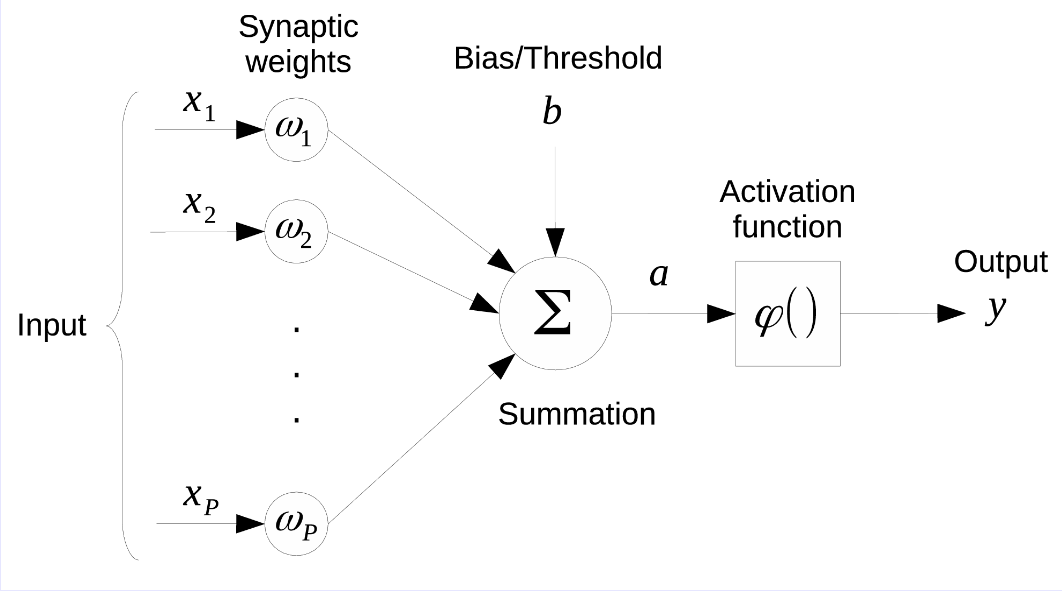
The perceptron - details



It is just a function

$$y = \varphi(\mathbf{x}, \boldsymbol{\omega})$$

We have data, typically called a training dataset!

$$\left\{ \mathbf{x}_n , d_n \right\}_{n=1,\dots,N}$$

Inputs

Targets

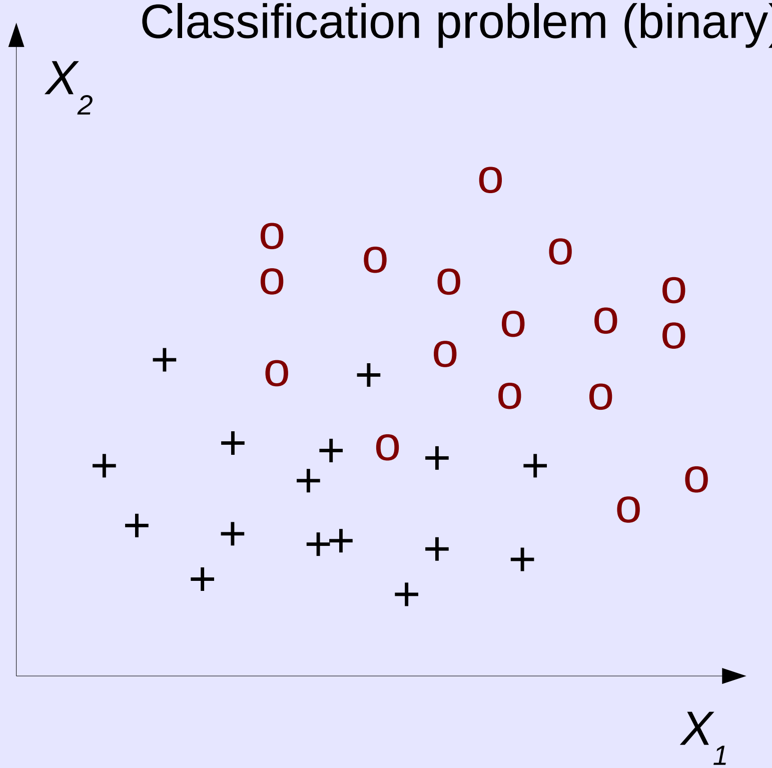
An example (MNIST database)!

Inputs = 28 x 28 grayscale images

Targets = {‘0’,’1’,’2’,’3’,’4’,’5’,’6’,’7’,’8’,’9’}

We are going to use simpler datasets, that are easier to illustrate

Classification problem (binary)



$$\mathbf{x}_n = (x_{n1}, x_{n2})$$

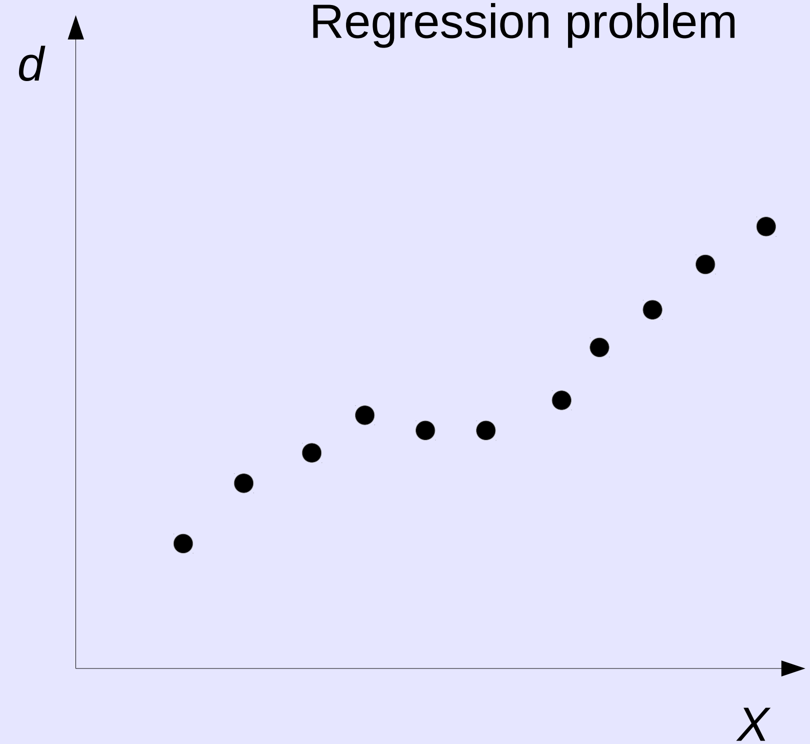
$$d_n = \{0, 1\}$$

Binary target

Class ‘+’

Class ‘o’

Regression problem



$$x_n$$

$$d_n$$

Continuous target

So, we have

$$y(\mathbf{x}_n) = \varphi(\mathbf{x}_n, \boldsymbol{\omega}) \qquad \left\{ \mathbf{x}_n, d_n \right\}$$

Perceptron

Training data

Task!

$$y_n = d_n \quad , \forall n$$

How?

Common approach, construct
an error function, e.g.

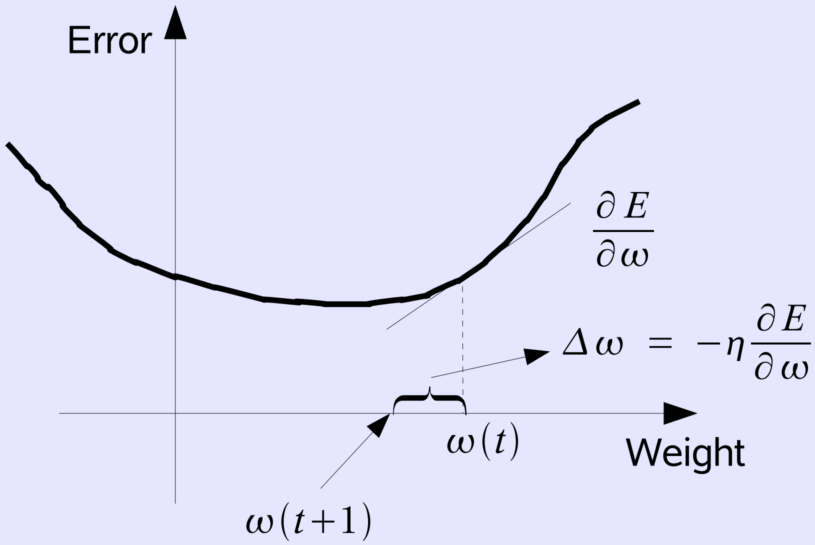
$$E(\boldsymbol{\omega}) = \frac{1}{N}\sum_{n=1}^{N}(y(\mathbf{x}_n) - d_n)^2$$

A function of the weights!

Minimizing *E(w)* means "solving" the task
(or at least an attempt to solve it)

How?

Minimize using gradient descent



$$\Delta\omega_i = -\eta\frac{\partial E(\boldsymbol{\omega})}{\partial\omega_i}$$

$$\Delta\omega_i = -\eta\frac{1}{N}\sum_n\frac{\partial E_n(\boldsymbol{\omega})}{\partial\omega_i}$$

Stochastic gradient descent (SGD)

$$\Delta\omega_i = -\eta\frac{1}{P}\sum_{p=1}^{P}\frac{\partial E_p(\boldsymbol{\omega})}{\partial\omega_i}$$

Where *P* is typically between 10-15
(randomly selected from the training data)

Finally!

$$y(\mathbf{x}_n) = \varphi(\mathbf{x}_n, \boldsymbol{\omega})$$

Perceptron

$$\left\{\mathbf{x}_n, d_n\right\}$$

Training data

Train the perceptron using SGD

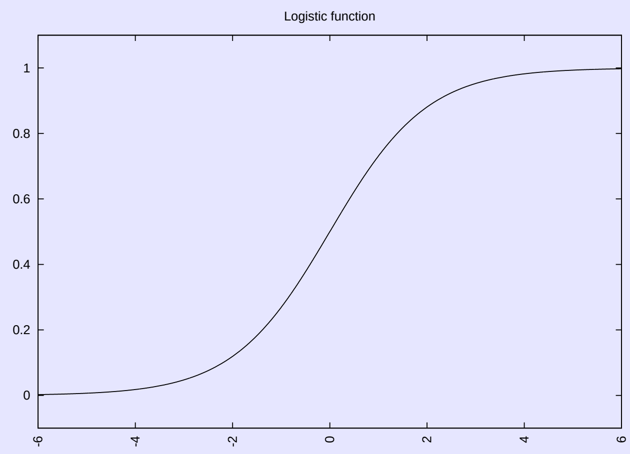
$$\omega_i \to \omega_i - \eta \frac{1}{P} \sum_{p=1}^{P} \frac{\partial E_p(\boldsymbol{\omega})}{\partial \omega_i}$$

Repeat until convergence!

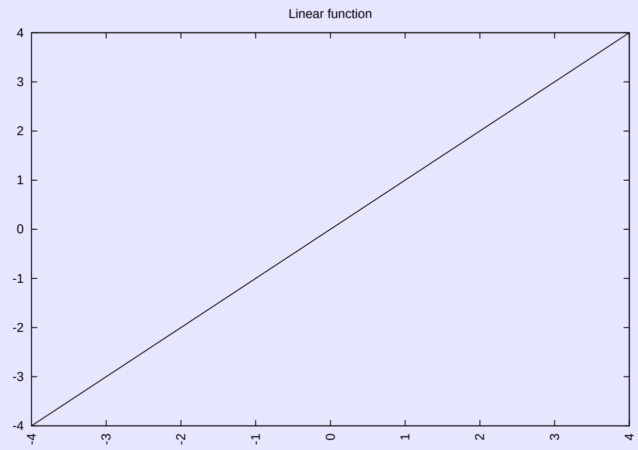
Summary of the “perceptron”

What about activation functions for the perceptron?

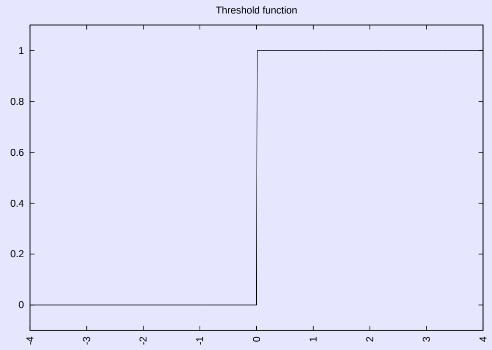
Classification problems



Regression problems

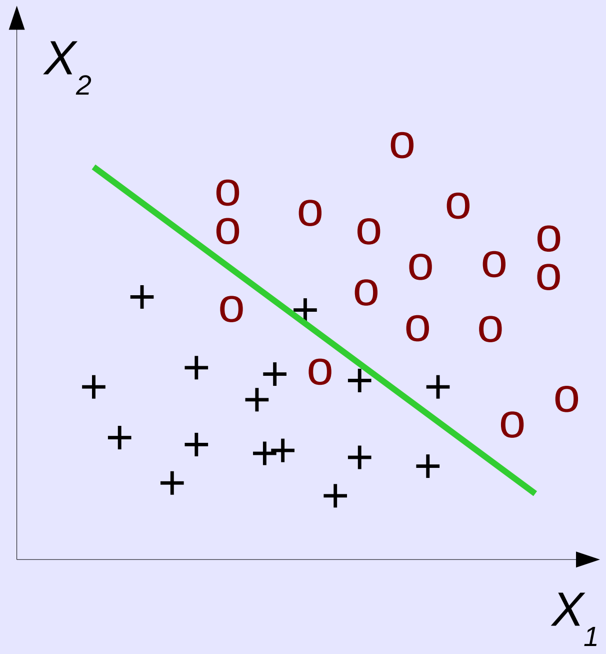


Not so common!

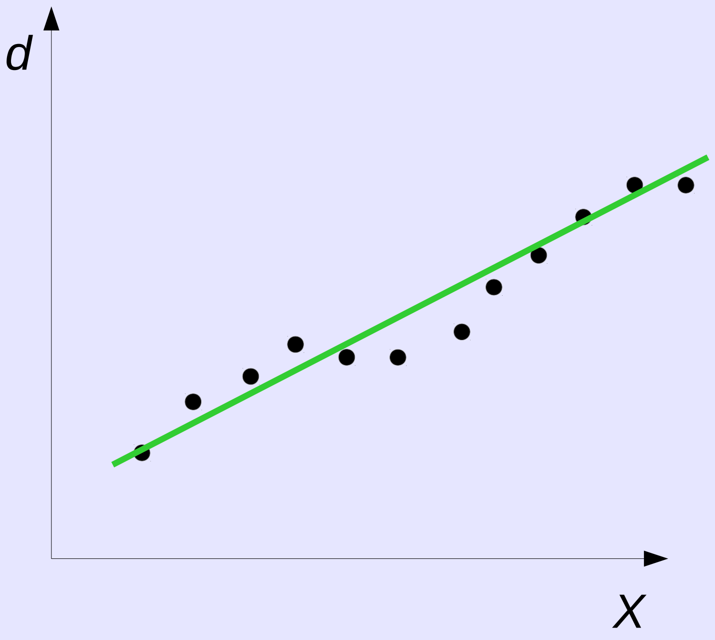


Fundamental limitation of the perceptron!

Linear boundary!



Linear regression!



To “get around” this limitation, we introduce a hidden layer → Multi-Layer Perceptron (MLP)