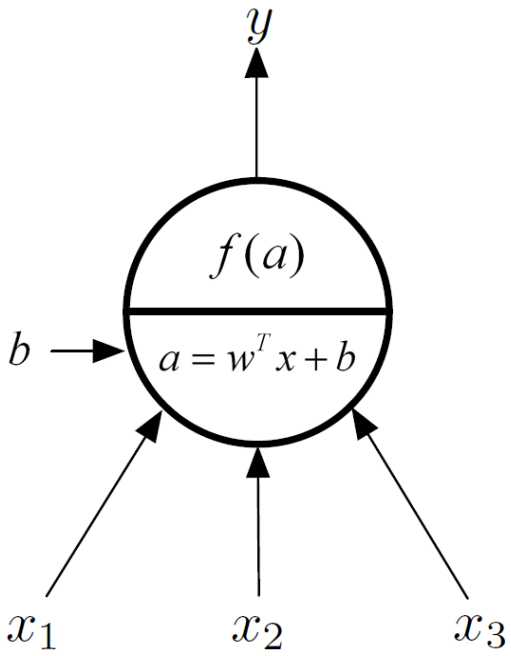
Seminar in Deep Learning

Lecture 1: Perceptron

Alexander Tkachenko
University of Tartu

16 September, 2014

Neuron model

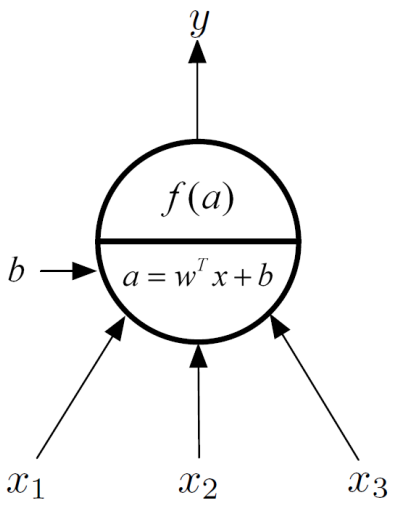


Let $x = (x_1, x_2, x_3)$ be input vector, $w = (w_1, w_2, w_3)$ be weights vector, and b - bias term.

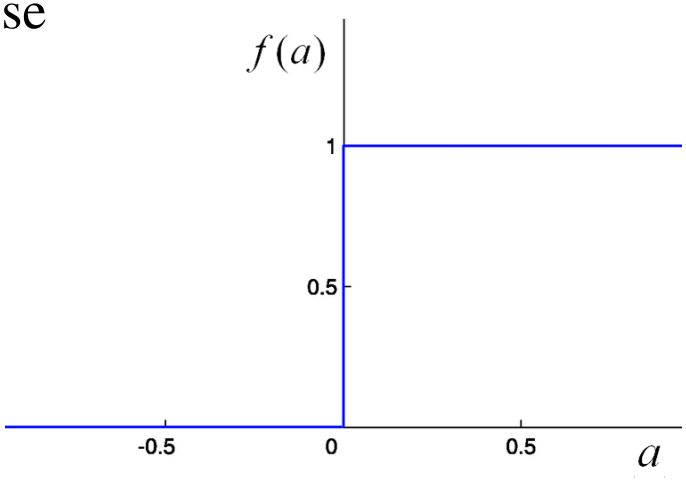
First inputs are linearly aggregated: $a = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$.

Then the output y is obtained as: $y = f(a)$.

Binary Threshold Neuron for Classification



$$y = f(a) = \begin{cases} 1 \text{ if } a \geq 0 \\ 0 \text{ otherwise} \end{cases}$$



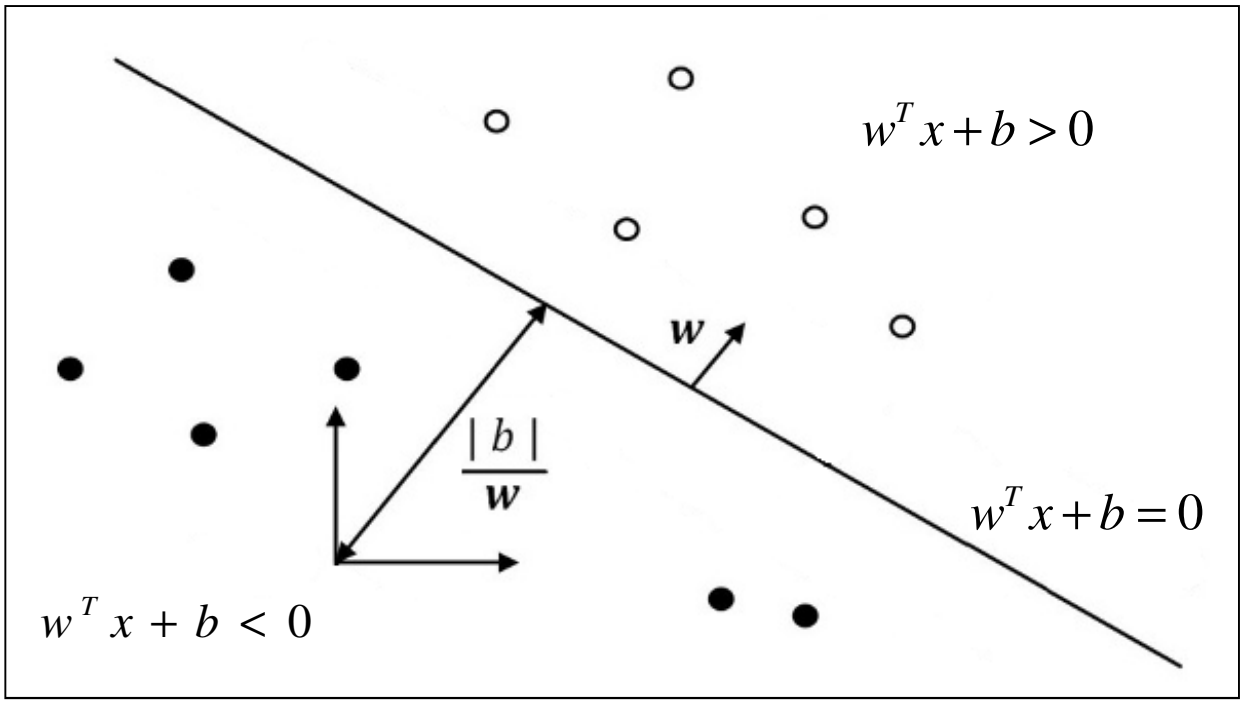
Let $x = (x_1, x_2, x_3)$ be input vector, $w = (w_1, w_2, w_3)$ be weights vector, and b - bias term.

First inputs are linearly aggregated: $a = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$.

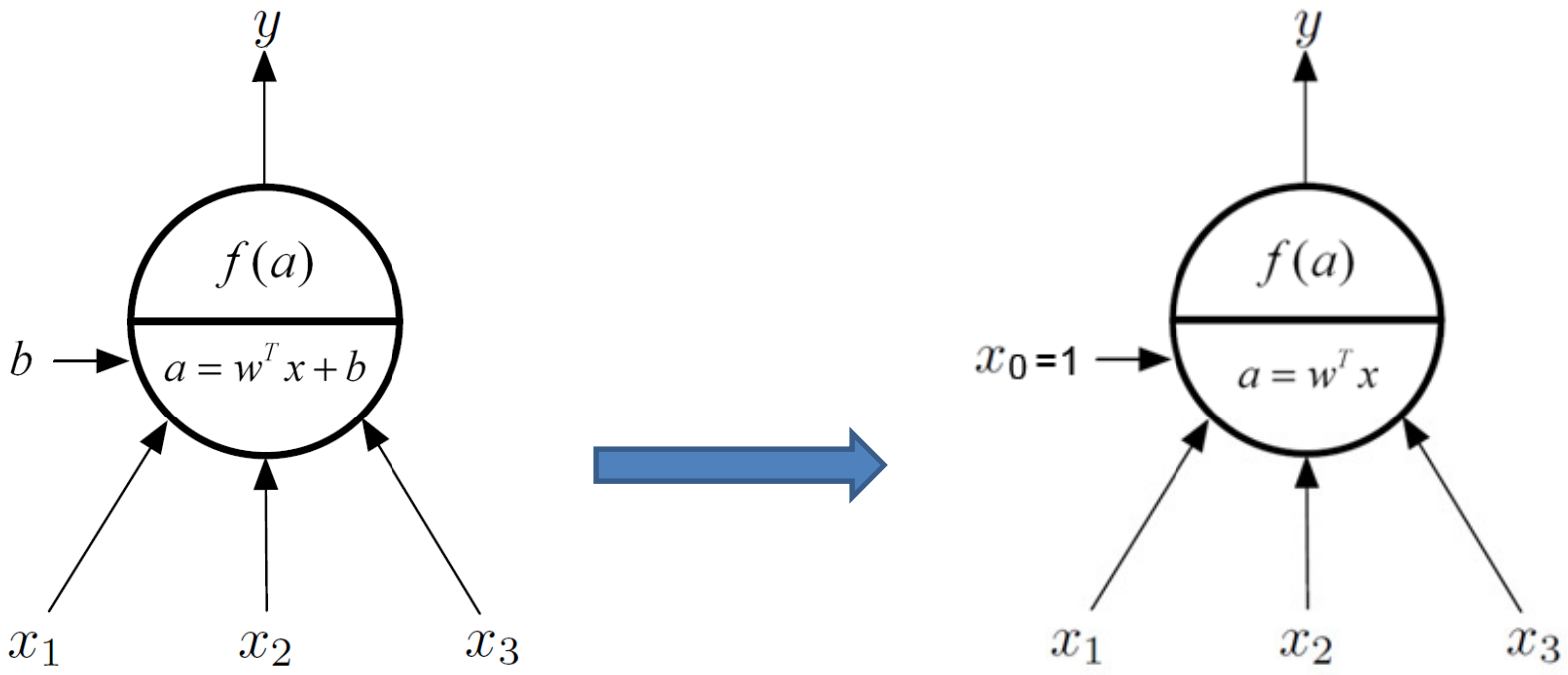
Then the output y is obtained as: $y = f(a)$.

Geometrical view

$$y = f(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad \Leftrightarrow \quad y(x) = \begin{cases} 1 & \text{if } w^T x + b \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



Eliminating bias term

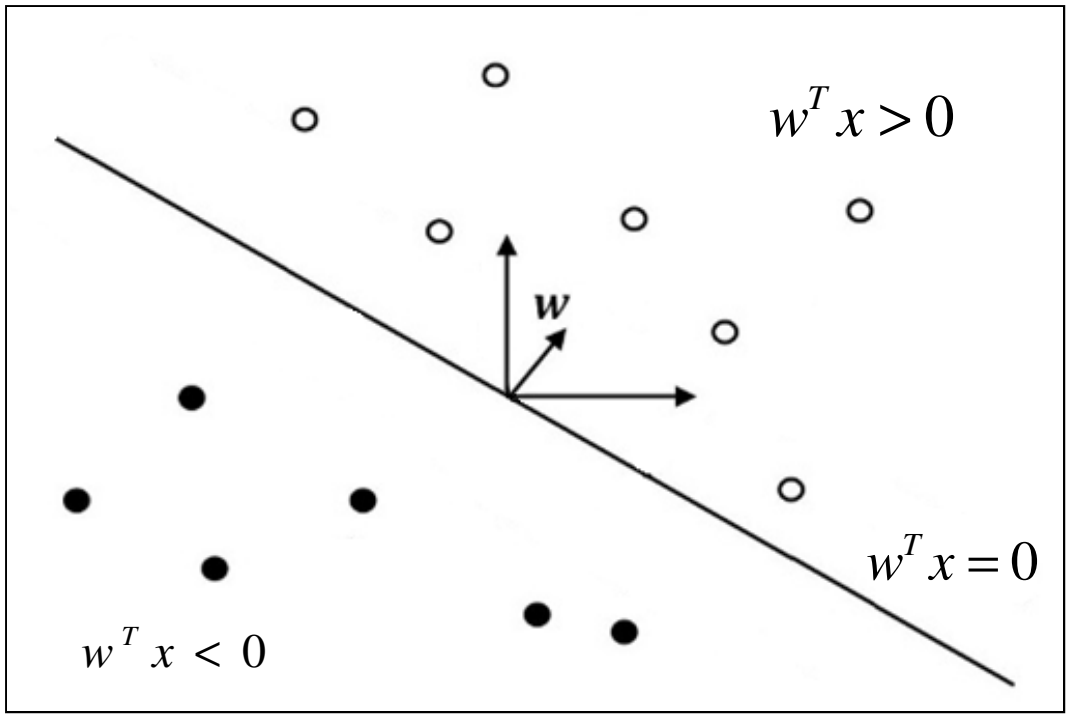


$$x = (x_1, x_2, x_3)$$
$$w = (w_1, w_2, w_3)$$
$$a = w^T x + b$$

$$x = (1, x_1, x_2, x_3)$$
$$w = (w_0, w_1, w_2, w_3)$$
$$a = w^T x$$

Geometrical view

$$y = f(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad \Leftrightarrow \quad y(x) = \begin{cases} 1 & \text{if } w^T x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

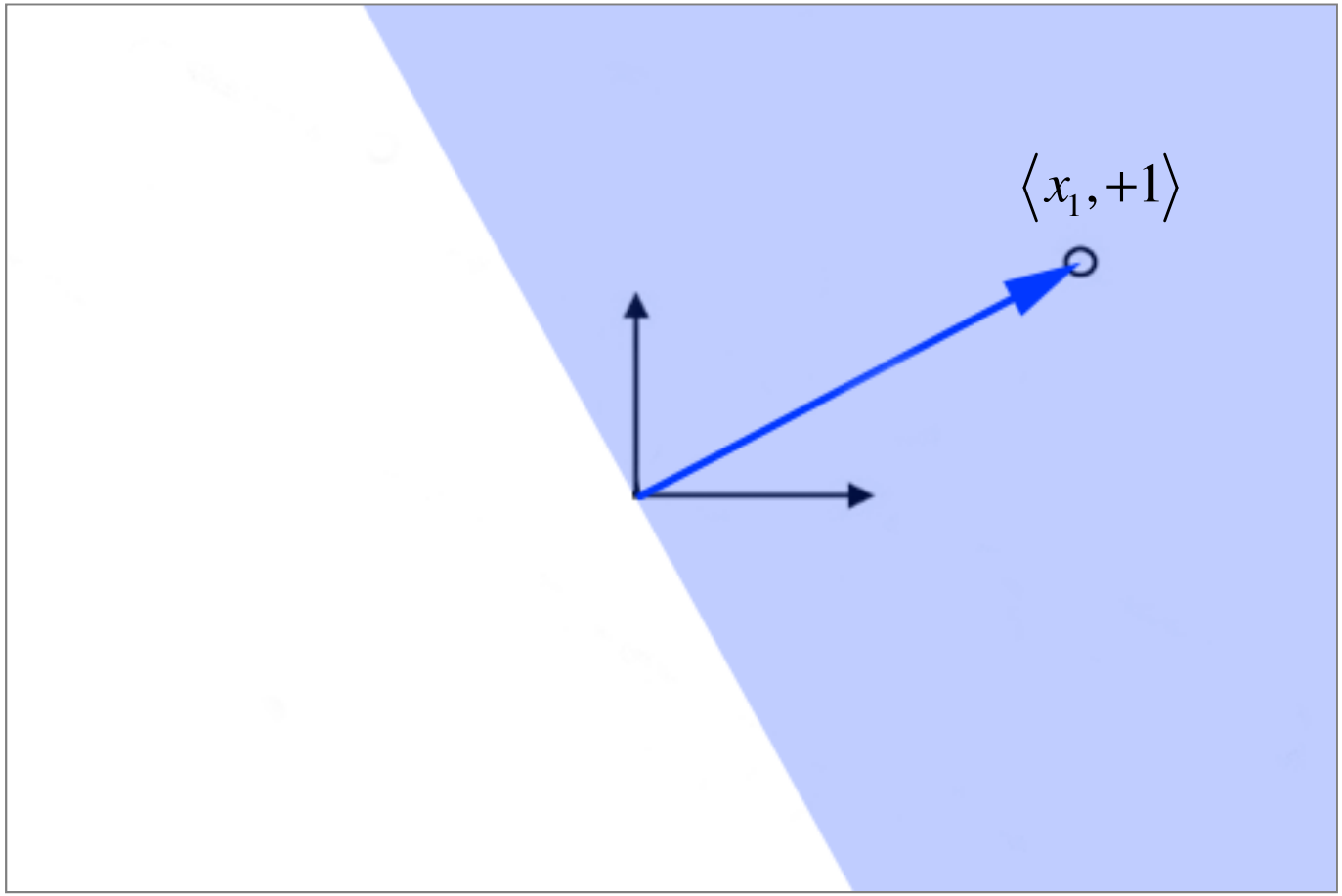


Learning a hyperplane

Given training data $\{\langle x_i, y_i \rangle\}$, $y \in \{-1,1\}$ find w such that $y_i(w^T x_i) > 0$ for all i.

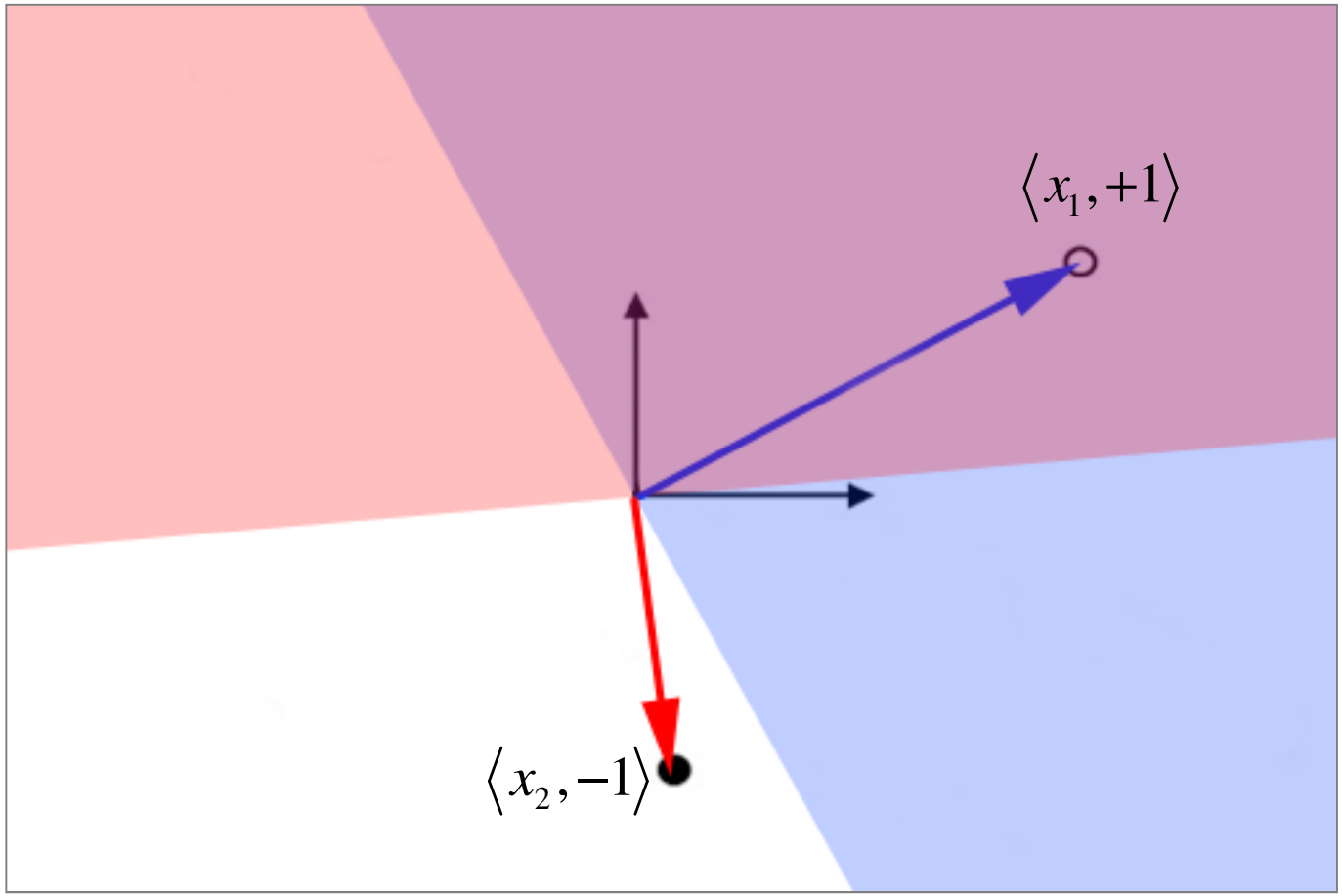
Learning a hyperplane

Given training data $\{\langle x_i, y_i \rangle\}$, $y \in \{-1,1\}$ find w such that $y_i(w^T x_i) > 0$ for all i.



Learning a hyperplane

Given training data $\{\langle x_i, y_i \rangle\}$, $y \in \{-1,1\}$ find w such that $y_i(w^T x_i) > 0$ for all i.



Learning a hyperplane

Given training data $\{\langle x_i, y_i \rangle\}$, $y \in \{-1, 1\}$ find w such that $y_i (w^T x_i) > 0$ for all i.

- What if there is no solution?
- If more than one solution *w* exists, then which one should we choose?
- How to solve the problem in a simple and efficient way?

Perceptron algorithm

Given training data $\{\langle x_i, y_i \rangle\}$, $y \in \{-1, 1\}$ find w such that $y_i (w^T x_i) > 0$ for all i.

initialize $w = 0$

while any training example $\langle x, y \rangle$ is not classified correctly

set $w := w + yx$

Perceptron: online version

Given an infinite stream of training examples $\{\langle x_i, y_i \rangle\}$

where time is indexed $t = 1, 2, 3...,$ this version is as follows:

initialize $w = 0$

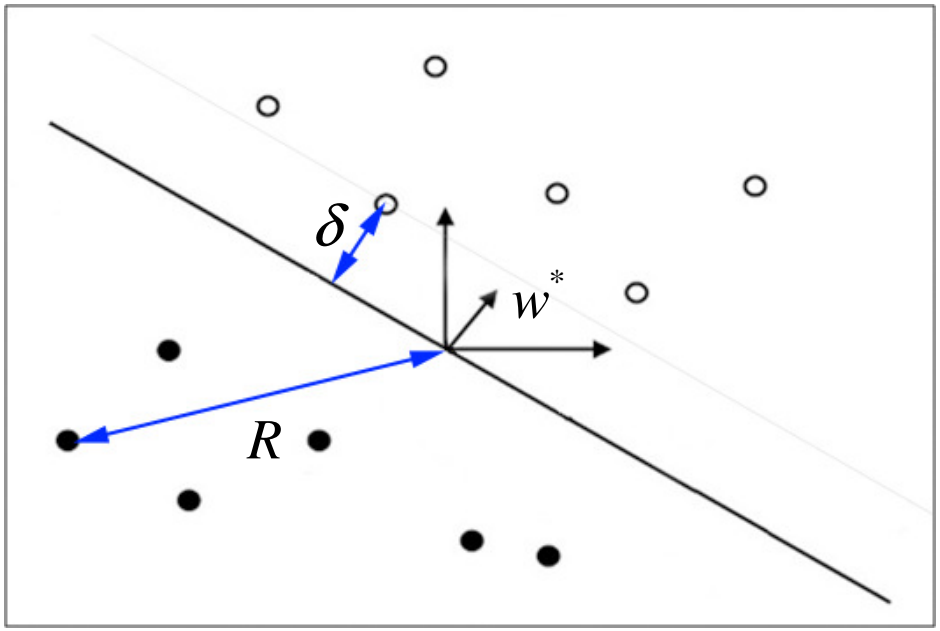
for $t = 1, 2, 3, ...$

if $y_t (w x_t) \leq 0$ then set $w := w + y_t x_t$

Perceptron convergence theorem [Novikoff, 1962]

Assumptions: Let the training set be finite or infinite. Let $R = \max_t ||x_t||$. If the training data are all on the unit sphere, then $R = 1$ for example. Suppose that the learning task is solvable, i.e. there exists some vector w^* of unit length and some $\delta > 0$ such that $y_t(w^* \cdot x_t) \geq \delta$ for all t.

Theorem: Under these assumptions, the perceptron algorithm converges after at most $(R/\delta)^2$ updates.



Perceptron convergence theorem [Novikoff, 1962]

Assumptions: Let the training set be finite or infinite. Let $R = \max_t ||x_t||$. If the training data are all on the unit sphere, then $R = 1$ for example. Suppose that the learning task is solvable, i.e. there exists some vector w^* of unit length and some $\delta > 0$ such that $y_t(w^* \cdot x_t) \geq \delta$ for all t.

Theorem: Under these assumptions, the perceptron algorithm converges after at most $(R/\delta)^2$ updates.

Proof: Let w_n be the w vector after n updates and let $w_0 = 0$. We will argue that whenever w is updated it becomes closer to w^*.

Suppose w_{n+1} is an update, i.e. w_n fails to classify x correctly and hence $w_{n+1} = w_n + yx$. Consider

$$w_{n+1} \cdot w^* = (w_n + yx) \cdot w^* = w_n \cdot w^* + yx \cdot w^* \geq w_n \cdot w^* + \delta.$$

This says that the projection of w_{n+1} onto w^* has increased. We would like this to mean that w_{n+1} is closer to w^*. However, what it really means is that w_{n+1} is closer to w^* and/or w_{n+1} has grown bigger. So, consider the Euclidean length of w_{n+1}:

$$||w_{n+1}||^2 = ||w_n + yx||^2 = ||w_n||^2 + 2y(w_n \cdot x) + ||x||^2 \leq ||w_n||^2 + R^2$$

since $y(w_n \cdot x) \leq 0$. Now, after N actual updates we know two facts: $||w_n||^2 \leq NR^2$ and $w_n \cdot w^* \geq N\delta$. Putting these together gives a contradiction if N is too large: $w_N \cdot w^* \leq ||w_N||||w^*|| = ||w_N||$ so $N\delta \leq ||w_N|| \leq R\sqrt{N}$ so $\sqrt{N} \leq R/\delta$.
End of proof.

The history of perceptron

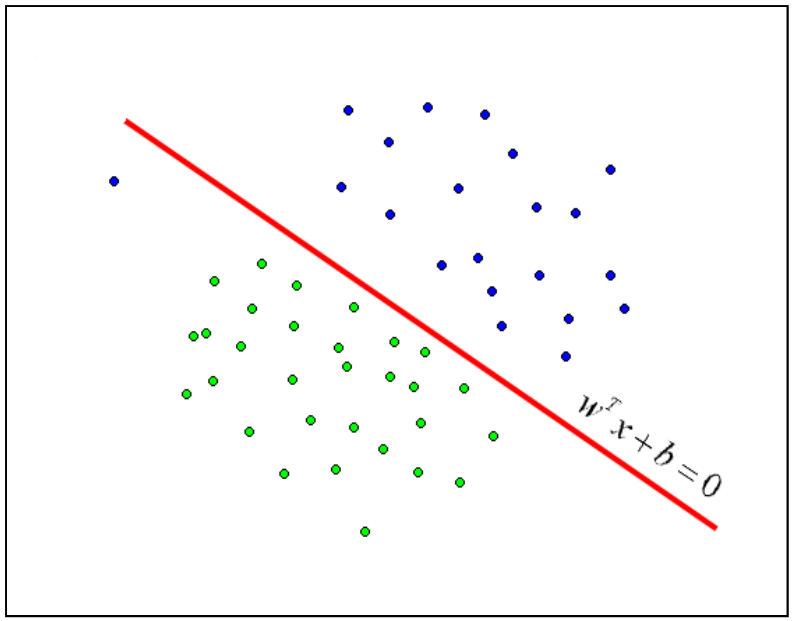
- They were popularised by Frank Rosenblatt in the early 1960's.
  - They appeared to have a very powerful learning algorithm.
  - Lots of grand claims were made for what they could learn to do.

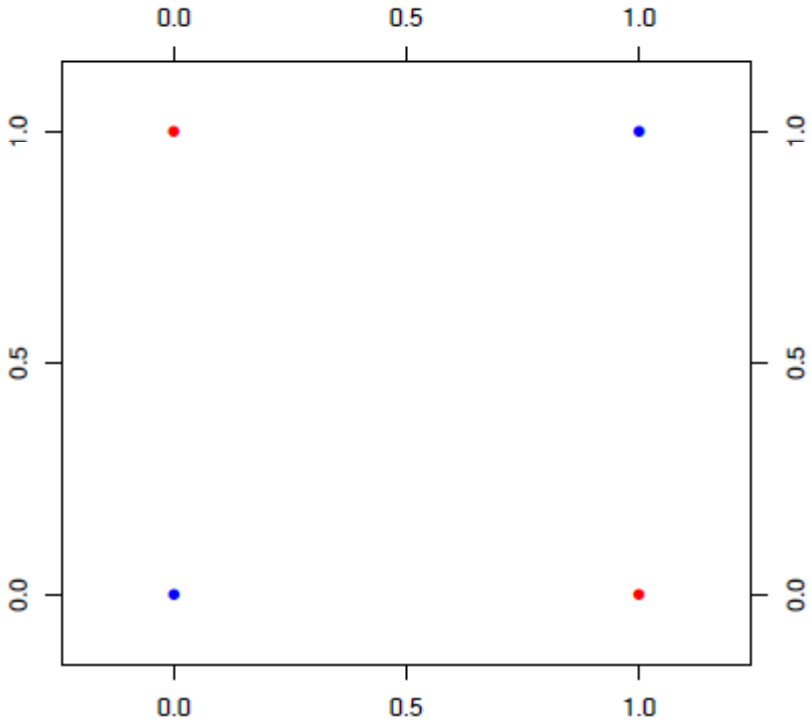
- In 1969, Minsky and Papert published a book called "Perceptrons" that analysed what they could do and showed their limitations.
  - Many people thought these limitations applied to all neural network models.

- The perceptron learning procedure is still widely used today for tasks with enormous feature vectors that contain many millions of features.

Perceptron Limitations

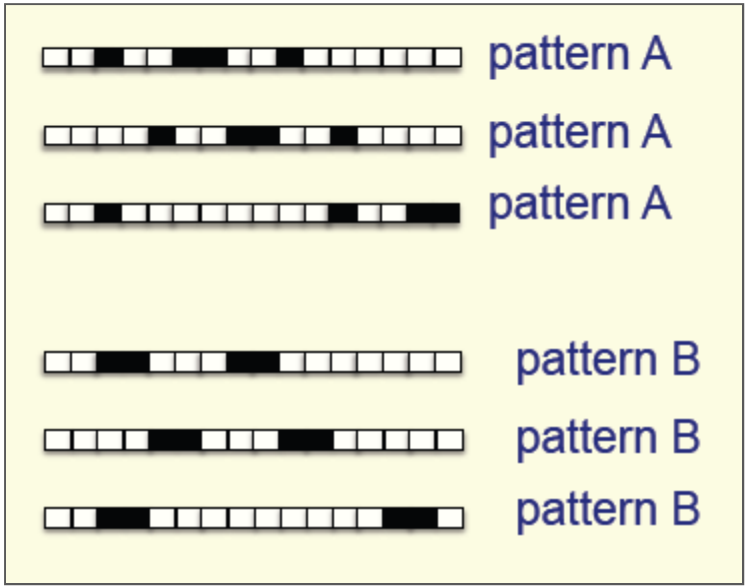
Binary threshold neuron cannot learn XOR





Limitations of a single neuron network

- Can a binary threshold unit discriminate between different patterns that have the same number of on pixels?
  - Not if the patterns can translate with wrap-around!



Learning with hidden units

- Networks without hidden units are very limited in the input-output mappings they can learn to model.
  - More layers of linear units do not help. Its still linear.
  - Fixed output non-linearities are not enough.

- We need multiple layers of adaptive, non-linear hidden units. But how can we train such nets?
  - We need an efficient way of adapting all the weights, not just the last layer. This is hard.
  - Learning the weights going into hidden units is equivalent to learning features.
  - This is difficult because nobody is telling us directly what the hidden units should do.

Materials

- Perceptron Classifiers, Charles Elkan, 2010
  http://www.cs.columbia.edu/~smaskey/CS6998-0412/supportmaterial/perceptron.pdf

- Neural Networks for Machine Learning, Lecture 2, Geoffrey Hinton
  https://d396qusza40orc.cloudfront.net/neuralnets/lecture_slides/lec2.pdf