Lecture 11. Introduction to Artificial Neural Networks

COMP90051 Statistical Machine Learning

Semester 2, 2015
Lecturer: Andrey Kan

Content is largely based on slides provided by Jeffrey Chan and Ben Rubinstein

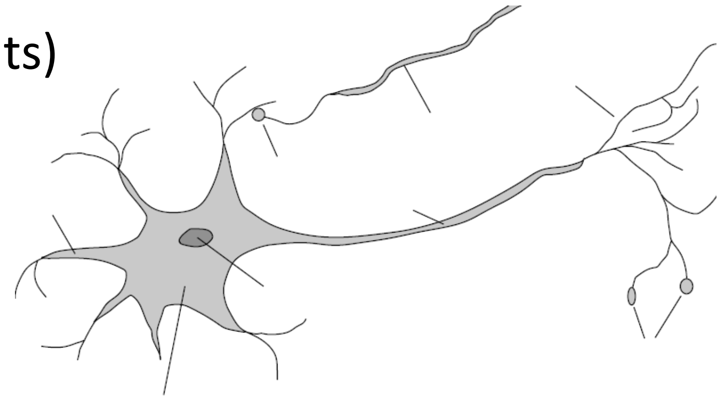
THE UNIVERSITY OF MELBOURNE

Introduction to Artificial Neural Networks

A biologically-inspired non-linear model that can be competitive with state-of-the-art, with learning algorithms based on gradient descent.

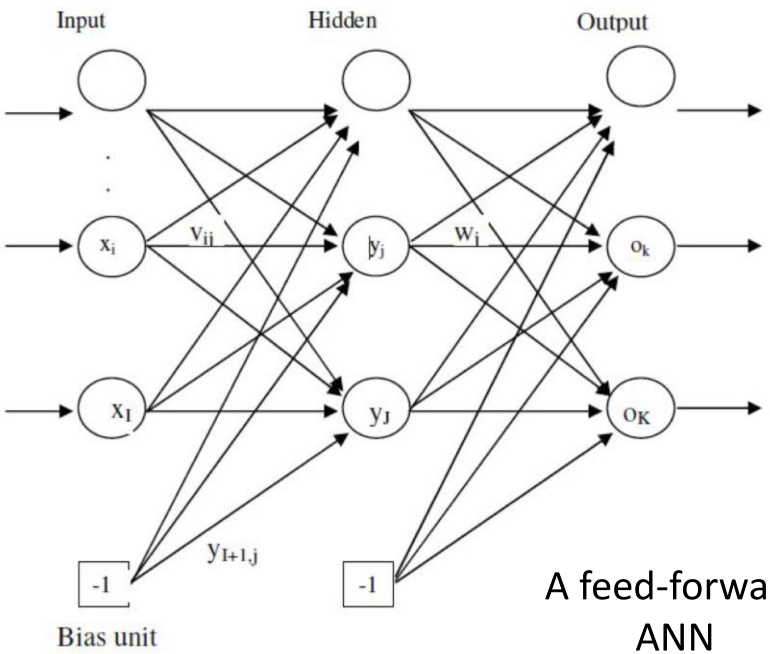
The Human Brain

- 10^{11} neurons of over 20 types, 10^{14} synapses
  - Signals are noisy "spike trains" of electrical potential
- Neurons (nerve cells) have:
  - *Dendrites* (inputs) and an *axon* (outputs)
- Synapses (connections b/w cells)
  - Can be excitatory or inhibitory
  - May change over time (plasticity)
- When the inputs reach some threshold an *action potential* (electrical pulse) is sent along the axon to the outputs



Artificial Neural Networks

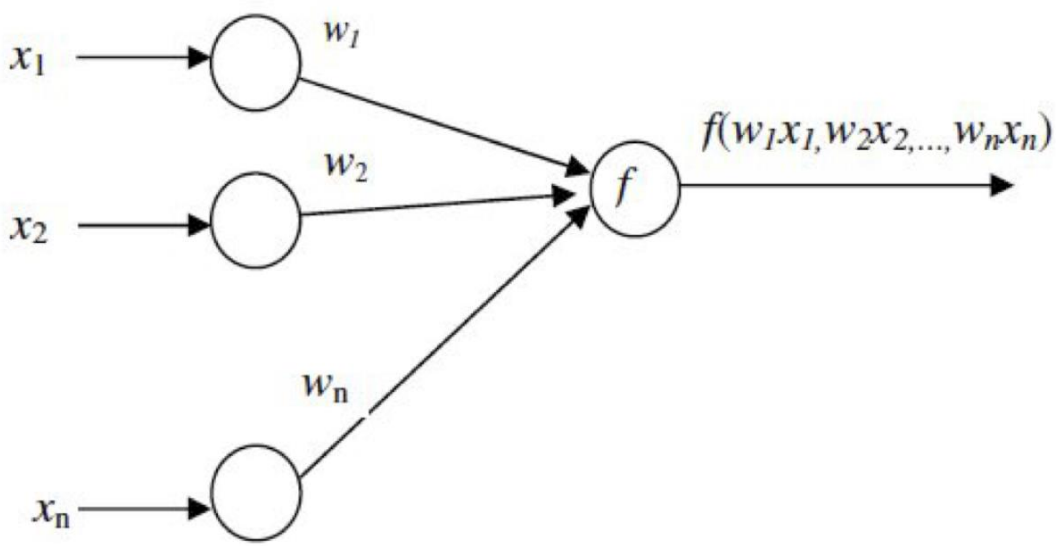
- ANNs made up of nodes having
  - inputs edges, each with some *weight*
  - outputs edges (with *weights*)
  - a *transfer function* (aka *activation function*) which is a function of inputs



A feed-forward ANN

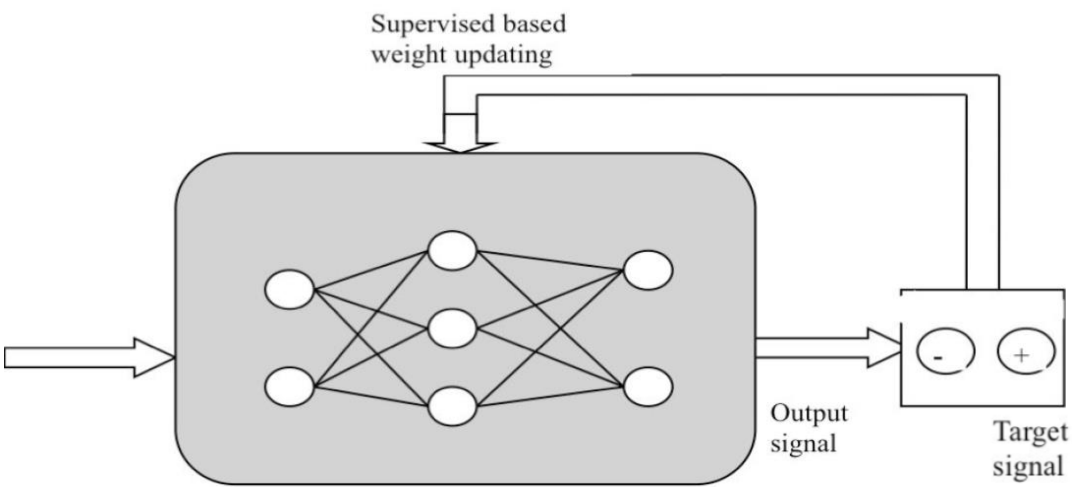
- Weights of edges can be positive or negative and may change over time (learning).

Artificial Neural Networks: Activation



- The *input function* is the weighted sum of input activation levels: $in_i = \sum_{j=1}^{n} w_j x_j$
- The transfer function is the function of input: $a_i = f(in_i) = f\left(\sum_{j=1}^{n} w_j x_j\right)$

Artificial Neural Networks: Architecture?

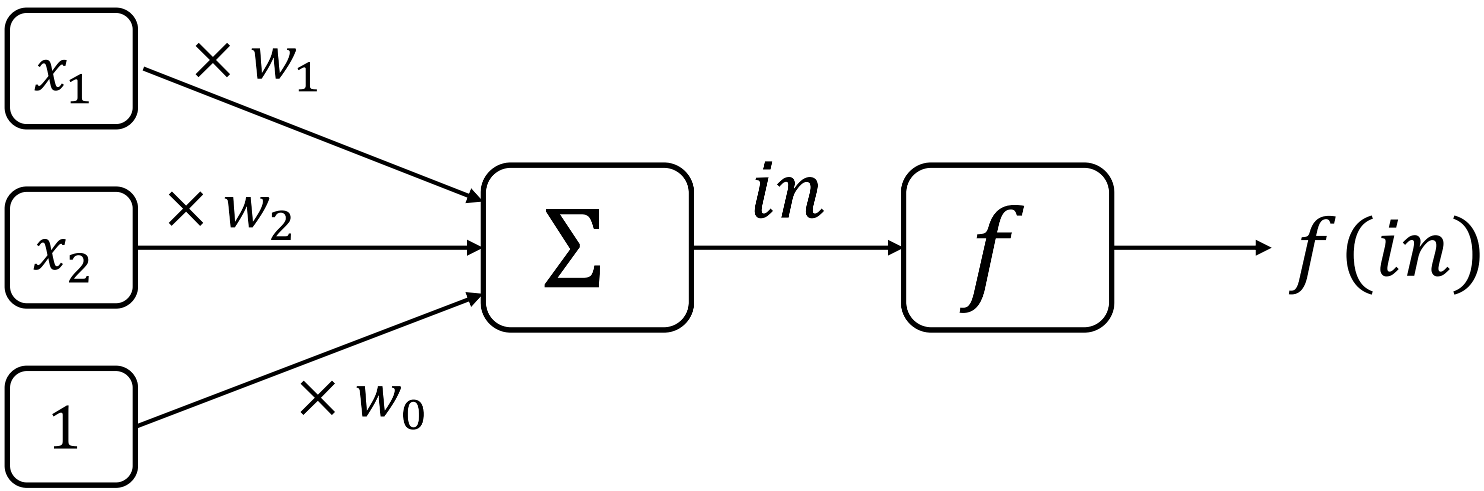


- How can the weights be changed to produce the desired output?
- What is the best topology?

The Perceptron

ANN building block; yet another linear learner.

Perceptron Model



Compare to linear regression and linear logistic regression

- x_1, x_2 – inputs
- w_1, w_2 – synaptic weights
- w_0 – bias weight
- f – transfer function

Transfer functions f

Step function
$$f(s) = \begin{cases} 1, & if\ s \geq 0 \\ 0, & if\ s < 0 \end{cases}$$

Sign function
$$f(s) = \begin{cases} 1, & if\ s \geq 0 \\ -1, & if\ s < 0 \end{cases}$$

Logistic function
$$f(s) = \frac{1}{1 + e^{-s}}$$

Many others: $tanh$, rectifier, etc.

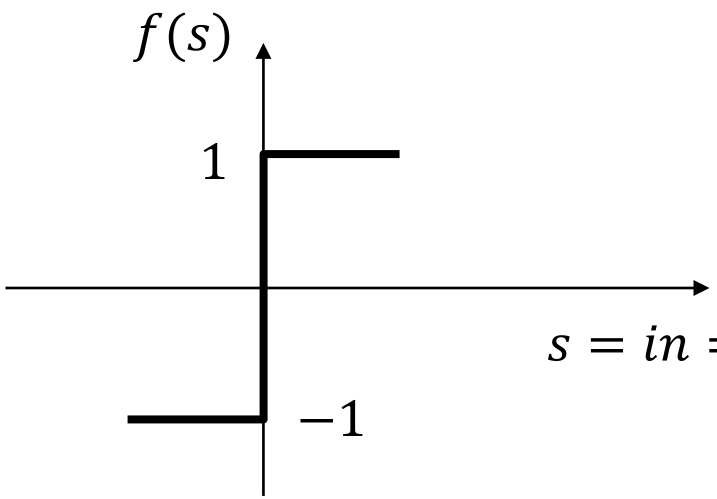
Still, for classification just a linear separator (s is a linear function of input)

Binary Classification

Consider a binary classification task with labels -1 and 1

Classifier: perceptron with sign function

$$f(s) = \begin{cases} 1, & if\ s \geq 0 \\ -1, & if\ s < 0 \end{cases}$$



$$s = in = \sum_{i=0}^{n} w_i x_i$$

Stochastic Gradient Descent

1. Initialisation: choose starting guess $\boldsymbol{w}^{(0)}, k = 0$
2. Randomly choose one training example $(\boldsymbol{x}, y)$
3. Compute discrepancy $D = \left(y - f\left(\sum_{i=0}^{n} w_i^{(k)} x_i\right)\right)^2$
4. Termination: decide whether to stop
5. Update: $w_i^{(k+1)} = w_i^{(k)} - \eta \frac{\partial D}{\partial w_i}$
6. Go to Step 2

Problems?

Stochastic Gradient Descent

1. Initialisation: choose starting guess $\boldsymbol{w}^{(0)}$, $k = 0$
2. Randomly choose one training example $(\boldsymbol{x}, y)$
3. Compute discrepancy $D = -y \sum_{i=0}^{n} w_i^{(k)} x_i$
4. Termination: decide whether to stop
5. Update: $w_i^{(k+1)} = w_i^{(k)} - \eta \frac{\partial D}{\partial w_k}$
6. Go to Step 2

Taking derivatives is convenient!

Perceptron (Online) Learning Rule

discrepancy $D = -y\sum_{i=0}^{n} w_i^{(k)} x_i$

If $f(s) = -1$, but $y = 1$:

$w_i \leftarrow w_i + \eta x_i$

$w_0 \leftarrow w_0 + \eta$

so that:

$$s \leftarrow s + \eta\left(1 + \sum_i x_i^2\right)$$

If $f(s) = 1$, but $y = -1$:

$w_i \leftarrow w_i - \eta x_i$

$w_0 \leftarrow w_0 - \eta$

so that:

$$s \leftarrow s - \eta\left(1 + \sum_i x_i^2\right)$$

Otherwise, weights are unchanged

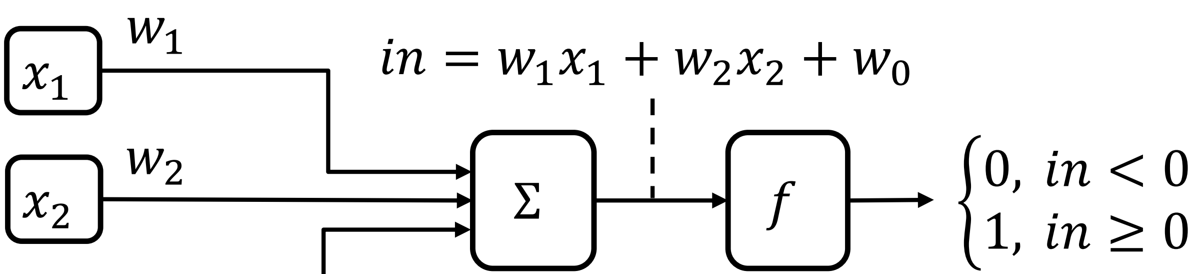
$\eta > 0$ is called *learning rate*

Perceptron (Online) Learning Rule

- This rule is equivalent to
  - Minimising loss over training data (like in linear regression)
  - Using gradient descent to do the minimisation (different from linear regression)
  - Stochastic gradient descent: applying gradient descent for training examples one by one (→ a method for online learning)
- Theorem: A perceptron will learn to classify the data correctly if:
  - The data is linearly separable
  - η is suitably small

Perceptron Learning Example

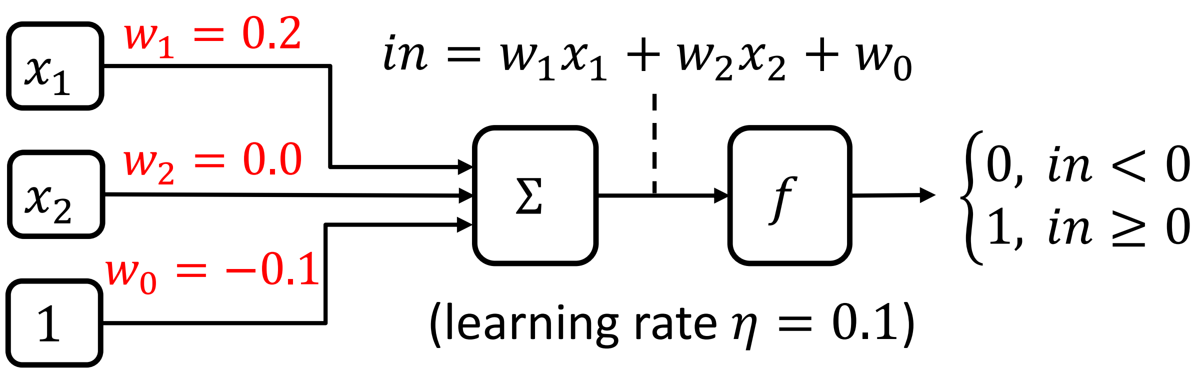
Basic setup

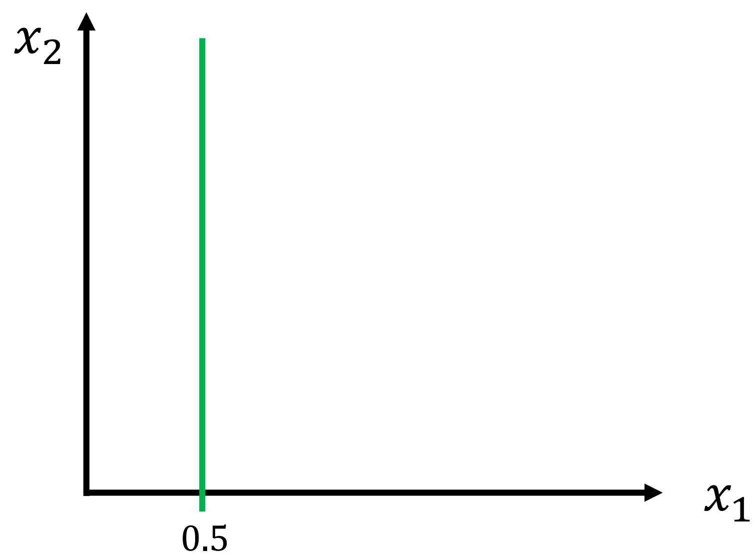


(learning rate $\eta = 0.1$)

Perceptron Learning Example

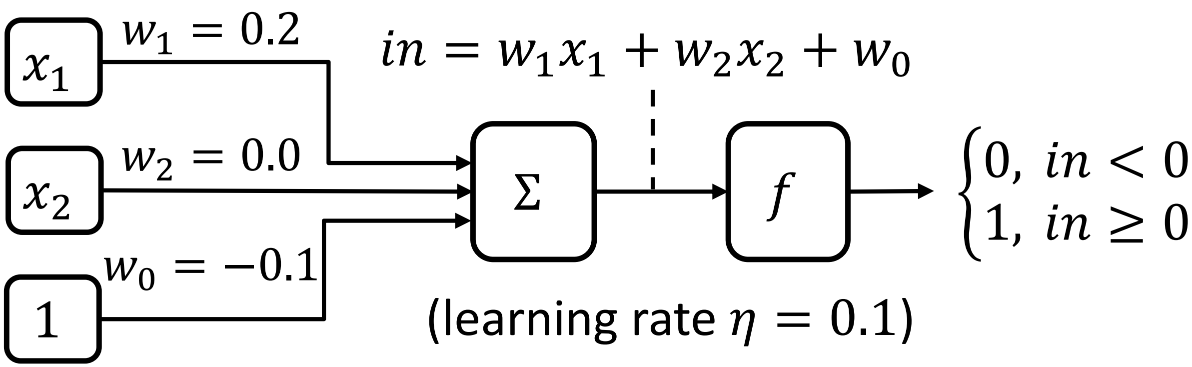
Start with random weights





Perceptron Learning Example

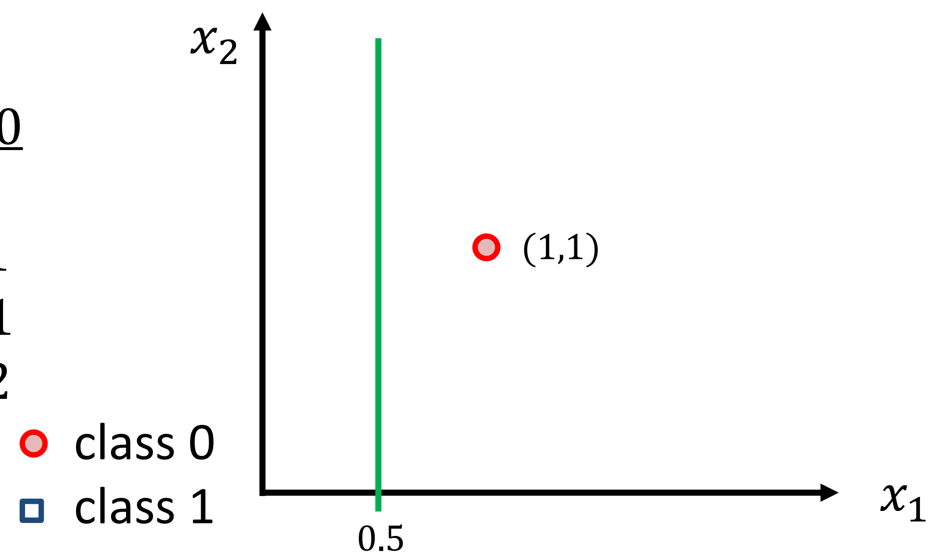
Consider training example 1



(learning rate $\eta = 0.1$)

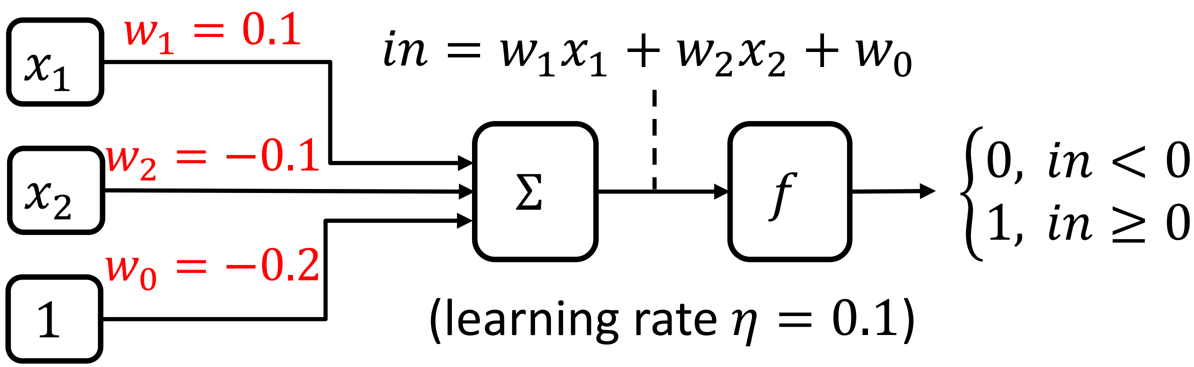
$0.2x_1 + 0.0x_2 - 0.1 > 0$

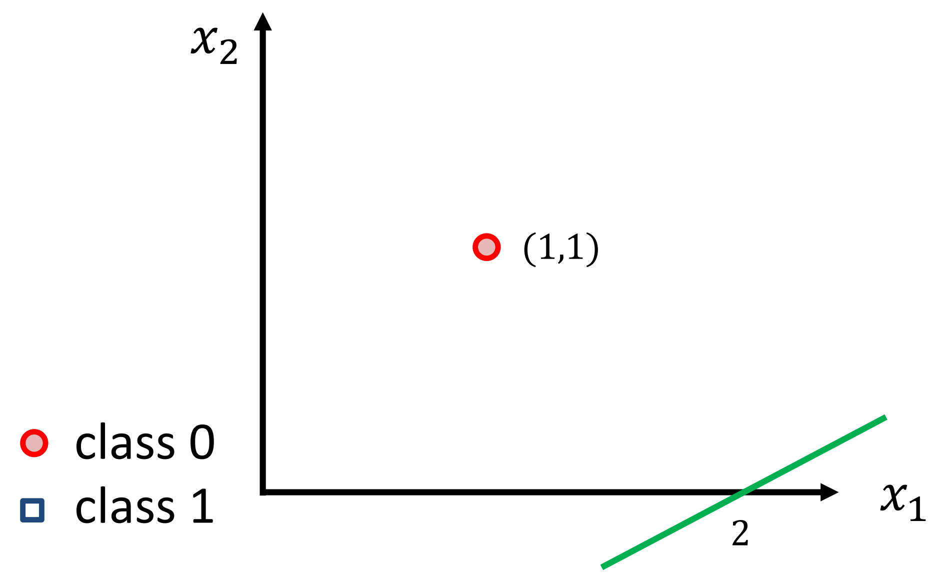
$$w_1 \leftarrow w_1 - \eta x_1 = 0.1$$
$$w_2 \leftarrow w_2 - \eta x_2 = -0.1$$
$$w_0 \leftarrow w_0 - \eta = -0.2$$



Perceptron Learning Example

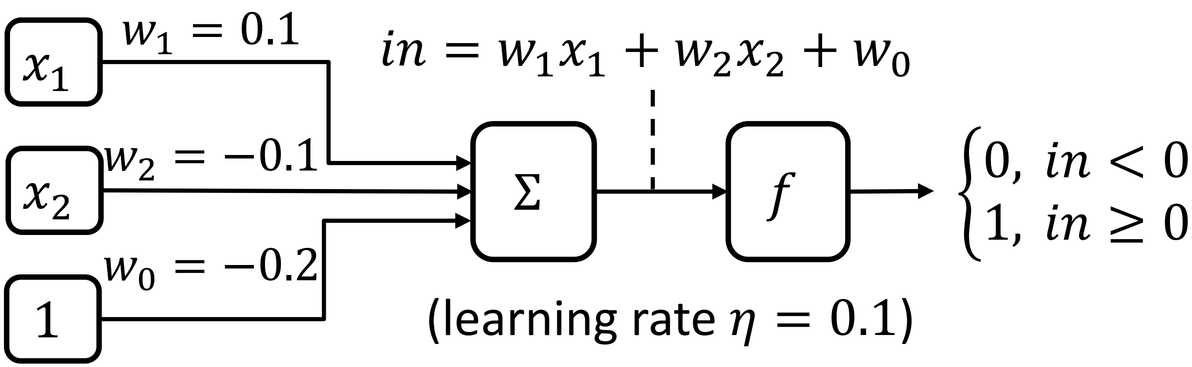
Update weights





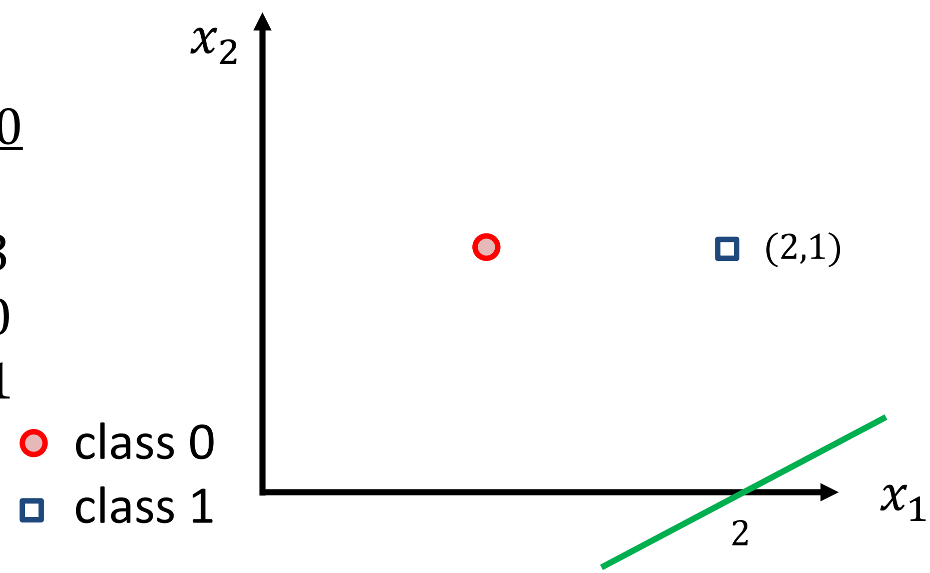
Perceptron Learning Example

Consider training example 2



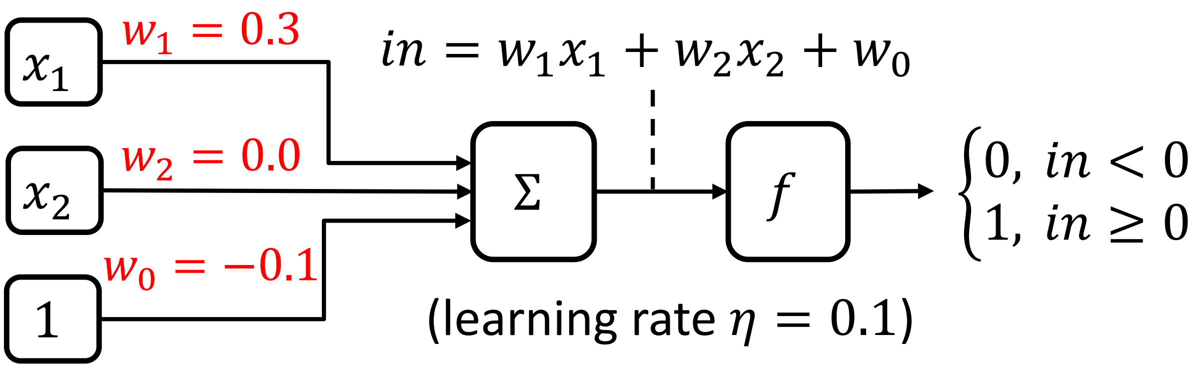
$$\underline{0.1x_1 - 0.1x_2 - 0.2 < 0}$$

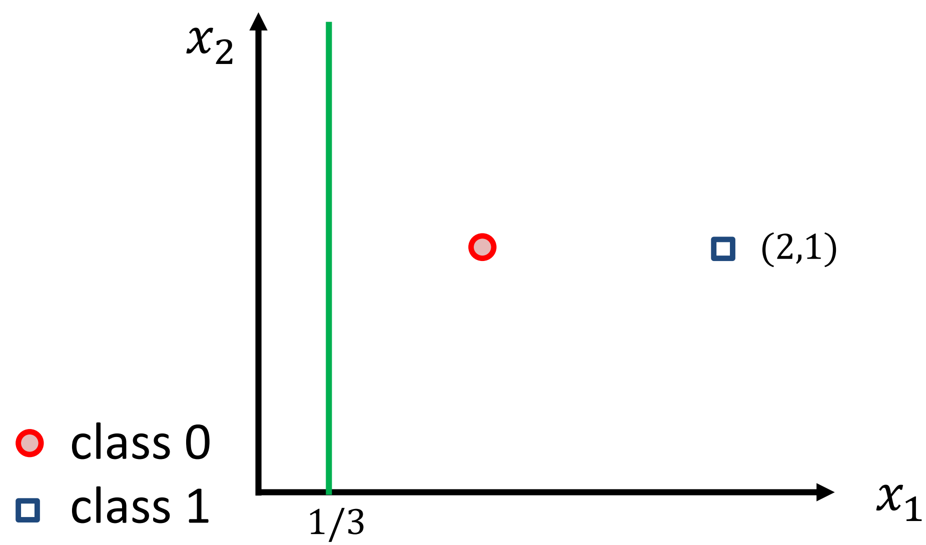
$$\begin{aligned} w_1 &\leftarrow w_1 + \eta x_1 = 0.3 \\ w_2 &\leftarrow w_2 + \eta x_2 = 0.0 \\ w_0 &\leftarrow w_0 + \eta = -0.1 \end{aligned}$$



Perceptron Learning Example

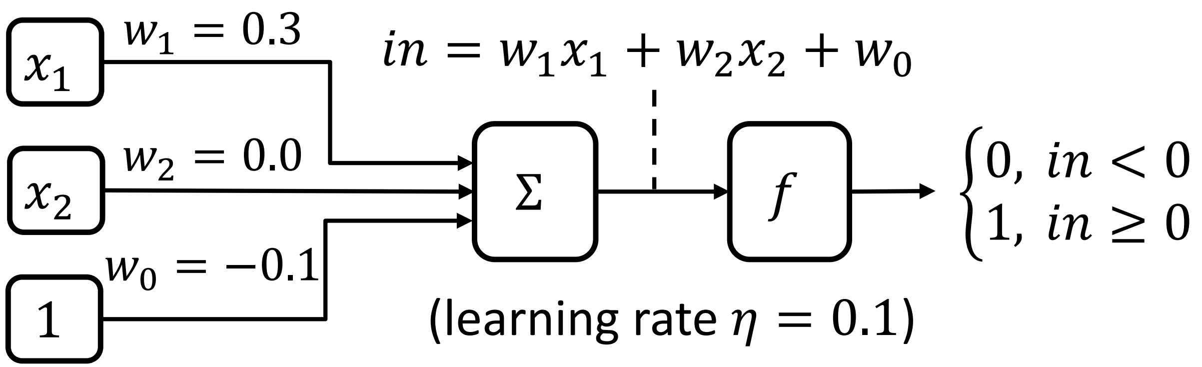
Update weights





Perceptron Learning Example

Further examples



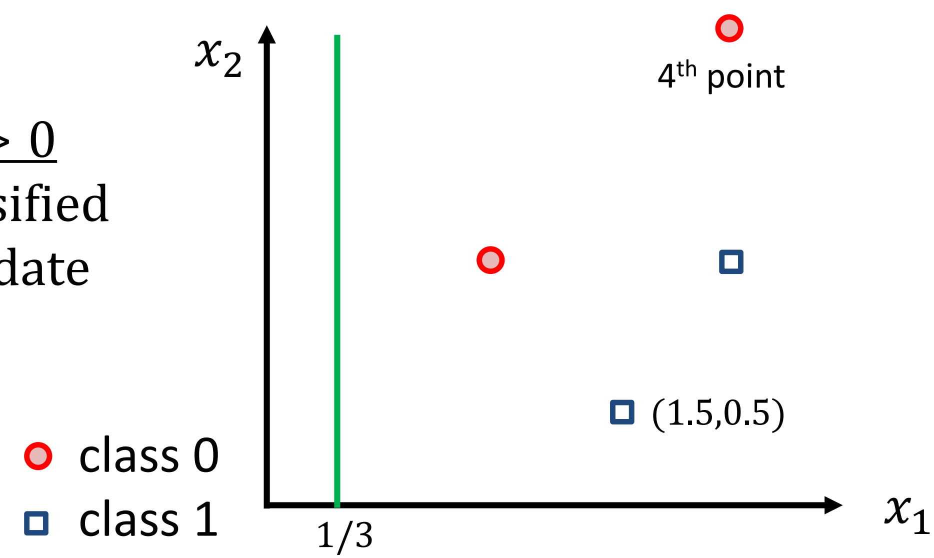
(learning rate $\eta = 0.1$)

$\underline{0.3x_1 - 0.0x_2 - 0.1 > 0}$

3rd point: correctly classified

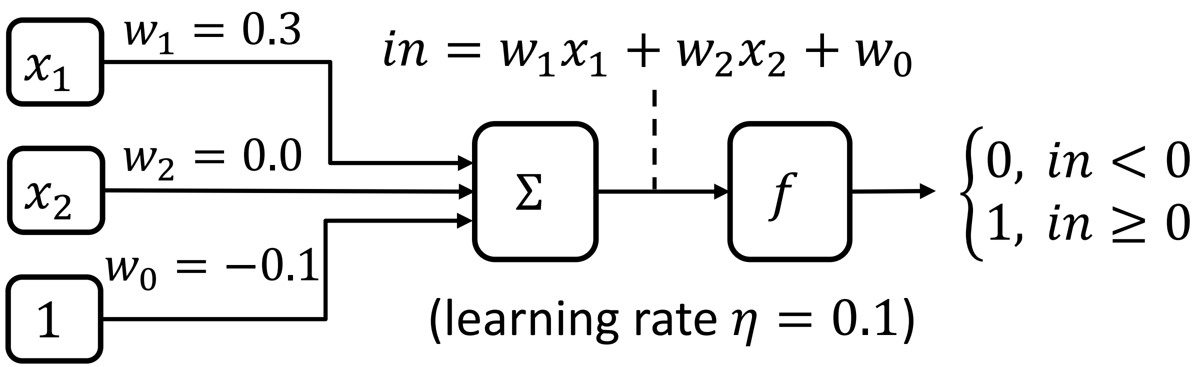
4th point: incorrect, update

etc.

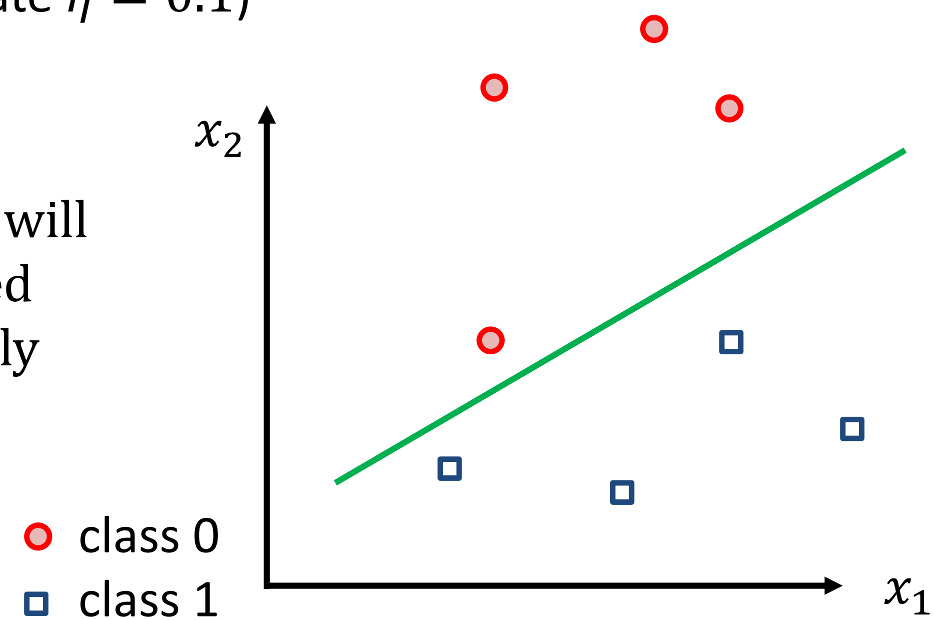


Perceptron Learning Example

Further examples



Eventually, all the data will be correctly classified (provided it is linearly separable)

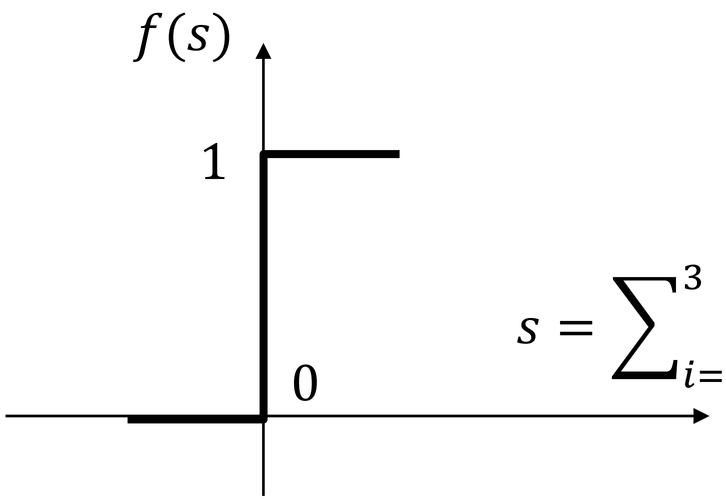


Modelling Boolean Functions

Consider a Boolean function of two variables, e.g., $y = x_1 \text{ AND } x_2$

Classifier: perceptron with step function

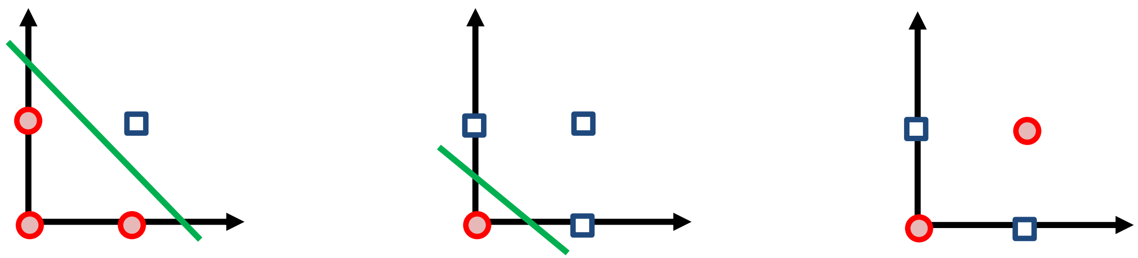
$$f(s) = \begin{cases} 1, & if\ s \geq 0 \\ 0, & if\ s < 0 \end{cases}$$



$$s = \sum_{i=0}^{3} w_i x_i$$

More Examples and Limitations

Some function are linearly separable, but many are not



AND OR XOR

Possible solution: composition

$$x_1 \text{ XOR } x_2 = (x_1 \text{ OR } x_2) \text{ AND } \overline{(x_1 \text{ AND } x_2)}$$

Summary

- Neural networks are biologically inspired
- Perceptron as a building block of ANN
  - Graphical representation of an equation
  - Linear model (plus transformation)
- Online learning rule
  - Perceptron can learn any linearly separable function