# Neural Network learning

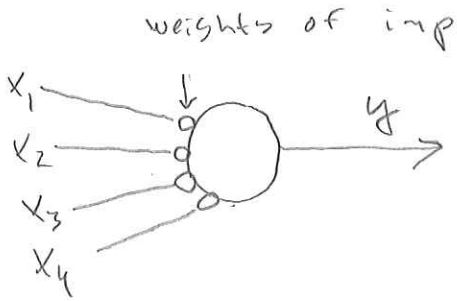
Biology : <SLIDE>

Dendrites => Inputs
Axon => Output
Neuron => model?
learning?

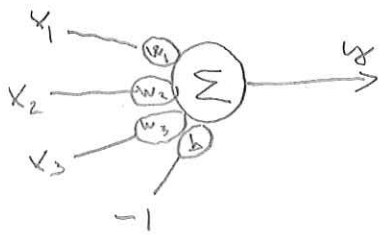
Synapses connect axons to dendrites of other neurons

Let's use the most investigated model: a linear model

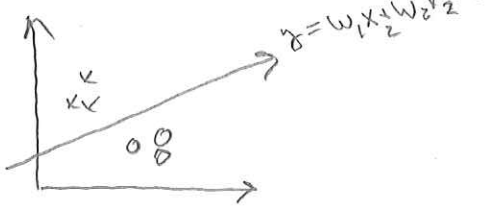
Hypothesis: a neuron produces a strong output when it is activated (e.g. "I see a bicycle!" → 1, "I don't see a bicycle" → 0). The neuron is activated by input of a "pattern" (bicycle features) to its dendrites. Synapses exhibit or inhibit specific inputs.



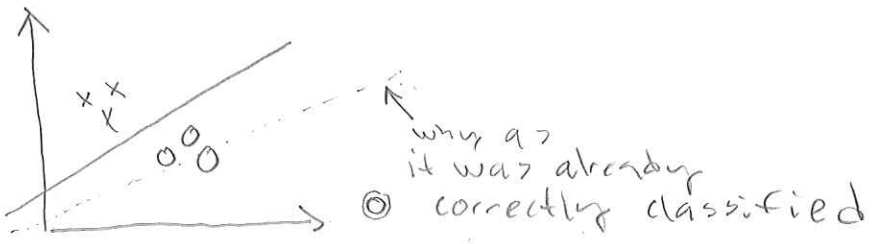

Recall the linear model: $y = w_1x_1 + w_2x_2 + \dots - b$



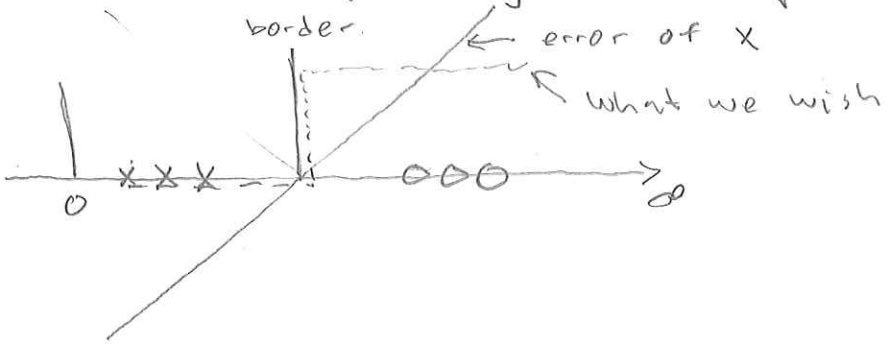

This is called a "perceptron" and it can classify inputs to two classes



Problem of the linear model for classification




⇒ error must not depend on how far on the correct/wrong side you are




Good approximation of the step fuction is logsig(x)

$$logsig(x) = \frac{1}{1+e^{-x}}$$

$logsig \to 1$ when $x \to \infty$

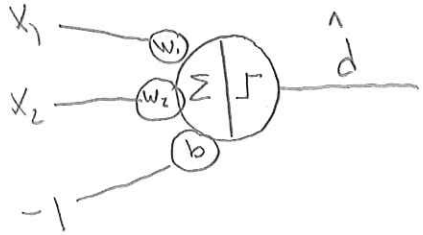
$logsig \to 0$ when $x \to -\infty$

Another nice property: $\frac{\partial logsig(x)}{\partial x} = logsig(x)(1-logsig(x))$ ← prove!

we have a model!, $\boxed{\hat{d} = logsig\left(\sum_{i=1}^{D} w_i x_i - b\right)}$

How about training method if we have N training examples

$$\begin{pmatrix} \bar{x}^{(0)}, d^{(0)} \\ \bar{x}^{(1)}, d^{(1)} \\ \vdots \\ \bar{x}^{(N)}, d^{(N)} \end{pmatrix}$$



⇒ Measure error and minimise it!

$$E(\bar{w}_i, \bar{X}^{(N)}) = MSE = \frac{1}{N}\sum_{k=0}^{N-1}\left(d^{(k)} - \hat{d}^{(k)}\right)^2$$

$\nabla E = 0$, no analytical solution, but start from a random point and move toward negative gradient by step $\mu$ (toward minimum)

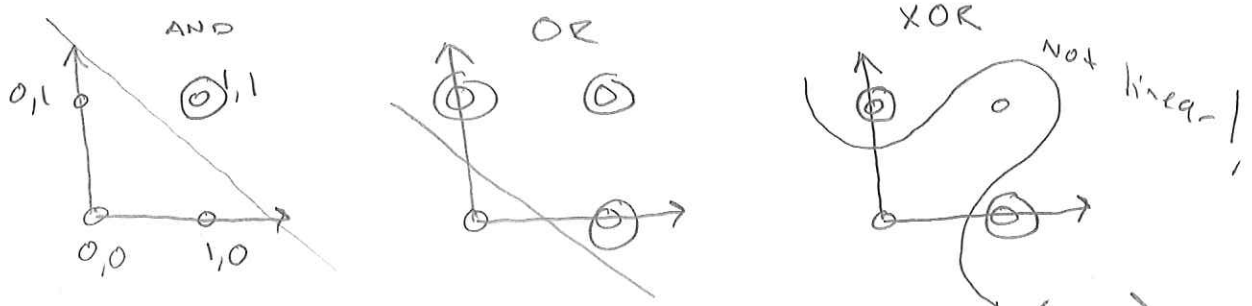
$$w_i^{t+1} = w_i^t - \mu \frac{\partial E}{\partial w_i}$$

$$b^{t+1} = b^t - \mu \frac{\partial E}{\partial b}$$

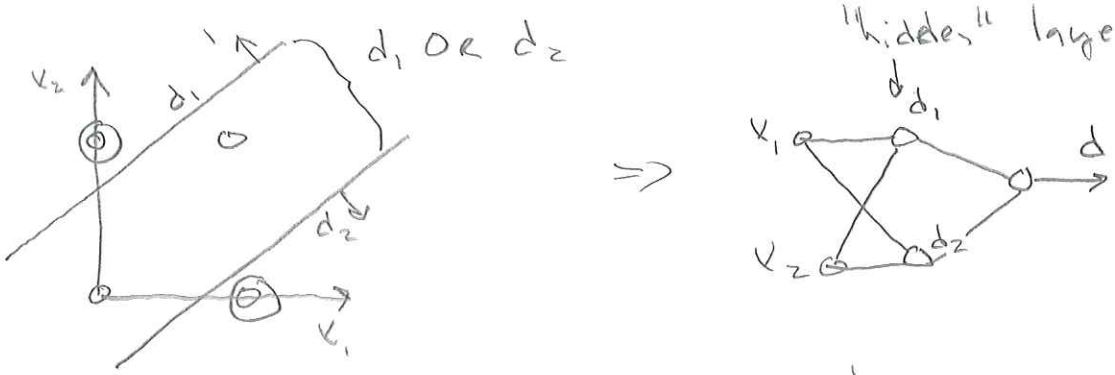
⇒ Gradient descent algorithm

A perceptron is a linear classifier

Example




1. solution to XOR: Multi-layer perceptron (MLP)




Can approximate any function!

Gradient descent can be used $\text{logsig}(w_1\text{logsig}(\ldots) + w_2\text{logsig}(\ldots) - b)$

black box params: # of hidden layers & # neurons

## ESIM 7 Sini-signaalin opetus Matlabissa

or do normalization

```
t = 0:0.1:10;
y = sin(2*pi*3/10*t);
plot(t,y);
net = newff([0 10],[n 1],{'tansig','tansig'});   % 1, 3, 5, 10
plot(t,y,'k-',t,sim(net,t),'k-');
net = train(net,t,y); %ajettava muutaman kerran
(net.trainParam.epochs)
```

test also outside the input intaval

## ESIM. 8 Ylioppiminen ja lokaali minimi

Huomattiin jo esim. 7 että neuronien lisäys parantaa tarkkuutta eli mitsei niin paljon kuin mahdollista