MONASH University

ETC3555

Statistical Machine Learning

Neural networks

21 August 2018

Outline (subject to change)

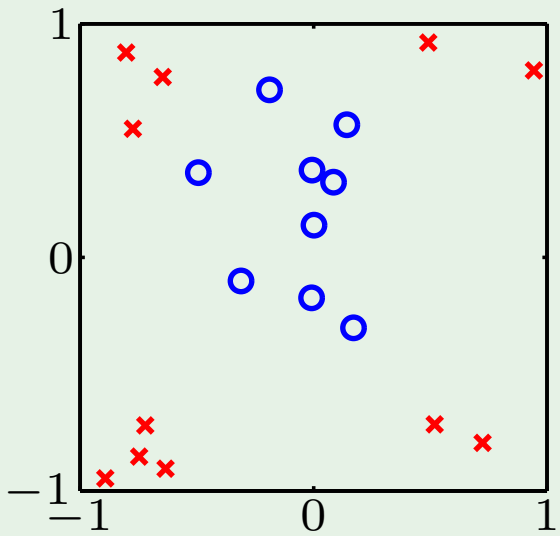
Week	Topic
1	Introduction/The learning problem
2	The learning problem
3	Linear models
4	Gradient descent
5	Neural Networks
6	Neural Networks
7	Deep Neural Networks
8	Support vector machines
9	Recommender systems and matrix completion
	Semester break
10	Text mining
11	Social networks
12	Project presentation

Outline

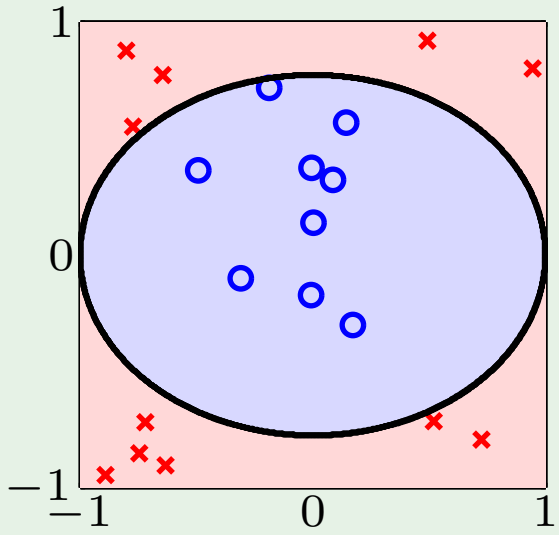
Linear is limited

Linear is limited

Data:



Hypothesis:



Linear in what?

Linear in what?

Linear regression implements

$$\sum_{i=0}^{d} w_i \, x_i$$

Linear classification implements

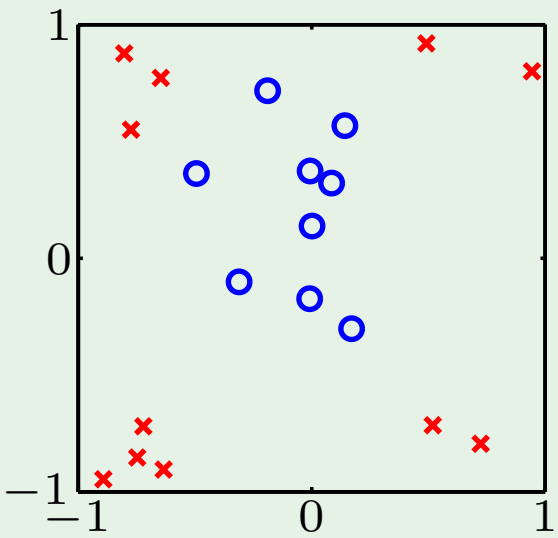
$$\text{sign}\left(\sum_{i=0}^{d} w_i \, x_i\right)$$

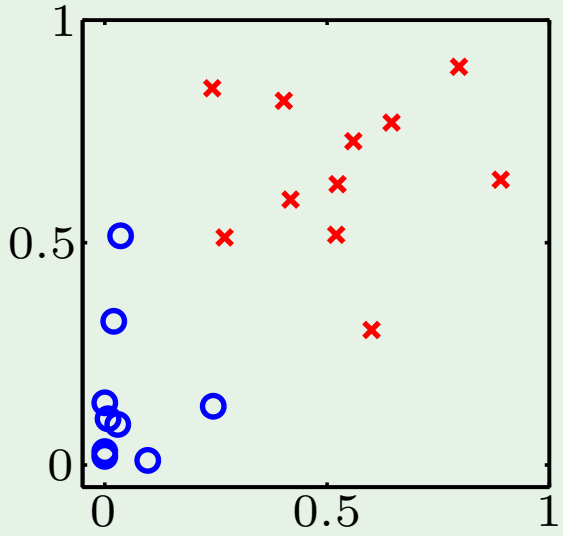
Algorithms work because of **linearity in the weights**

Transformation

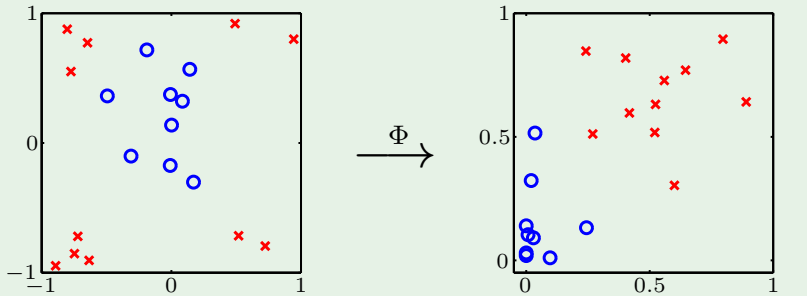
Transform the data nonlinearly

$$(x_1, x_2) \xrightarrow{\Phi} (x_1^2, x_2^2)$$



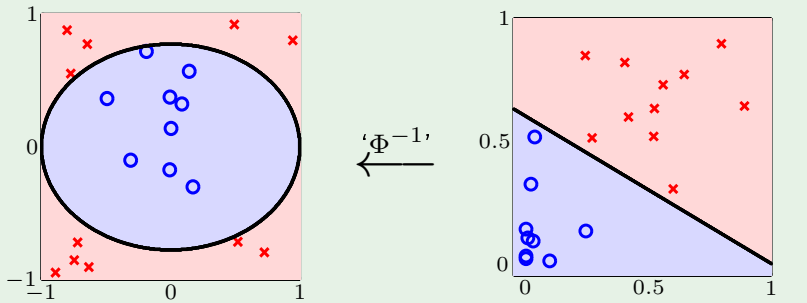


Transformation



1. Original data
$\mathbf{x}_n \in \mathcal{X}$

2. Transform the data
$\mathbf{z}_n = \Phi(\mathbf{x}_n) \in \mathcal{Z}$



4. Classify in $\mathcal{X}$-space
$g(\mathbf{x}) = \tilde{g}(\Phi(\mathbf{x})) = \text{sign}(\tilde{\mathbf{w}}^{\mathsf{T}}\Phi(\mathbf{x}))$

3. Separate data in $\mathcal{Z}$-space
$\tilde{g}(\mathbf{z}) = \text{sign}(\tilde{\mathbf{w}}^{\mathsf{T}}\mathbf{z})$

What transforms to what

What transforms to what

$$\mathbf{x} = (x_0, x_1, \cdots, x_d) \xrightarrow{\Phi} \mathbf{z} = (z_0, z_1, \cdots\cdots\cdots, z_{\tilde{d}})$$

$$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N \xrightarrow{\Phi} \mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_N$$

$$y_1, y_2, \cdots, y_N \xrightarrow{\Phi} y_1, y_2, \cdots, y_N$$

No weights in $\mathcal{X}$ $\qquad$ $\tilde{\mathbf{w}} = (w_0, w_1, \cdots\cdots\cdots, w_{\tilde{d}})$

$$g(\mathbf{x}) = \text{sign}(\tilde{\mathbf{w}}^{\mathsf{T}}\Phi(\mathbf{x}))$$

Nonlinear transforms

Nonlinear transforms

$$\mathbf{x} = (x_0, x_1, \cdots, x_d) \xrightarrow{\Phi} \mathbf{z} = (z_0, z_1, \cdots\cdots\cdots, z_{\tilde{d}})$$

Each $z_i = \phi_i(\mathbf{x})$ $\qquad$ $\mathbf{z} = \Phi(\mathbf{x})$

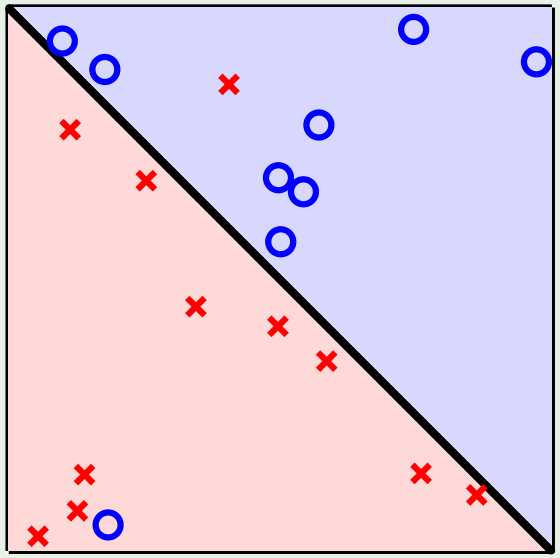
Example: $\mathbf{z} = (1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$

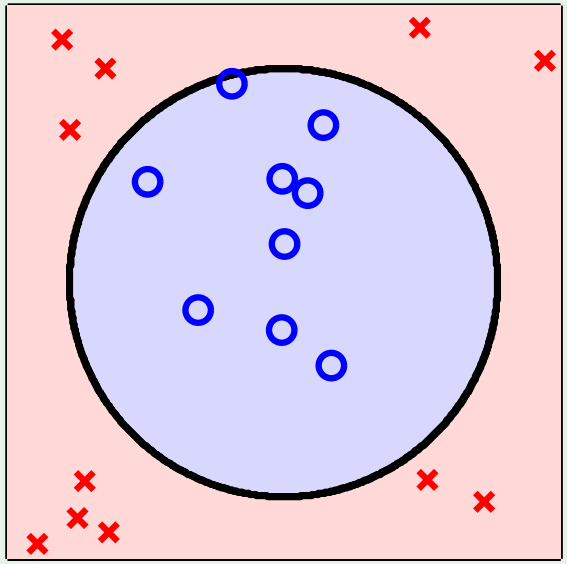
Final hypothesis $g(\mathbf{x})$ in $\mathcal{X}$ space:

$\text{sign}\left(\tilde{\mathbf{w}}^\mathsf{T}\Phi(\mathbf{x})\right)$ $\qquad$ or $\qquad$ $\tilde{\mathbf{w}}^\mathsf{T}\Phi(\mathbf{x})$

Two cases

Two non-separable cases





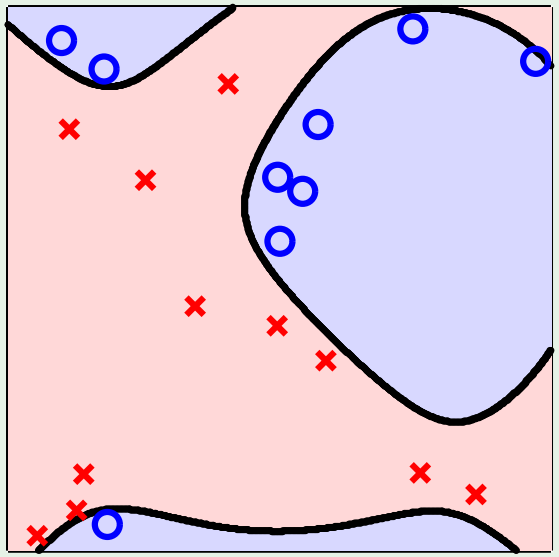
First case

First case

Use a linear model in $\mathcal{X}$; accept $E_{\text{in}} > 0$

or

Insist on $E_{\text{in}} = 0$; go to high-dimensional $\mathcal{Z}$



Second case

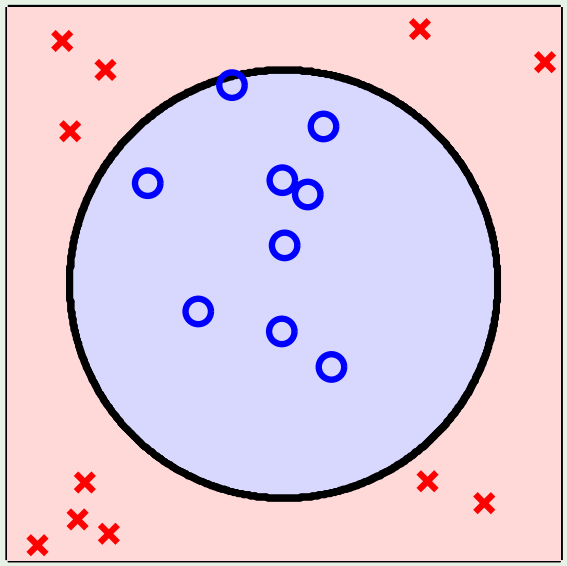
Second case

$\mathbf{z} = (1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$

Why not: $\mathbf{z} = (1, x_1^2, x_2^2)$

or better yet: $\mathbf{z} = (1, x_1^2 + x_2^2)$

or even: $\mathbf{z} = (x_1^2 + x_2^2 - 0.6)$



Data snooping

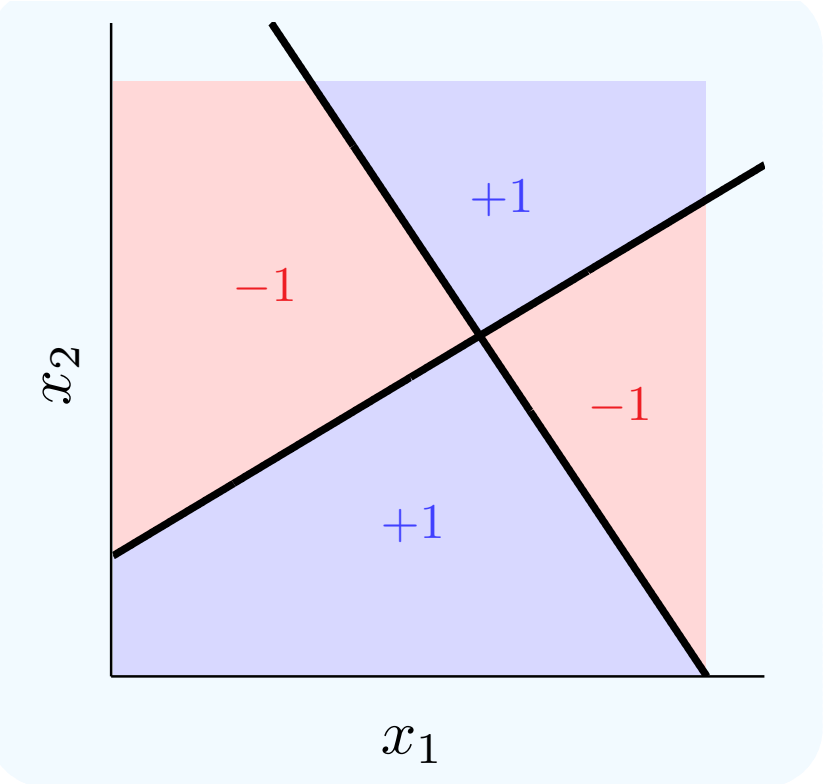
Lesson learned

Looking at the data *before* choosing the model can be hazardous to your E_{out}

Data snooping

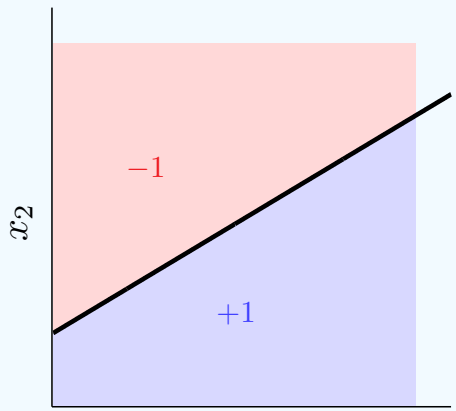
Outline

Example

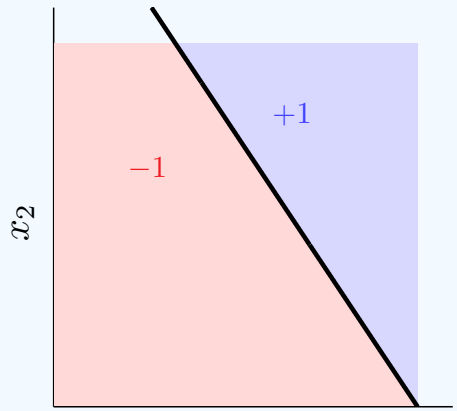


Can we learn this function with perceptrons?

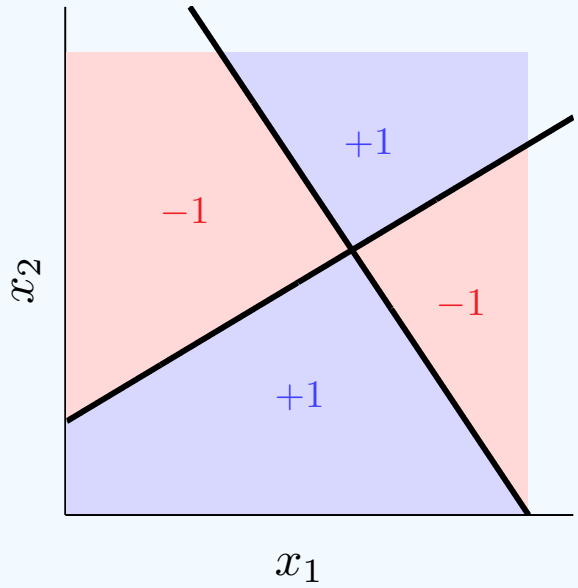
Example



$h_1(\mathbf{x}) = \text{sign}(\mathbf{w}_1^{\text{T}}\mathbf{x})$

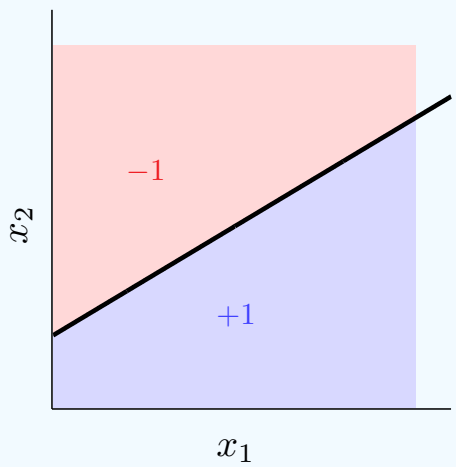


$h_2(\mathbf{x}) = \text{sign}(\mathbf{w}_2^{\text{T}}\mathbf{x})$

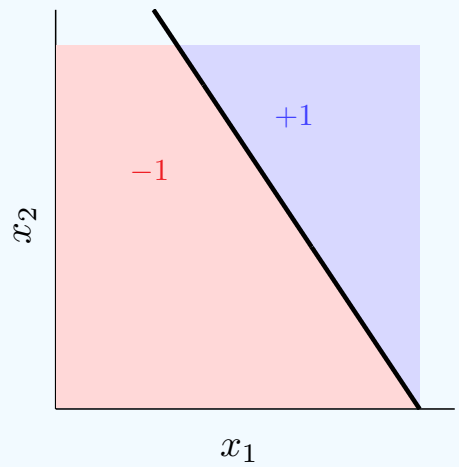


How to learn the function with these two hypotheses?

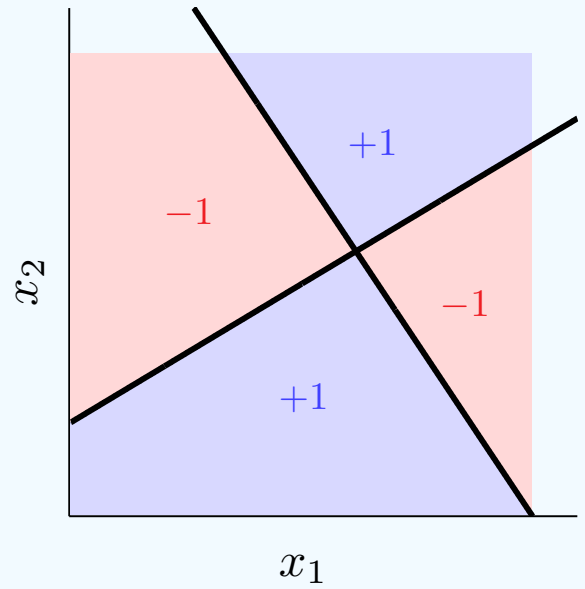
Example



$h_1(\mathbf{x}) = \text{sign}(\mathbf{w}_1^{\text{T}}\mathbf{x})$



$h_2(\mathbf{x}) = \text{sign}(\mathbf{w}_2^{\text{T}}\mathbf{x})$



How to learn the function with these two hypotheses?

The target $f(x_1, x_2) = +1$ when $h_1(x_1, x_2) = +1$ and $h_2(x_1, x_2) = -1$ or when $h_1(x_1, x_2) = -1$ and $h_2(x_1, x_2) = +1$.
This is the Boolean XOR function: $f = \text{XOR}(h_1, h_2)$.

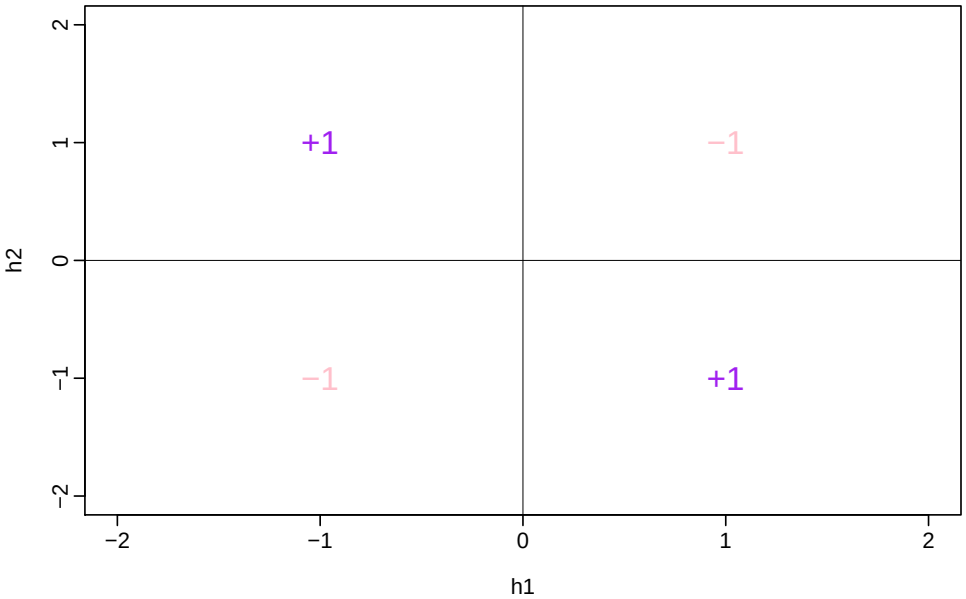
The XOR function

1 = TRUE and 0 = FALSE

h_1	h_2	XOR(h_1, h_2)
1	0	1
1	1	0
0	1	1
0	0	0

+1 = TRUE and -1 = FALSE

h_1	h_2	XOR(h_1, h_2)
+1	-1	+1
+1	+1	-1
-1	+1	+1
-1	-1	-1



Can we use a perceptron?

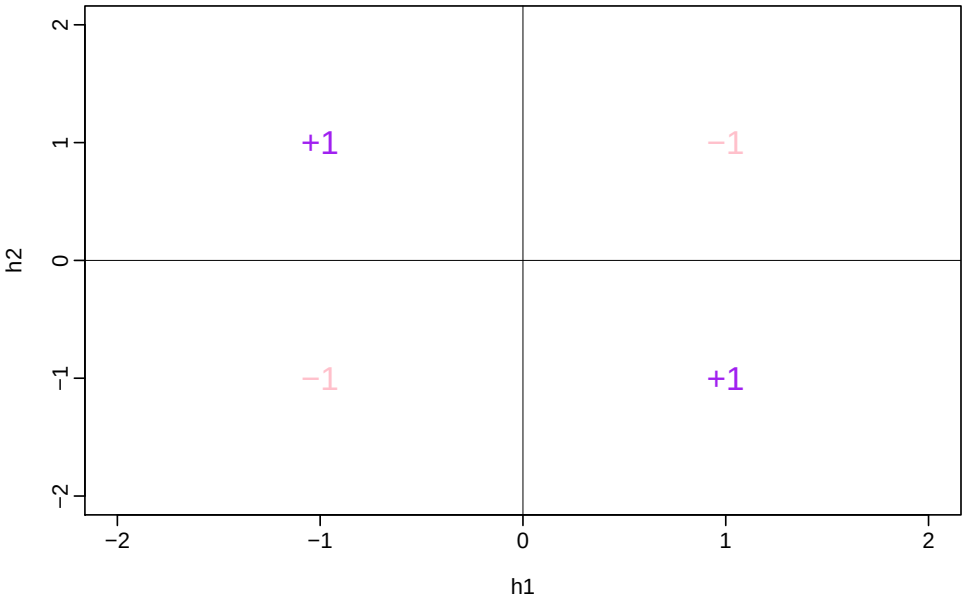
The XOR function

1 = TRUE and 0 = FALSE

h_1	h_2	XOR(h_1, h_2)
1	0	1
1	1	0
0	1	1
0	0	0

+1 = TRUE and -1 = FALSE

h_1	h_2	XOR(h_1, h_2)
+1	-1	+1
+1	+1	-1
-1	+1	+1
-1	-1	-1



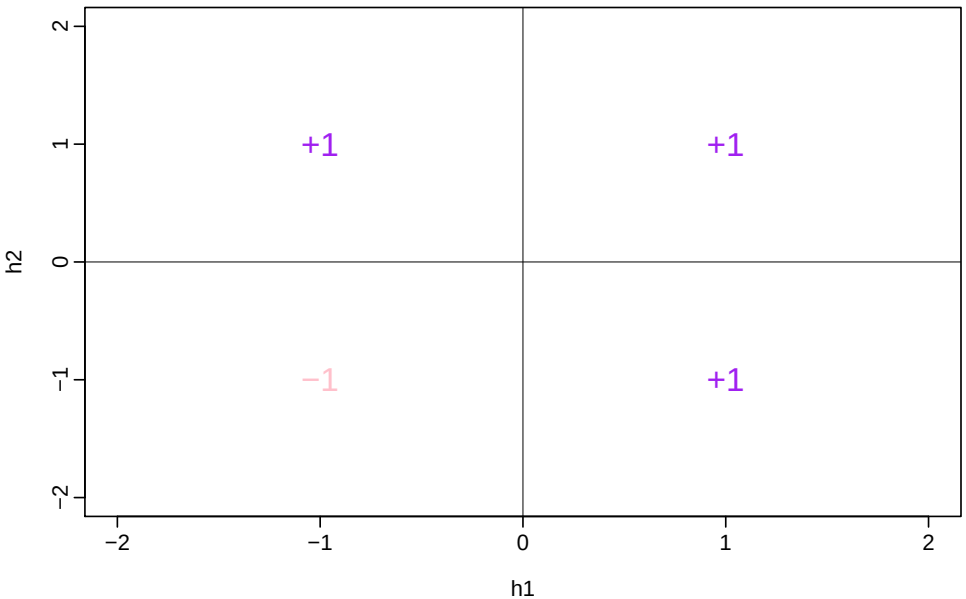
Can we use a perceptron?

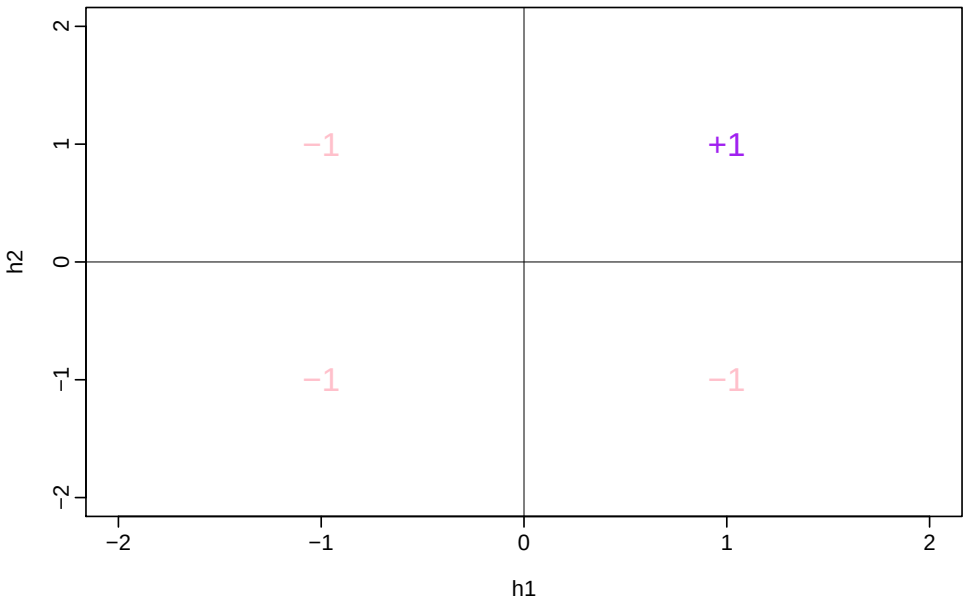
No! Not linearly separable.

The OR and AND functions

h_1	h_2	OR(h_1, h_2)
+1	-1	+1
+1	+1	+1
-1	+1	+1
-1	-1	-1

h_1	h_2	AND(h_1, h_2)
+1	-1	-1
+1	+1	+1
-1	+1	-1
-1	-1	-1



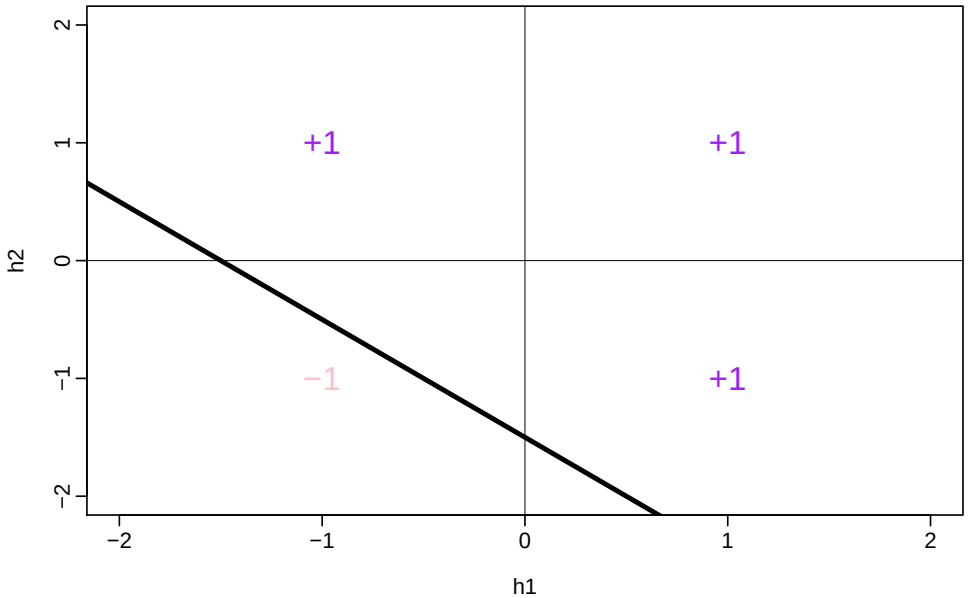


What are the perceptron weights for both OR and AND?

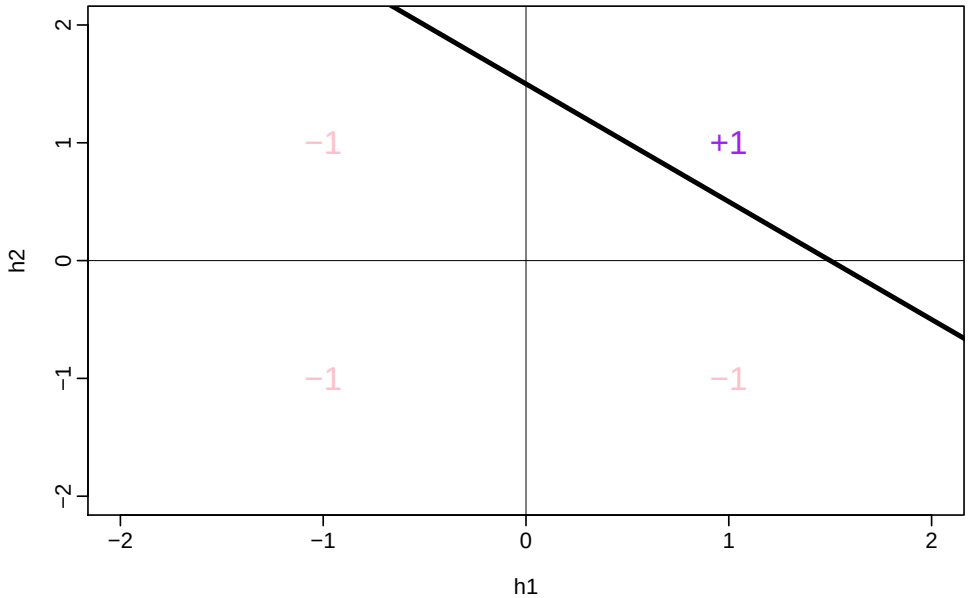
The OR and AND functions

h_1	h_2	OR(h_1, h_2)
+1	-1	+1
+1	+1	+1
-1	+1	+1
-1	-1	-1

h_1	h_2	AND(h_1, h_2)
+1	-1	-1
+1	+1	+1
-1	+1	-1
-1	-1	-1



OR(h_1, h_2) = sign(h_1 + h_2 + 1.5)



AND(h_1, h_2) = sign(h_1 + h_2 - 1.5)

Back to the XOR function

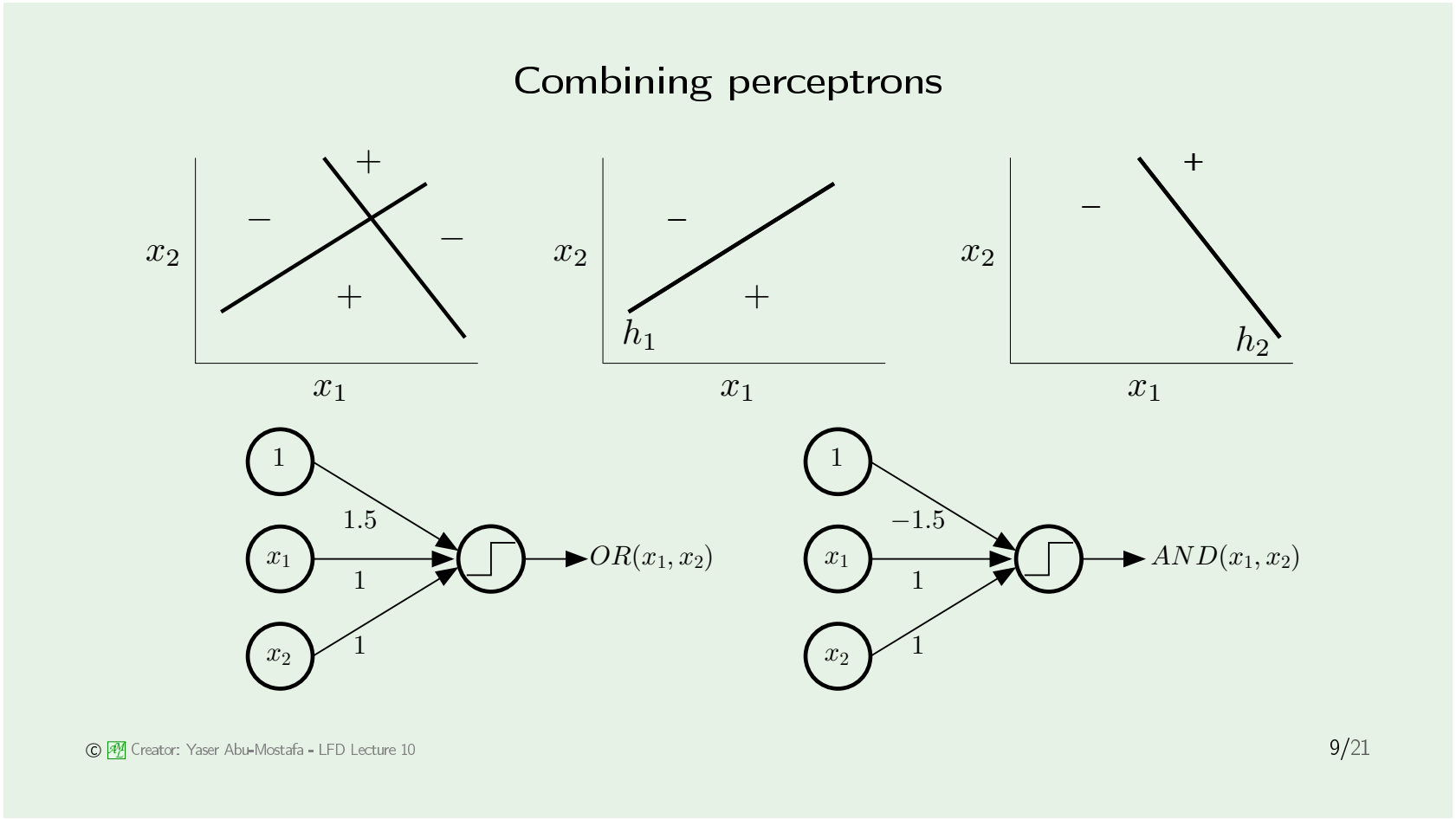
- We can write f using the simpler OR and AND operations:

$$\begin{aligned} f &= \text{XOR}(h_1, h_2) \\ &= \text{OR}(\text{AND}(h_1, \text{NOT}(h_2)), \text{AND}(\text{NOT}(h_1), h_2)) \\ &= h_1\bar{h}_2 + \bar{h}_1 h_2 \end{aligned}$$

$\rightarrow$ Prove it

- This is a good news because OR and AND functions can be implemented by the perceptron.
- In other words, the (more complicated) target f is essentially a combination of perceptrons.

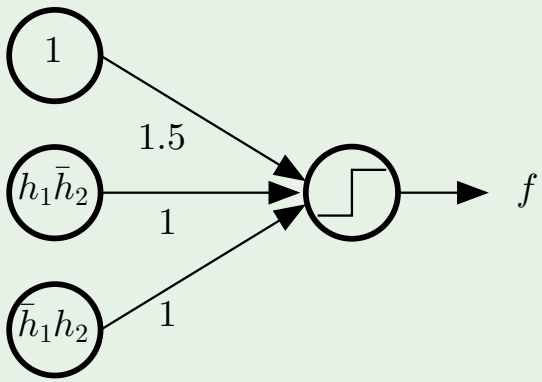
Combining perceptrons

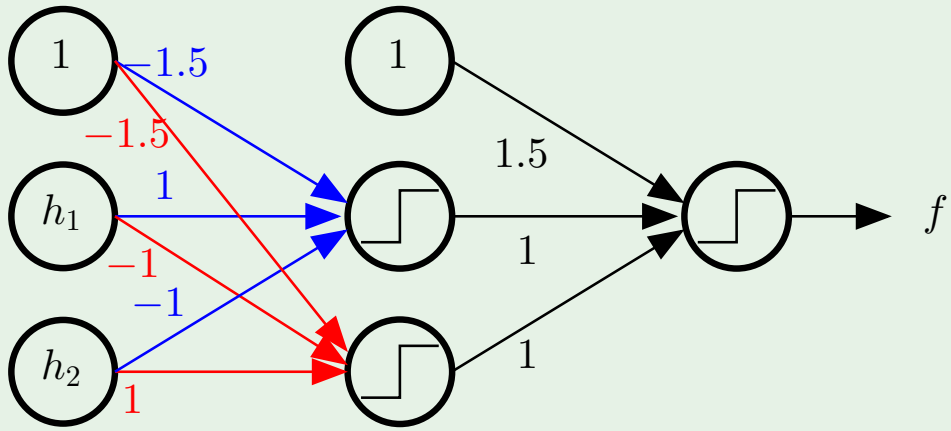


The *computation graph* gives a convenient graph representation of the different operations to compute a function.

Creating layers

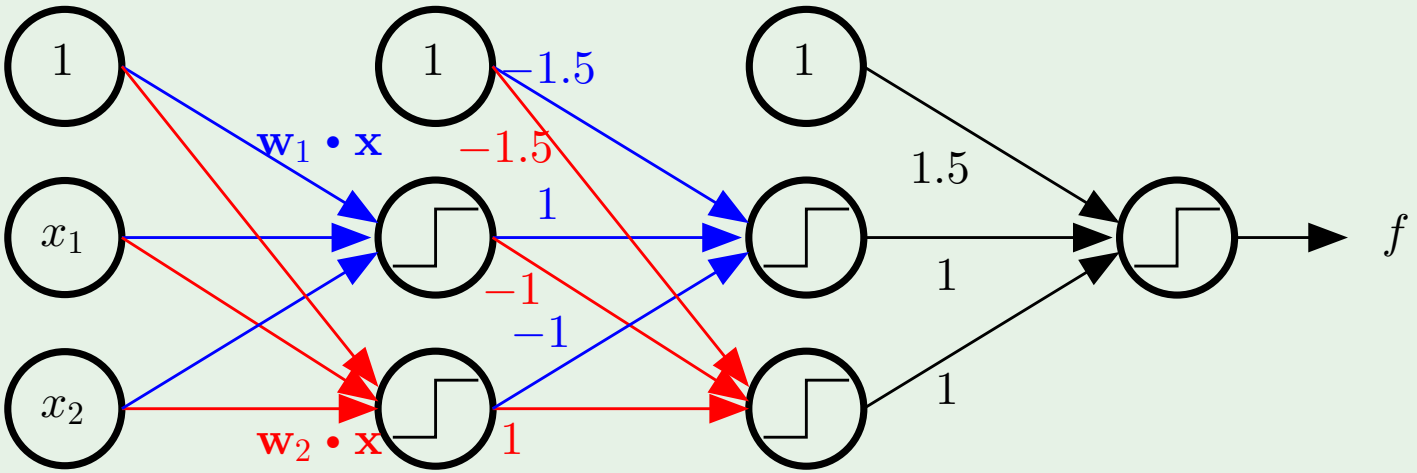
Creating layers





The multilayer perceptron (MLP)

The multilayer perceptron



3 layers "feedforward"

Exercise

Use the computation graph to write an explicit formula for f.

$$f(x_1, x_2) = f(\boldsymbol{x}) = ??$$

Exercise

Use the computation graph to write an explicit formula for f.

$$f(x_1, x_2) = f(\boldsymbol{x}) = ??$$

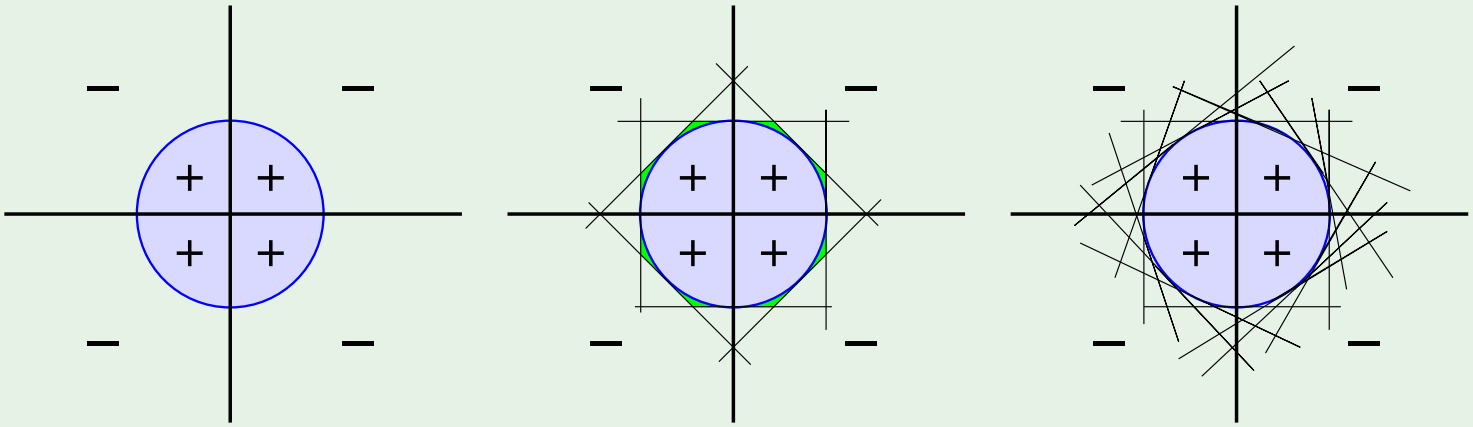
$$f(\boldsymbol{x}) = \text{sign}\left[\text{sign}\left(h_1(\boldsymbol{x}) - h_2(\boldsymbol{x}) - \frac{3}{2}\right) - \text{sign}\left(h_1(\boldsymbol{x}) - h_2(\boldsymbol{x}) + \frac{3}{2}\right) + \frac{3}{2}\right]$$

where

$$h_1(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}_1^T \boldsymbol{x}) \text{ and } h_2(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}_2^T \boldsymbol{x}).$$

A powerful model

A powerful model



Target 8 perceptrons 16 perceptrons

2 red flags for **generalization** and **optimization**