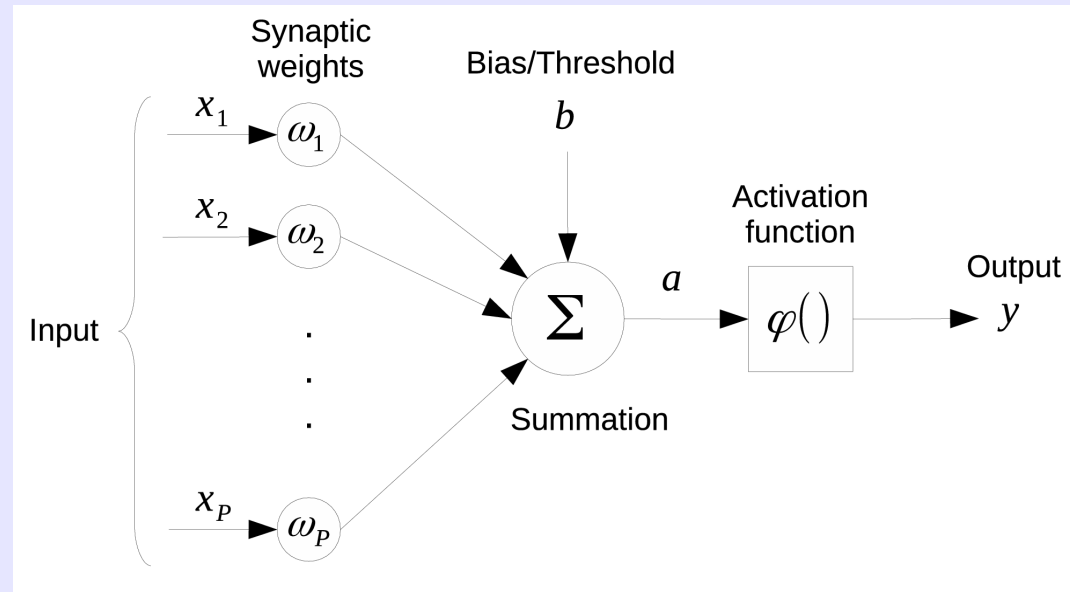


The perceptron - details

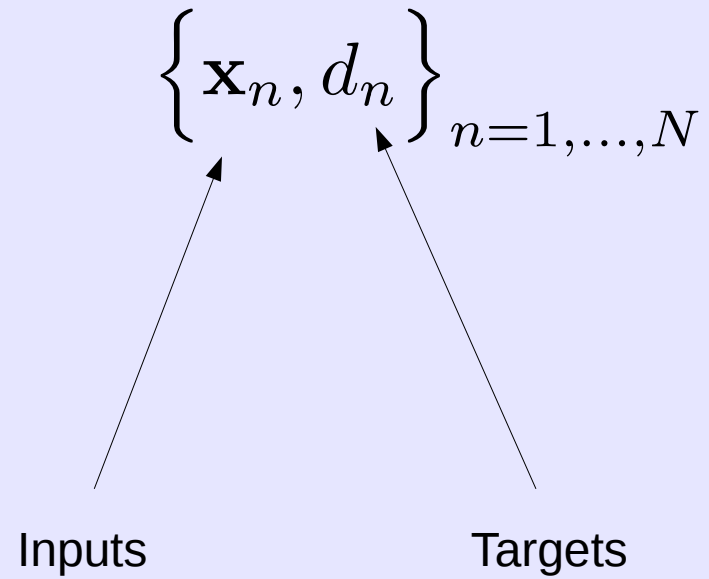


It is just a function

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

Summary of the “perceptron”

We have data, typically called a training dataset!



Summary of the “perceptron”

An example (MNIST database)!



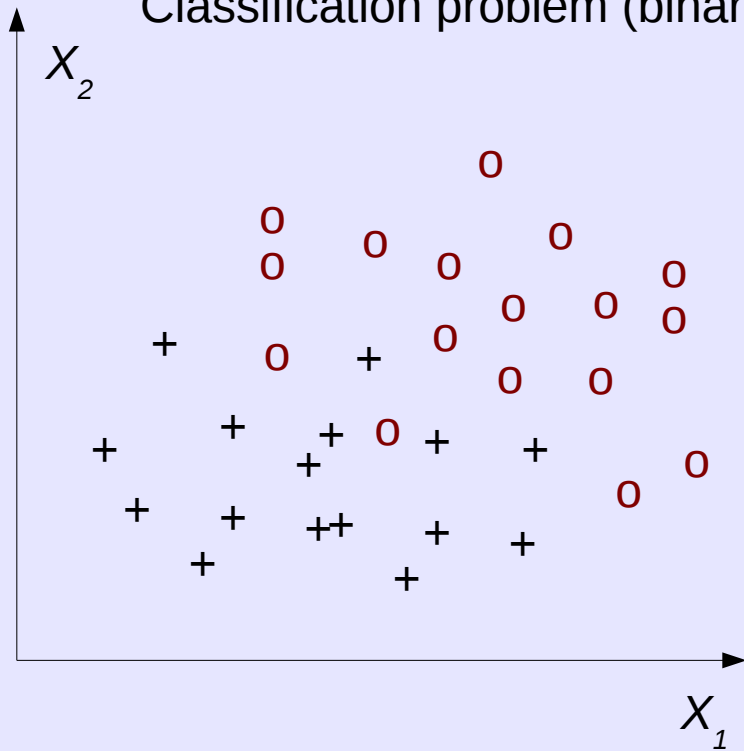
Inputs = 28 x 28 grayscale images

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

Summary of the “perceptron”

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

Summary of the “perceptron”

So, we have

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

Perceptron Training data

The diagram shows the equation $y(\mathbf{x}_n) = \varphi(\mathbf{x}_n, \boldsymbol{\omega})$ on the left and the set notation $\left\{ \mathbf{x}_n, d_n \right\}$ on the right. Below the equation, the word "Perceptron" is written, with an arrow pointing from it to the φ symbol in the equation. Below the set notation, the words "Training data" are written, with an arrow pointing from them to the $\left\{ \mathbf{x}_n, d_n \right\}$ expression.

Task!

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

How?

Summary of the “perceptron”

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

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

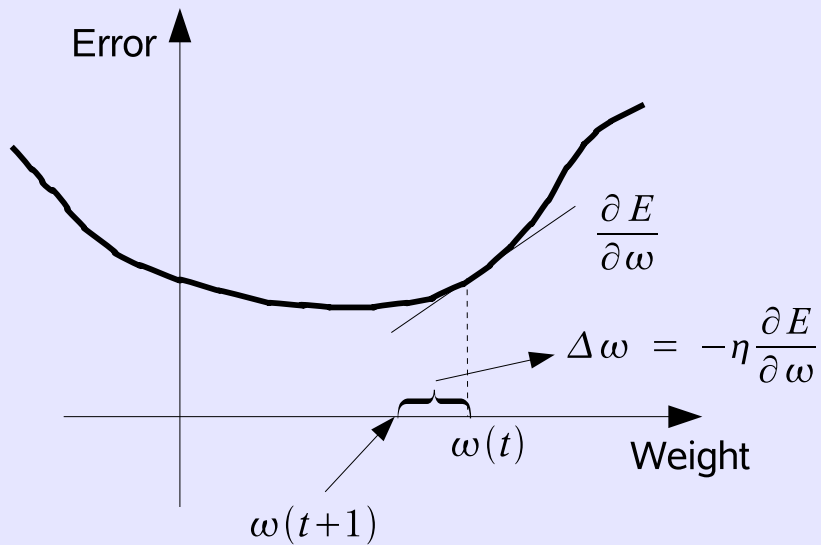
A function of the weights!



Minimizing $E(w)$ means “solving” the task
(or at least an attempt to solve it)

How?

Minimize using gradient descent



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

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

Stochastic gradient descent (SGD)

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

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

Finally!

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

Perceptron Training data

The diagram shows the equation $y(\mathbf{x}_n) = \varphi(\mathbf{x}_n, \boldsymbol{\omega})$ on the left and the set notation $\left\{ \mathbf{x}_n, d_n \right\}$ on the right. Below the equation, the word "Perceptron" is written, with an arrow pointing from it to the φ function. Below the set notation, the words "Training data" are written, with an arrow pointing from them to the $\left\{ \mathbf{x}_n, d_n \right\}$ set.

Train the perceptron using SGD

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

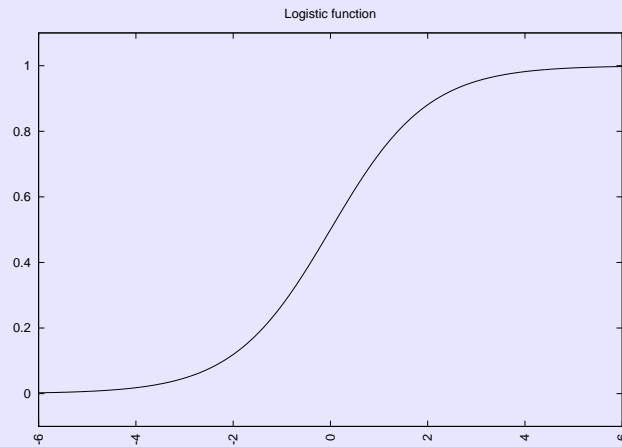
Repeat until convergence!

The diagram shows the SGD update equation $\omega_i \rightarrow \omega_i - \eta \frac{1}{P} \sum_{p=1}^P \frac{\partial E_p(\boldsymbol{\omega})}{\partial \omega_i}$. Below the equation, the text "Repeat until convergence!" is written, with an arrow pointing from it up to the equation.

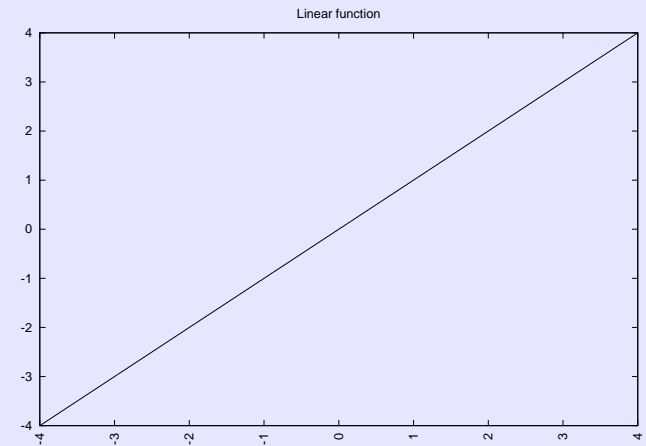
Summary of the “perceptron”

What about activation functions for the perceptron?

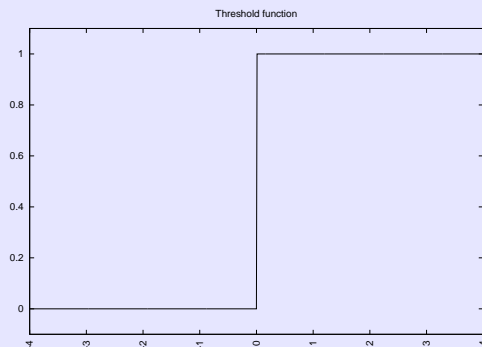
Classification problems



Regression problems

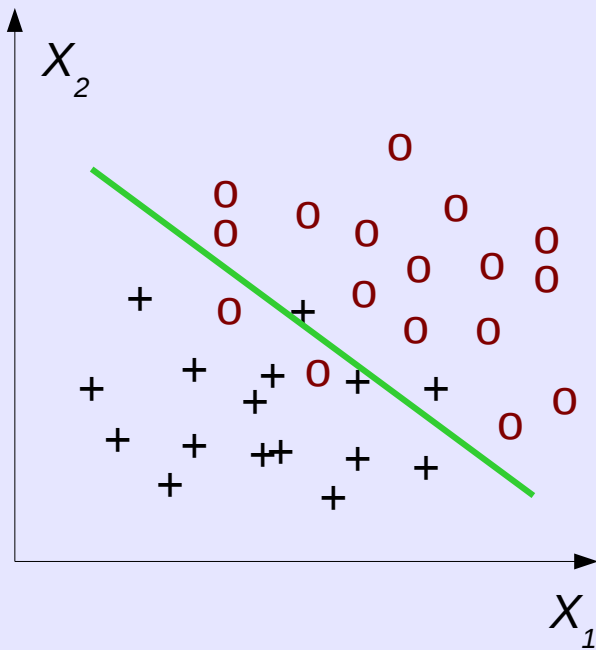


Not so common!

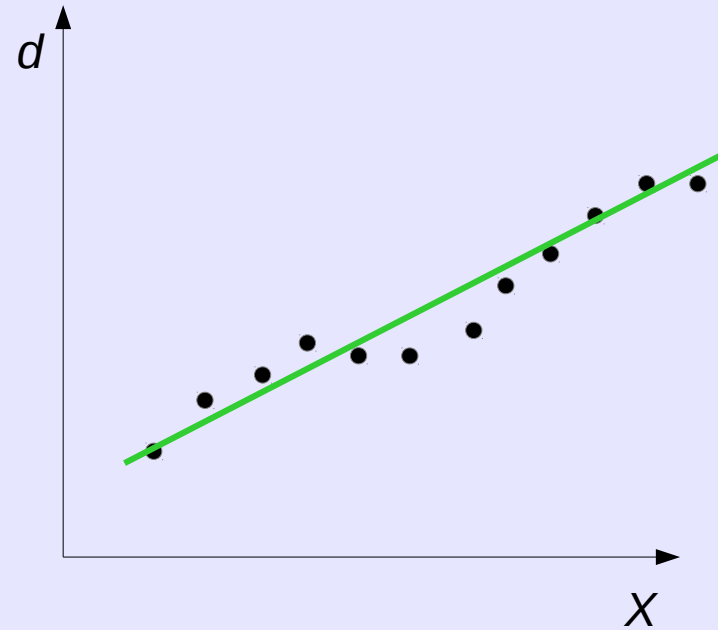


Fundamental limitation of the perceptron!

Linear boundary!



Linear regression!



Summary of the “perceptron”

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

