_{(j)} \in \mathbf{M}} y_{(j)} \overline{\mathbf{x}}_{(j)} \\ w_{0,\text{new}} = w_{0,\text{old}} - \lambda\left(-\sum_{\mathbf{x}_{(j)} \in \mathbf{M}} y_{(j)}\right) = w_{0,\text{old}} + \lambda \sum_{\mathbf{x}_{(j)} \in \mathbf{M}} y_{(j)} \end{cases}, \lambda > 0$$

- Update weights by using the **stochastic gradient descent method**

$$\begin{cases} \overline{\mathbf{w}}_{\text{new}} = \overline{\mathbf{w}}_{\text{old}} - \lambda\left(-y_{(j)} \overline{\mathbf{x}}_{(j)}\right) = \overline{\mathbf{w}}_{\text{old}} + \lambda y_{(j)} \overline{\mathbf{x}}_{(j)} \\ w_{0,\text{new}} = w_{0,\text{old}} - \lambda\left(-y_{(j)}\right) = w_{0,\text{old}} + \lambda y_{(j)} \end{cases}, \lambda > 0 \text{ (update one by one)}$$

until all points are correctly classified.



Example: Application of perceptron

Considering we have 4 points (x_1, x_2): [(0,0), (0,1), (1,0), (1,1)], where (0,0) is labelled as negative ($y_{(1)} = -1$), and (0,1), (1,0), (1,1) are positive ($y_{(2)} = 1, y_{(3)} = 1, y_{(4)} = 1$, respectively).

We want to find a classifier (perceptron) to separate the positive and negative points. The perceptron can be represented as

$$z = w_0 + w_1 x_1 + w_2 x_2,\ y = \begin{cases} 1 & z > 0 \\ -1 & z \le 0 \end{cases}$$

Let $\lambda = 1$. Denote the initial weights are

$$w_0 = 0, \bar{\mathbf{w}} = [w_1, w_2]^{\mathrm{T}} = [0,0]^{\mathrm{T}}$$

a) By using **gradient descent method**

1- Substituting (0,0), (0,1), (1,0), (1,1) into $z = 0 + 0x_1 + 0x_2 = 0$, (0,1), (1,0), (1,1) are wrongly classified. According to the gradient descent method, there are

$$\begin{cases} w_{0,\text{new}} = w_{0,\text{old}} + \lambda(y_{(2)} + y_{(3)} + y_{(4)}) = 0 + 1\times(1+1+1) = 3 \\ w_{1,\text{new}} = w_{1,\text{old}} + \lambda(y_{(2)}x_{1,(2)} + y_{(3)}x_{1,(3)} + y_{(4)}x_{1,(4)}) \\ \quad = 0 + 1\times(1\times 0 + 1\times 1 + 1\times 1) = 2 \\ w_{2,\text{new}} = w_{2,\text{old}} + \lambda(y_{(2)}x_{2,(2)} + y_{(3)}x_{2,(3)} + y_{(4)}x_{2,(4)}) \\ \quad = 0 + 1\times(1\times 1 + 1\times 0 + 1\times 1) = 2 \end{cases}$$

2- Substituting (0,0), (0,1), (1,0), (1,1) into $z = 3 + 2x_1 + 2x_2$, (0,0) is wrongly classified. According to the gradient descent method, there are

$$\begin{cases} w_{0,\text{new}} = w_{0,\text{old}} + \lambda y_{(1)} = 3 + 1\times(-1) = 2 \\ w_{1,\text{new}} = w_{1,\text{old}} + \lambda y_{(1)}x_{1,(1)} = 2 + 1\times(-1\times 0) = 2 \\ w_{2,\text{new}} = w_{2,\text{old}} + \lambda y_{(1)}x_{2,(1)} = 2 + 1\times(-1\times 0) = 2 \end{cases}$$

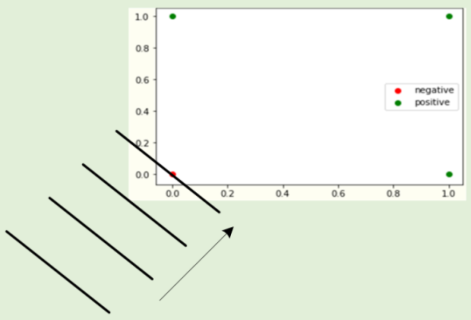
3- Substituting (0,0), (0,1), (1,0), (1,1) into $z = 2 + 2x_1 + 2x_2$, (0,0) is wrongly classified. According to the gradient descent method, there are

$$\begin{cases} w_{0,\text{new}} = w_{0,\text{old}} + \lambda y_{(1)} = 2 + 1\times(-1) = 1 \\ w_{1,\text{new}} = w_{1,\text{old}} + \lambda y_{(1)} x_{1,(1)} = 2 + 1\times(-1\times 0) = 2 \\ w_{2,\text{new}} = w_{2,\text{old}} + \lambda y_{(1)} x_{2,(1)} = 2 + 1\times(-1\times 0) = 2 \end{cases}$$

4- Substituting (0,0), (0,1), (1,0), (1,1) into $z = 1 + 2x_1 + 2x_2$, (0,0) is wrongly classified. According to the gradient descent method, there are

$$\begin{cases} w_{0,\text{new}} = w_{0,\text{old}} + \lambda y_{(1)} = 1 + 1\times(-1) = 0 \\ w_{1,\text{new}} = w_{1,\text{old}} + \lambda y_{(1)} x_{1,(1)} = 2 + 1\times(-1\times 0) = 2 \\ w_{2,\text{new}} = w_{2,\text{old}} + \lambda y_{(1)} x_{2,(1)} = 2 + 1\times(-1\times 0) = 2 \end{cases}$$

5- Substituting (0,0), (0,1), (1,0), (1,1) into $z = 0 + 2x_1 + 2x_2$, all points are correctly classified.



b) By using **stochastic gradient descent method**

1-1 Substituting (0,0) into $z = 0 + 0x_1 + 0x_2 = 0$, correctly classified.

1-2 Substituting (0,1) into $z = 0 + 0x_1 + 0x_2 = 0$, wrongly classified. According to the stochastic gradient descent method, there are

$$\begin{cases} w_{0,\text{new}} = w_{0,\text{old}} + \lambda y_{(2)} = 0 + 1\times 1 = 1 \\ w_{1,\text{new}} = w_{1,\text{old}} + \lambda y_{(2)} x_{1,(2)} = 0 + 1\times 1\times 0 = 0 \\ w_{2,\text{new}} = w_{2,\text{old}} + \lambda y_{(2)} x_{2,(2)} = 0 + 1\times 1\times 1 = 1 \end{cases}$$

1-3 Substituting (1,0) into $z = 1 + 0x_1 + 1x_2$, correctly classified.

1-4 Substituting (1,1) into $z = 1 + 0x_1 + 1x_2$, correctly classified.

2-1 Substituting (0,0) into $z = 1 + 0x_1 + 1x_2$, wrongly classified. According to the stochastic gradient descent method, there are

$$\begin{cases} w_{0,\text{new}} = w_{0,\text{old}} + \lambda y_{(1)} = 1 + 1\times(-1) = 0 \\ w_{1,\text{new}} = w_{1,\text{old}} + \lambda y_{(1)} x_{1,(1)} = 0 + 1\times(-1)\times 0 = 0 \\ w_{2,\text{new}} = w_{2,\text{old}} + \lambda y_{(1)} x_{2,(1)} = 1 + 1\times(-1)\times 0 = 1 \end{cases}$$

2-2 Substituting (0,1) into $z = 0 + 0x_1 + 1x_2$, correctly classified.

2-3 Substituting (1,0) into $z = 0 + 0x_1 + 1x_2$, wrongly classified. According to the stochastic gradient descent method, there are

$$\begin{cases} w_{0,\text{new}} = w_{0,\text{old}} + \lambda y_{(1)} = 0 + 1\times 1 = 1 \\ w_{1,\text{new}} = w_{1,\text{old}} + \lambda y_{(1)} x_{1,(1)} = 0 + 1\times 1\times 1 = 1 \\ w_{2,\text{new}} = w_{2,\text{old}} + \lambda y_{(1)} x_{2,(1)} = 1 + 1\times 1\times 0 = 1 \end{cases}$$

2-4 Substituting (1,1) into $z = 1 + 1x_1 + 1x_2$, correctly classified.

3-1 Substituting (0,0) into $z = 1 + 1x_1 + 1x_2$, wrongly classified. According to the stochastic gradient descent method, there are

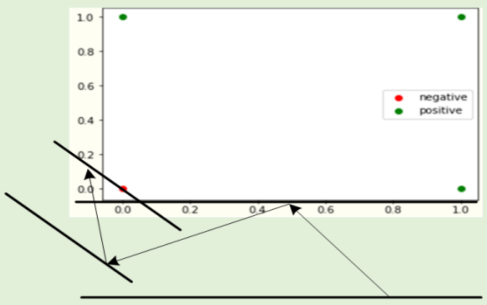
$$\begin{cases} w_{0,\text{new}} = w_{0,\text{old}} + \lambda y_{(1)} = 1 + 1\times(-1) = 0 \\ w_{1,\text{new}} = w_{1,\text{old}} + \lambda y_{(1)} x_{1,(1)} = 1 + 1\times(-1)\times 0 = 1 \\ w_{2,\text{new}} = w_{2,\text{old}} + \lambda y_{(1)} x_{2,(1)} = 1 + 1\times(-1)\times 0 = 1 \end{cases}$$

3-2 Substituting (0,1) into $z = 0 + 1x_1 + 1x_2$, correctly classified.

3-3 Substituting (1,0) into $z = 0 + 1x_1 + 1x_2$, correctly classified.

3-4 Substituting (1,1) into $z = 0 + 1x_1 + 1x_2$, correctly classified.

4 All points are correctly classified



Quiz 2.5:

Considering we have 2 points (x_1, x_2): [(1,1), (2,2)], where (1,1) is labelled as Negative ($y_{(1)} = -1$), and (2,2) is Positive ($y_{(2)} = 1$).

Find a perceptron to separate the positive and negative points.