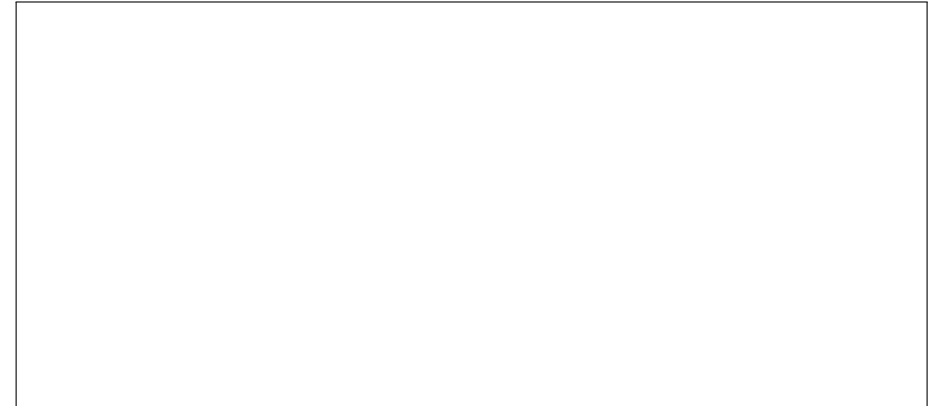


Types of learning

1. Learning with a teacher: supervised learning.
 2. Learning with a critic: reinforcement learning.
 3. Learning on your own: unsupervised learning.
- How we study learning in neural networks:

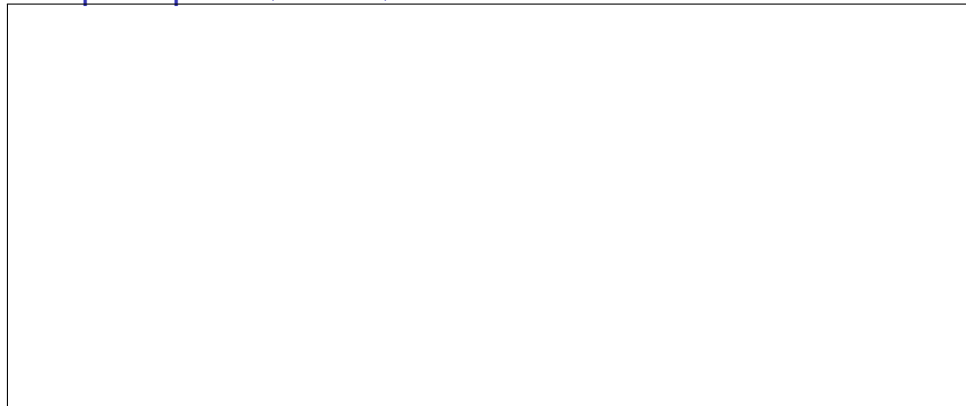


The perceptron

1 / 16

2 / 16

The perceptron (Rosenblatt 1957)



Notation:

- x_i : activity of input unit i (binary or $[0,1]$).
- w_i synaptic weight from unit i .
- $z = \sum_i w_i x_i$ total input to the output unit.
- y : activity of output unit.
- t : desired output (of use later). (μ superscript denotes training sample.)
- $f(\cdot)$: transfer function; $y = f(z)$.

3 / 16

Training set

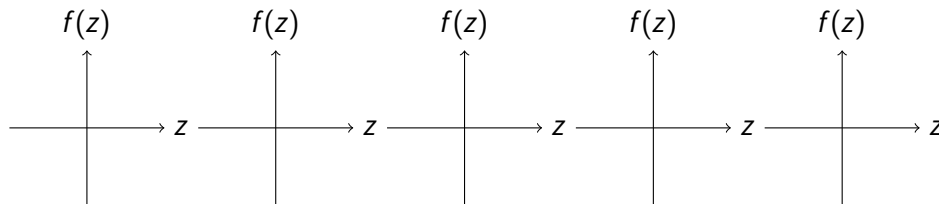
Sample	x_1	x_2	t
$\mu = 1$	0	0	0
$\mu = 2$	0	1	0
$\mu = 3$	1	0	0
$\mu = 4$	1	1	1

4 / 16

Transfer function

Given total weighted input to neuron, what is its output?

1. identity: $f(z) = z$.
2. threshold: $f(z, \theta) = \begin{cases} 1 & \text{if } z \geq \theta \\ 0 & \text{otherwise} \end{cases}$
3. sigmoidal: $f(z) = \frac{1}{1 + \exp(-kz)}$
4. tanh: $f(z) = \tanh(z)$
5. rectified linear unit (ReLU): $f(z) = \max(0, z)$



How to choose? One key property: differentiable.

5 / 16

Perceptron decisions

Whether a perceptron's output is 0 or 1 depends on whether $\sum_{i=1}^N w_i x_i$ is less or greater than θ .

The equation

$$\sum_{i=1}^N w_i x_i = \theta$$

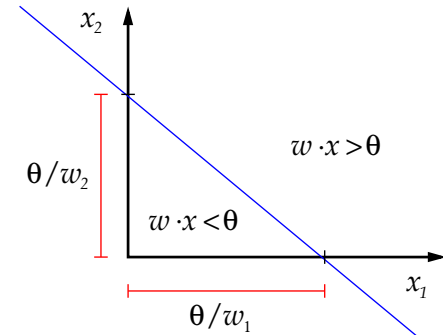
defines a hyperplane in N -dimensional space.

This hyperplane cuts the space in two.

$$w_1 x_1 + w_2 x_2 = \theta$$

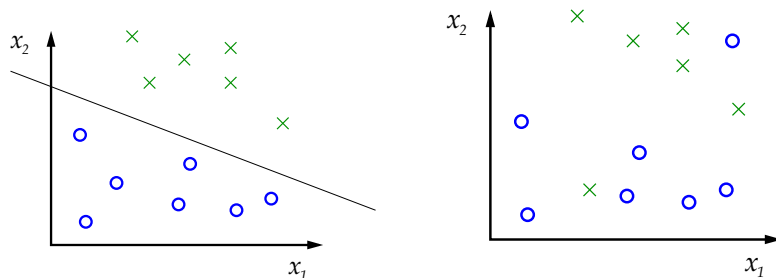
$$x_2 = \left(\frac{-w_1}{w_2} \right) x_1 + \frac{\theta}{w_2}$$

This is an equation for a straight line.



6 / 16

Linearly separable problems and the perceptron

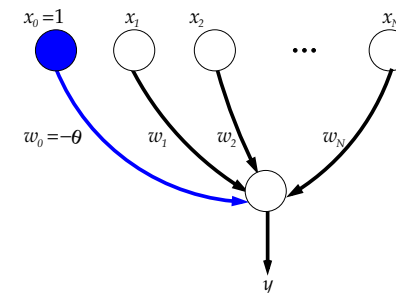


Learning involves adjusting the values of w and θ so that the decision plane can correctly divide the two classes.

7 / 16

Threshold & Bias

The threshold, θ , can be treated as just another weight from a new input unit which always has a value of +1 (or -1). This new input unit is called the **bias** unit.



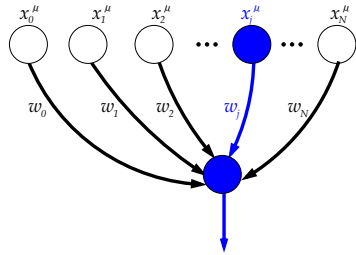
Instead of comparing $\sum_{i=1}^N w_i x_i$ with θ , we compare $\sum_{i=0}^N w_i x_i$ with 0, or

$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^N w_i x_i > 0, \\ 0 & \text{if } \sum_{i=0}^N w_i x_i \leq 0. \end{cases}$$

Now, learning is about jiggling **weights** only.

8 / 16

Intuitively... (positive valued inputs)



$$y^\mu = \text{step} \left(\sum_{i=0}^N w_i x_i^\mu \right)$$

Consider w_j 's contribution to $\sum_i w_i x_i^\mu$ in different cases:

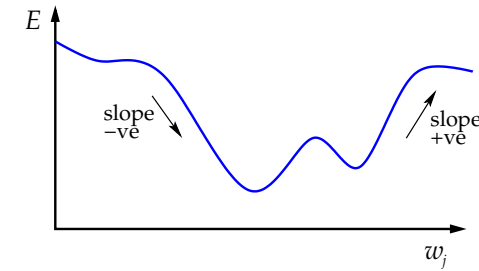
1. $y^\mu = t^\mu$ Perceptron has classified input μ correctly – **change nothing**
2. $x_j^\mu = 0$ Changing w_j will not affect the $\sum_i w_i x_i^\mu$ – **change nothing**
3. $x_j^\mu \neq 0, y^\mu < t^\mu$ The sum $\sum_i w_i x_i^\mu$ is too low – **so increase it**
4. $x_j^\mu \neq 0, y^\mu > t^\mu$ The sum $\sum_i w_i x_i^\mu$ is too high – **so decrease it**

The local rule:

$$\Delta w_j \propto (t^\mu - y^\mu) x_j^\mu$$

Example of a perceptron learning

Perceptron learning rule



$$y = f(\mathbf{w} \cdot \mathbf{x}) \quad \text{e.g.} \quad f(z) = 1/(1 + \exp(-z)), \quad f(z) = z$$

$$E = \frac{1}{2}(t - y)^2 \quad t \text{ is target output}$$

$$\Delta w_j = -\epsilon \frac{\partial E}{\partial w_j} = -\epsilon \frac{\partial E}{\partial y} \frac{\partial y}{\partial w_j} =$$

This is the method of **gradient descent** with learning-rate parameter ϵ .

Perceptron pros and cons

- The *perceptron convergence theorem* (Dayan and Abbott, page 327) guarantees that solution will be found **iff there is a linearly separable solution**.
- Many complex problems are not linearly separable, e.g. XOR problem.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0