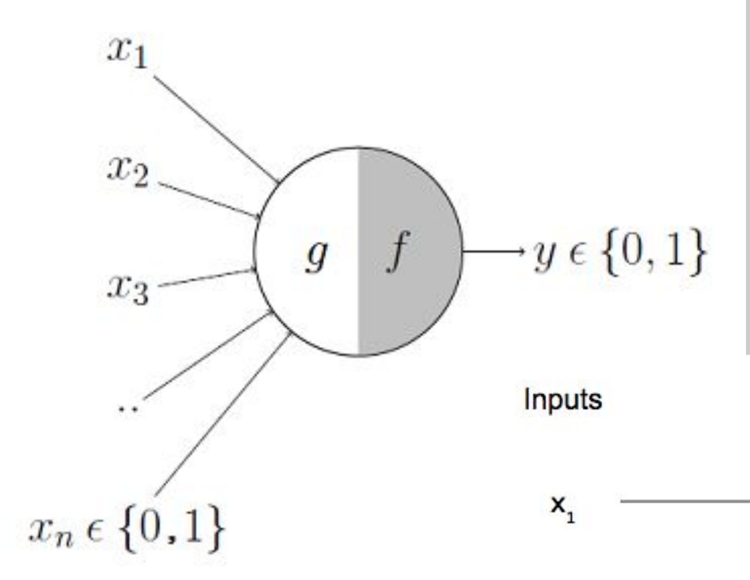
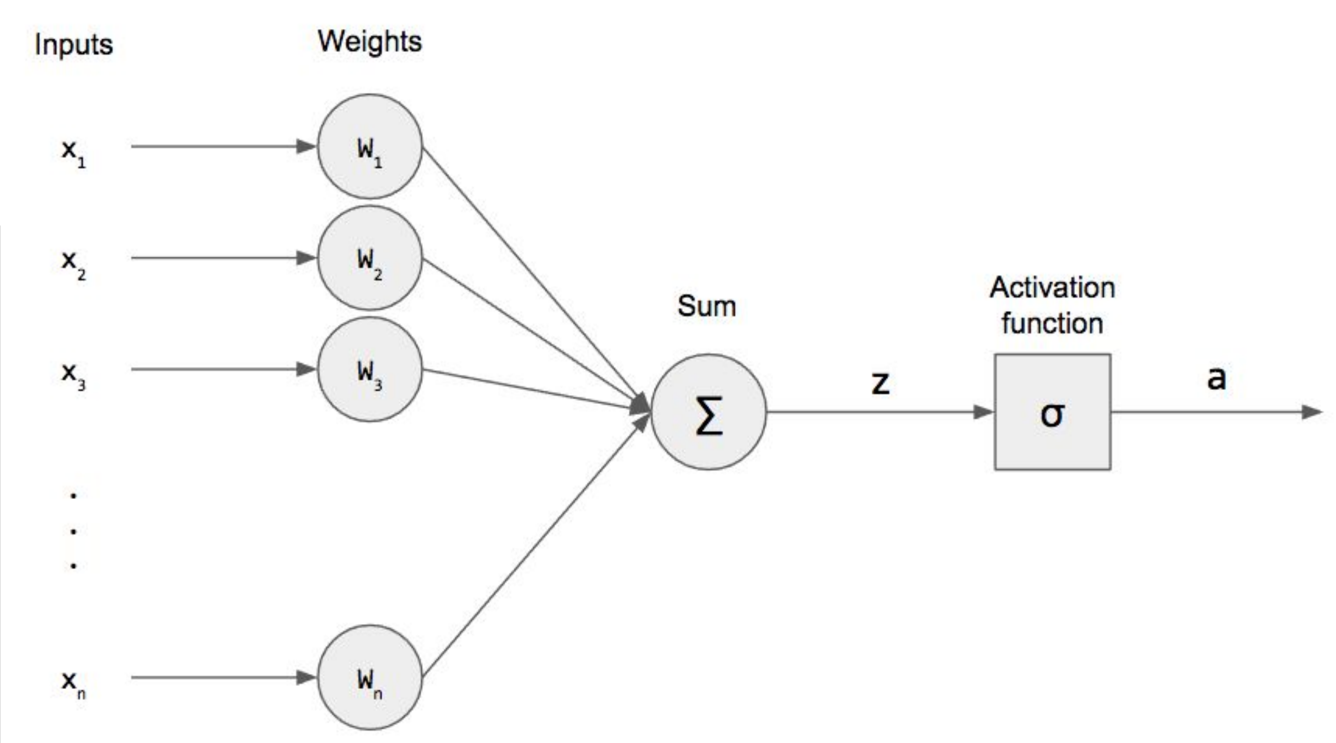
Perceptrons, and Artificial neural networks

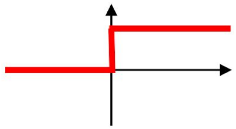
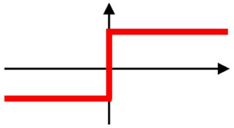
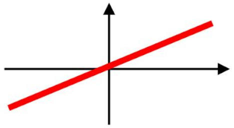
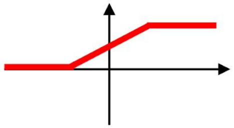
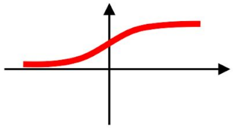
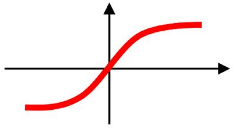
Chao Huang

McCulloch-Pitts Neuron and Perceptron



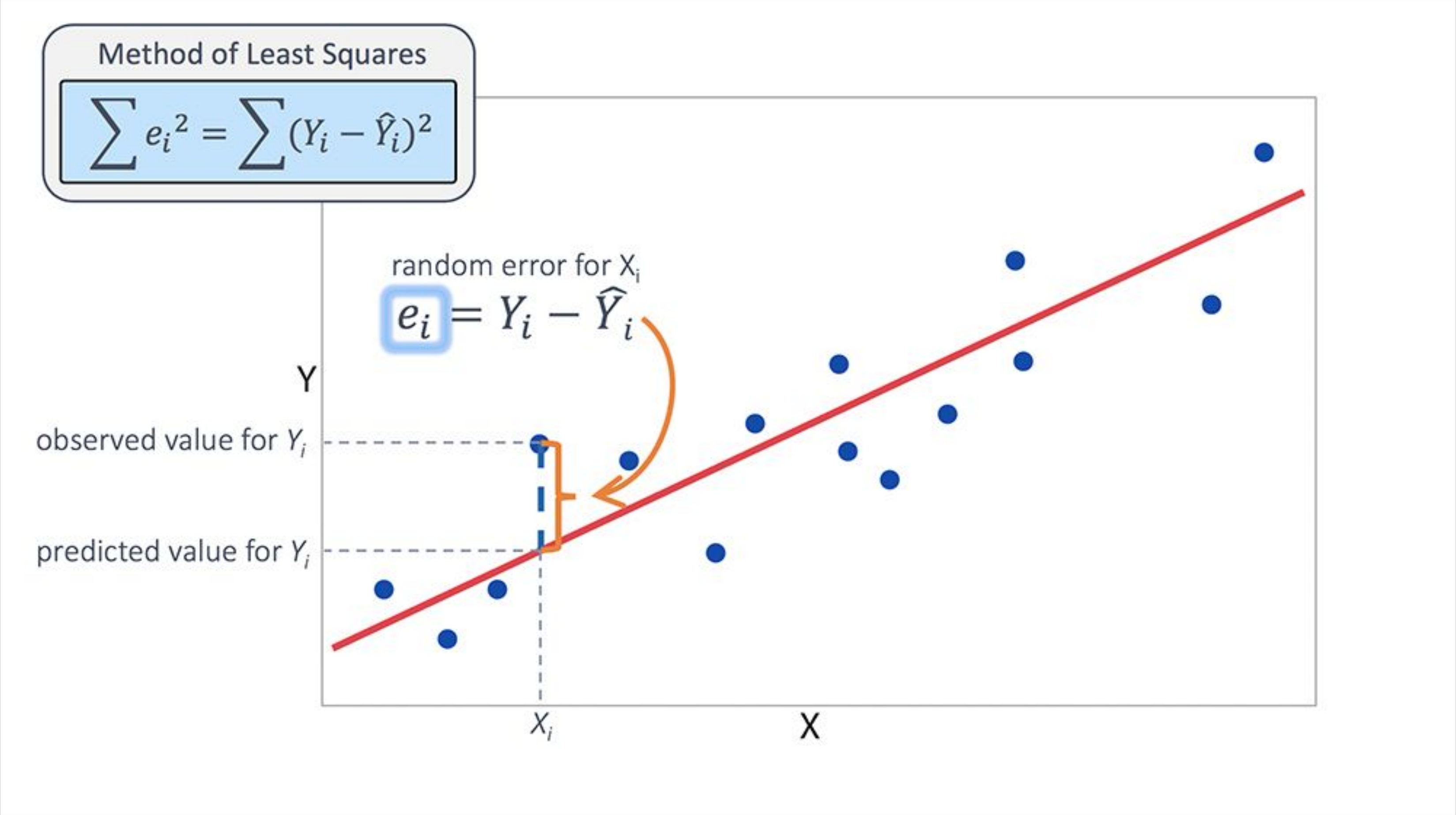


Activation function of the Perceptron

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer NN	

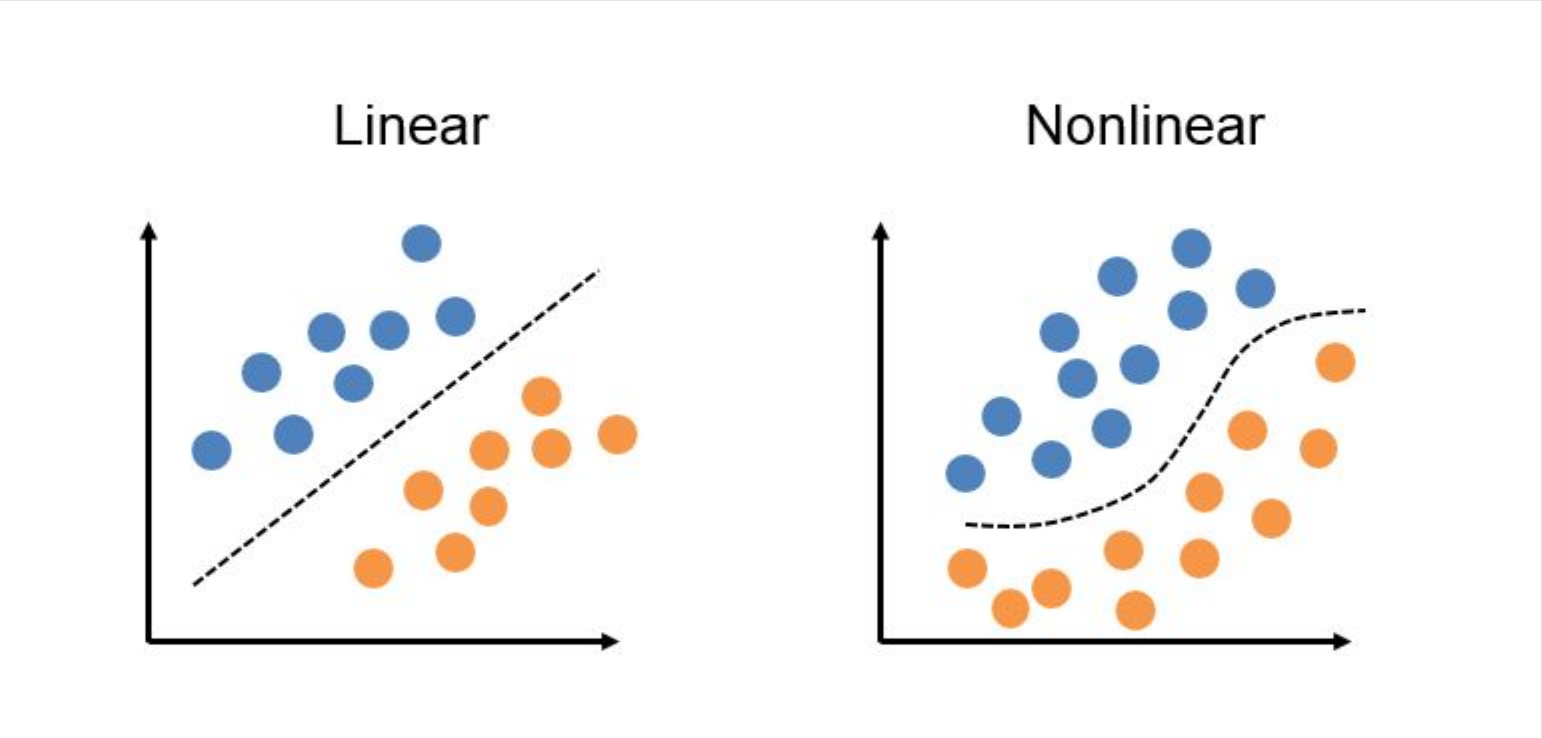
Regression problems

E.g. linear model fitting

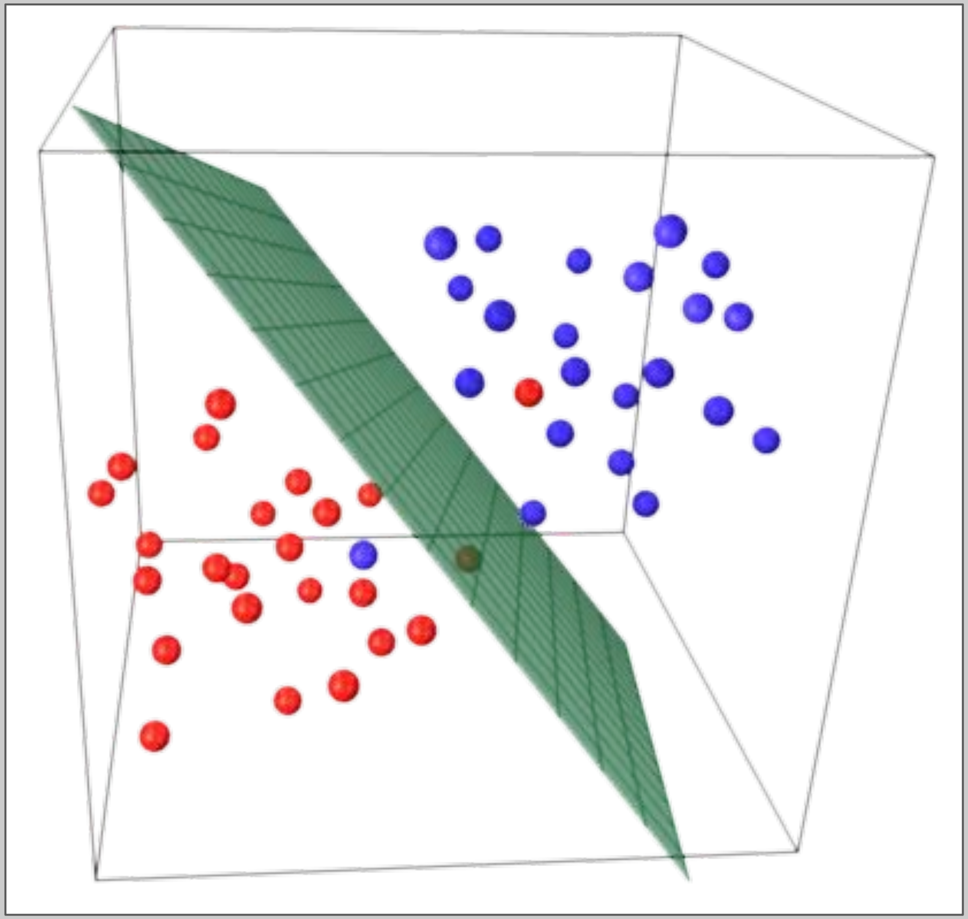


Classification problems

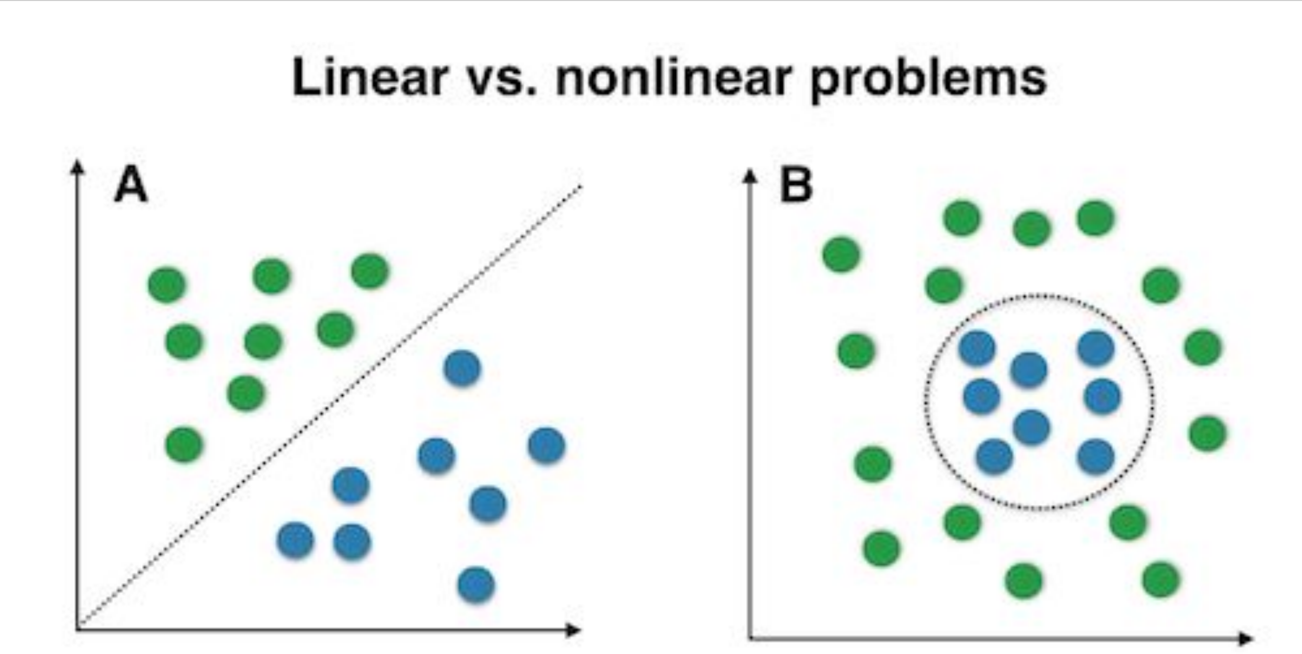
E.g. spam email detection, face detection, auto-driving cars, ...



Perceptron as a linear classifier



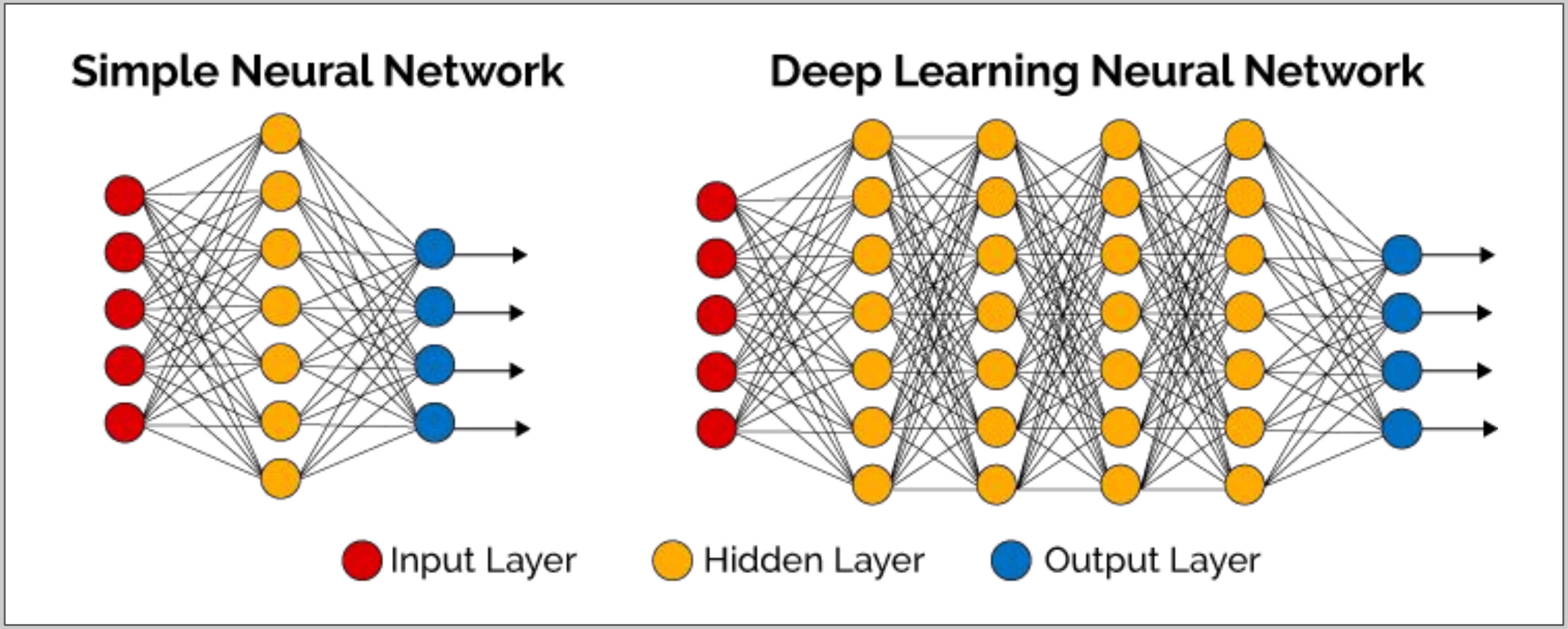
Perceptron as a linear classifier



Non-linear problems?

The Universal Approximation Theorem

...artificial neural networks with hidden layers can approximate continuous functions with finite number of neurons, under mild assumptions on the activation function



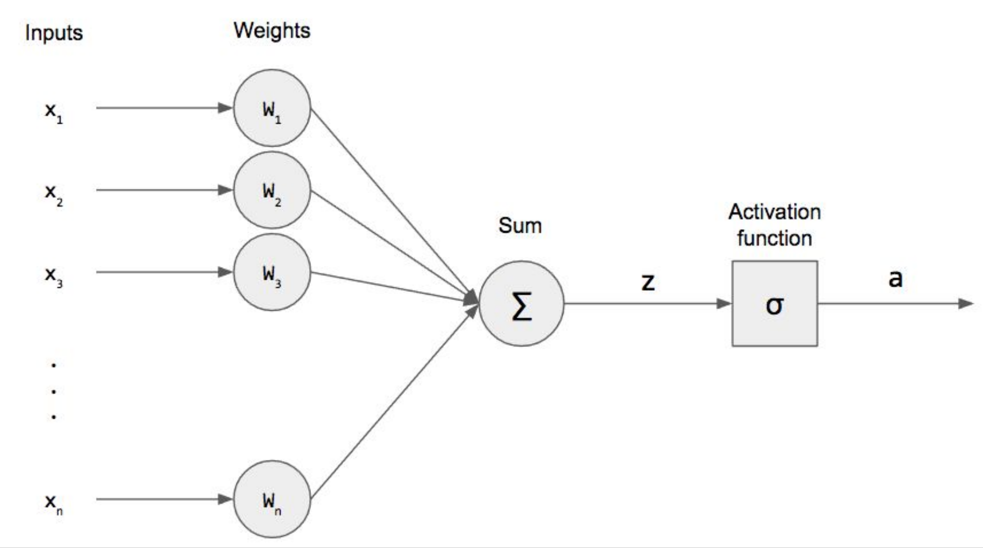
Training the model

Task:

Find the appropriate weights ***w*** so the difference between the model output and the data is minimized

How?

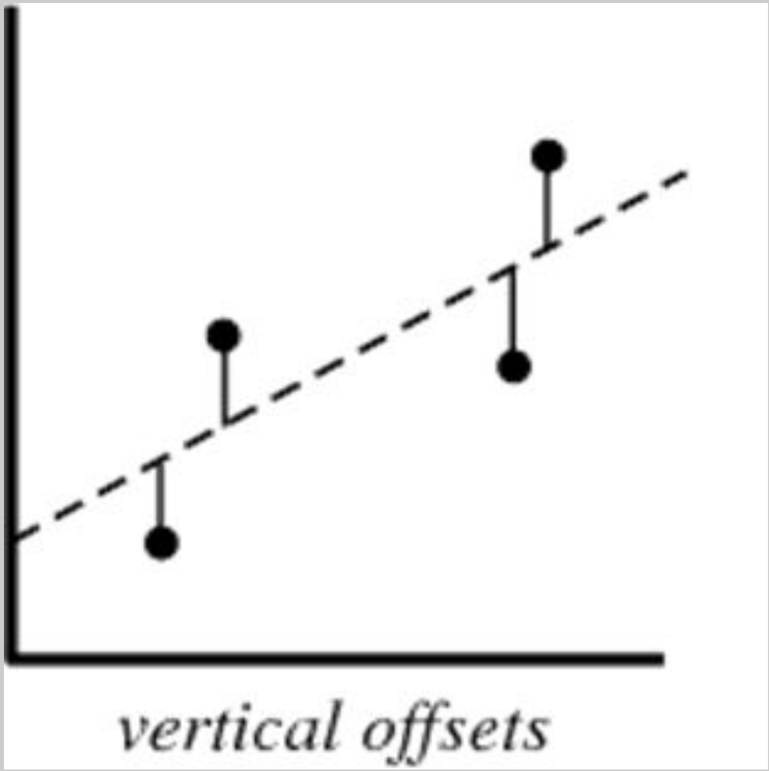
Need a quantitative measure of the difference as a function of the weights: the loss function



The Loss function(s)

Mean square error (MSE)

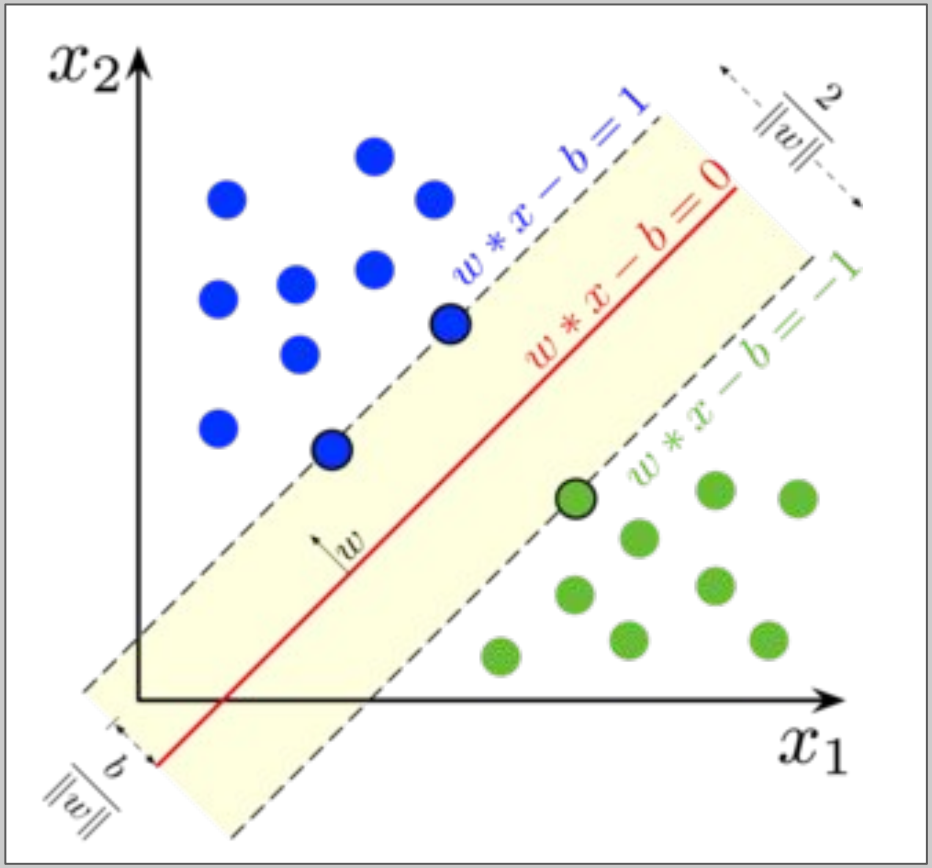
$$MSE = \frac{1}{n} \Sigma \underbrace{\left(y - \widehat{y} \right)}_{\text{The square of the difference between actual and predicted}}^2$$



The Loss function(s)

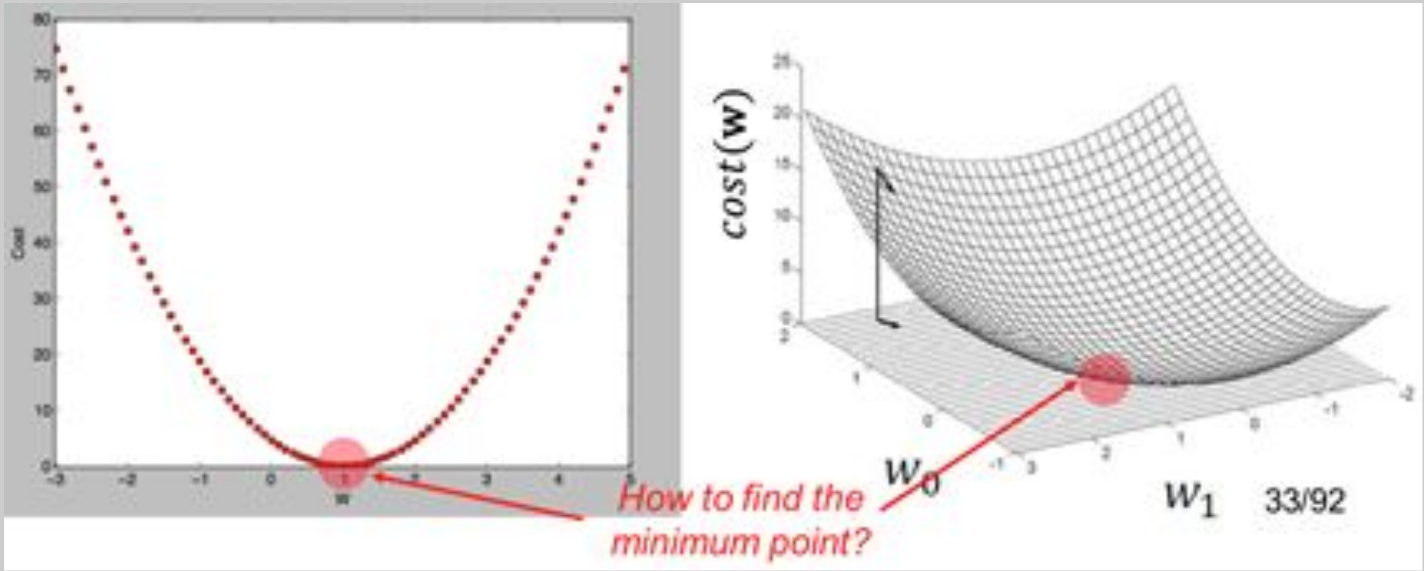
Hinge loss

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(w \cdot x_i - b)\right)\right] + \lambda\|w\|^2.$$



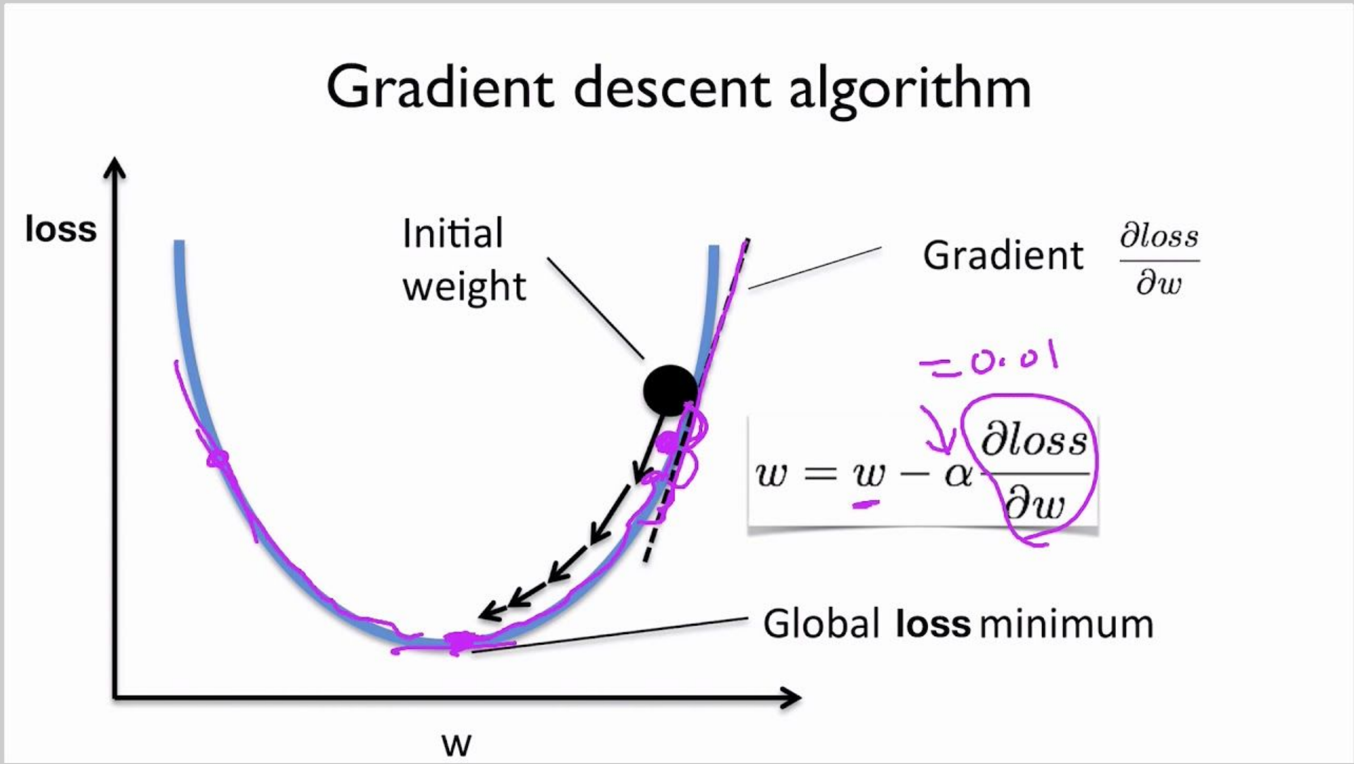
Training the model

How to find the correct weights ***w*?**



Training the model

How to find the correct weights ***w*?**



Training the model

How to find the correct weights ***w*?**

1. Start with initial guesses
   - Start at random value
2. Each weight is updated by taking a step into the opposite direction of the gradient $\Delta w_i = -\eta \times \frac{\partial E}{\partial w_i}$
   - Compute the partial derivative of the cost function $\frac{\partial E}{\partial w_i}$ for each weight
3. Repeat until you converge to a local minimum