

Security Analytics

Topic 6: Perceptron and Support Vector Machine

Purdue University

Prof. Ninghui Li

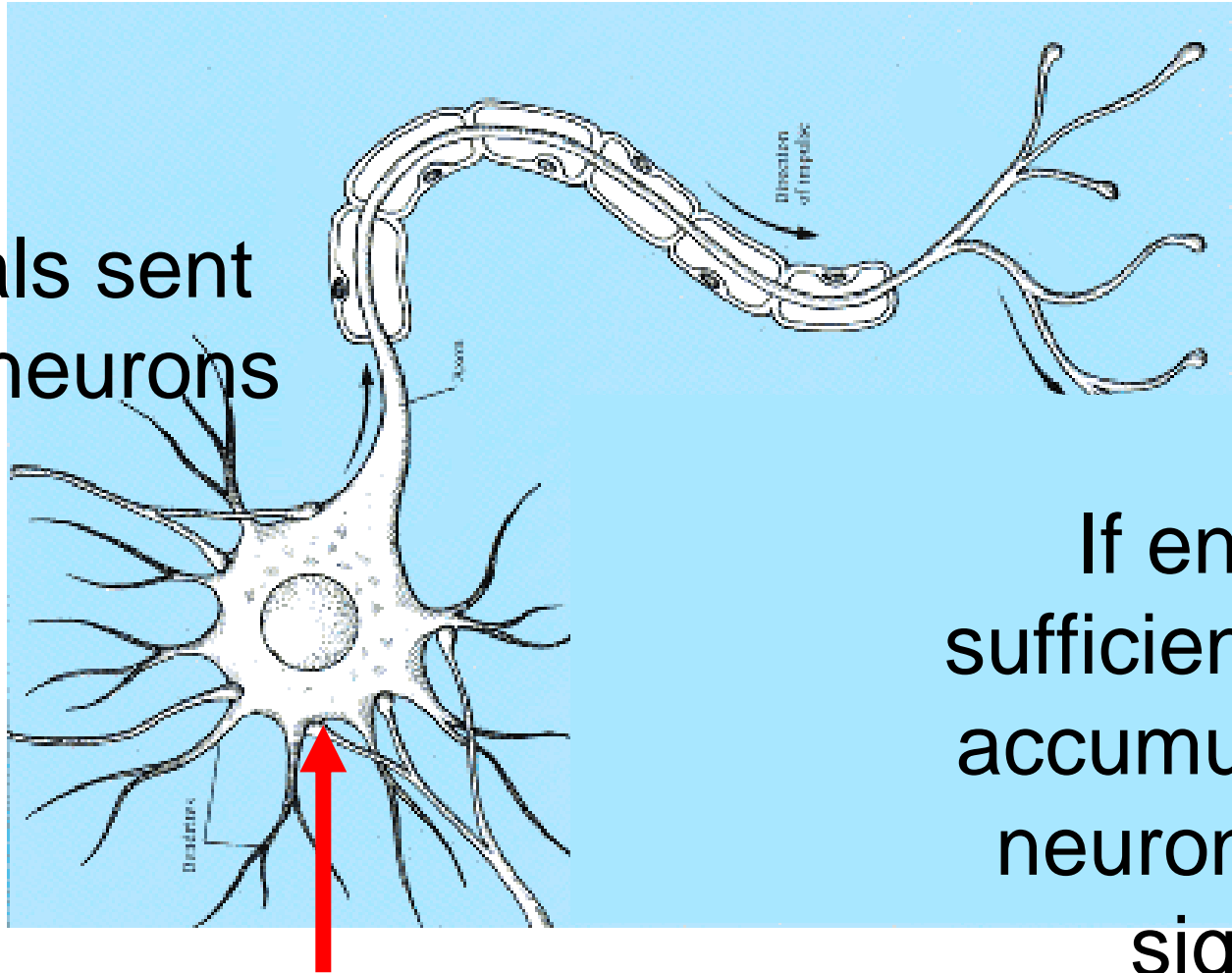
Based on slides by Prof. Jenifer Neville and
Chris Clifton

Readings

- Principle of Data Mining
 - Chapter 10: Predictive Modeling for Classification
 - 10.3 Perceptron

PERCENTRON

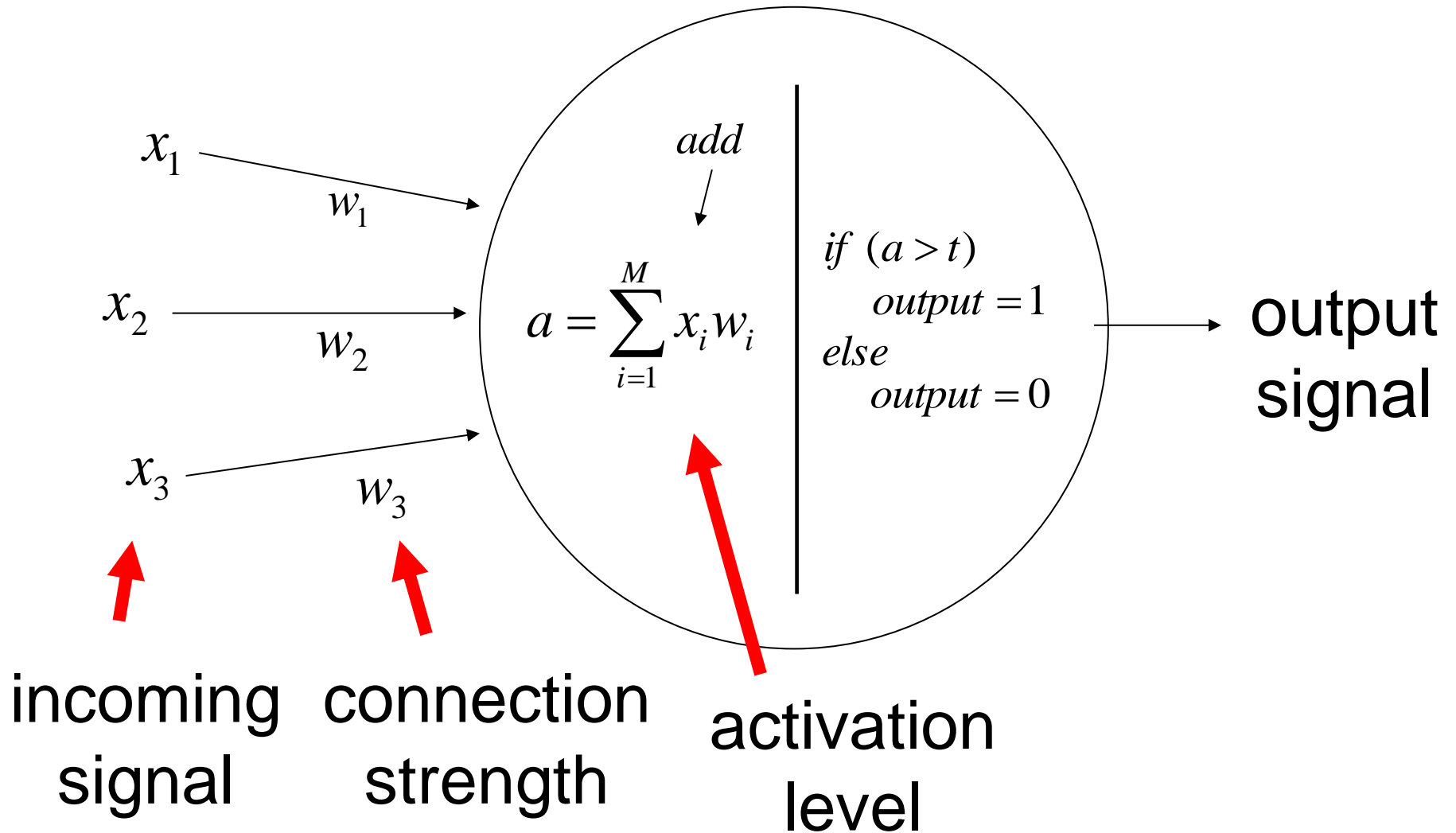
Input signals sent from other neurons



If enough sufficient signals accumulate, the neuron fires a signal.

Connection strengths determine how the signals are accumulated

- input signals 'x' and coefficients 'w' are multiplied
 - weights correspond to connection strengths
- signals are added up – if they are enough, FIRE!

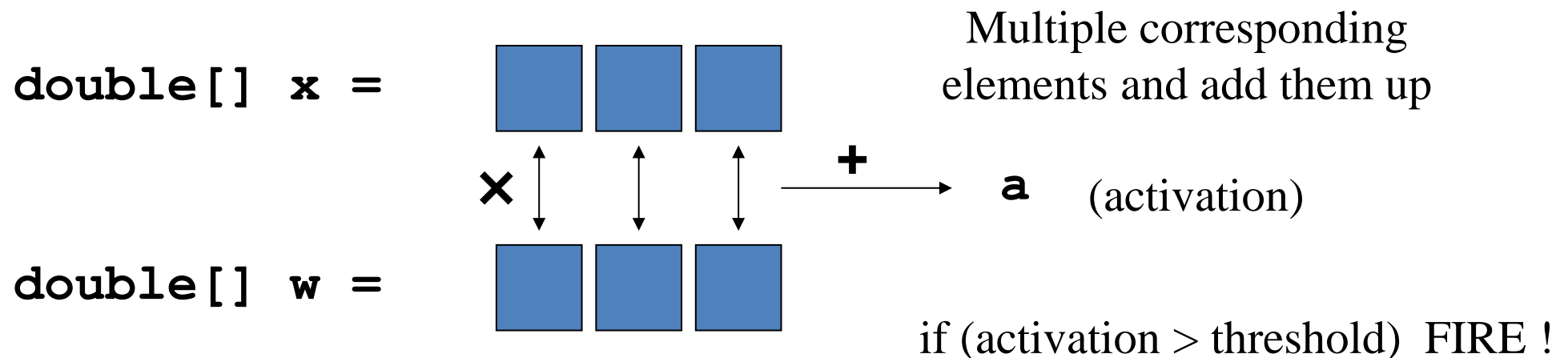


Calculation...

$$a = \sum_{i=1}^M x_i w_i$$

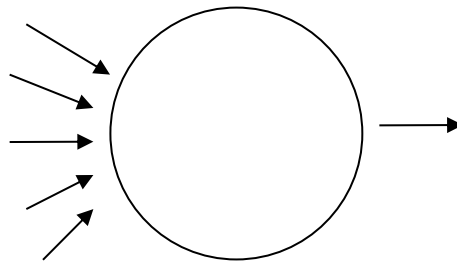
Sum notation

(just like a loop from 1 to M)

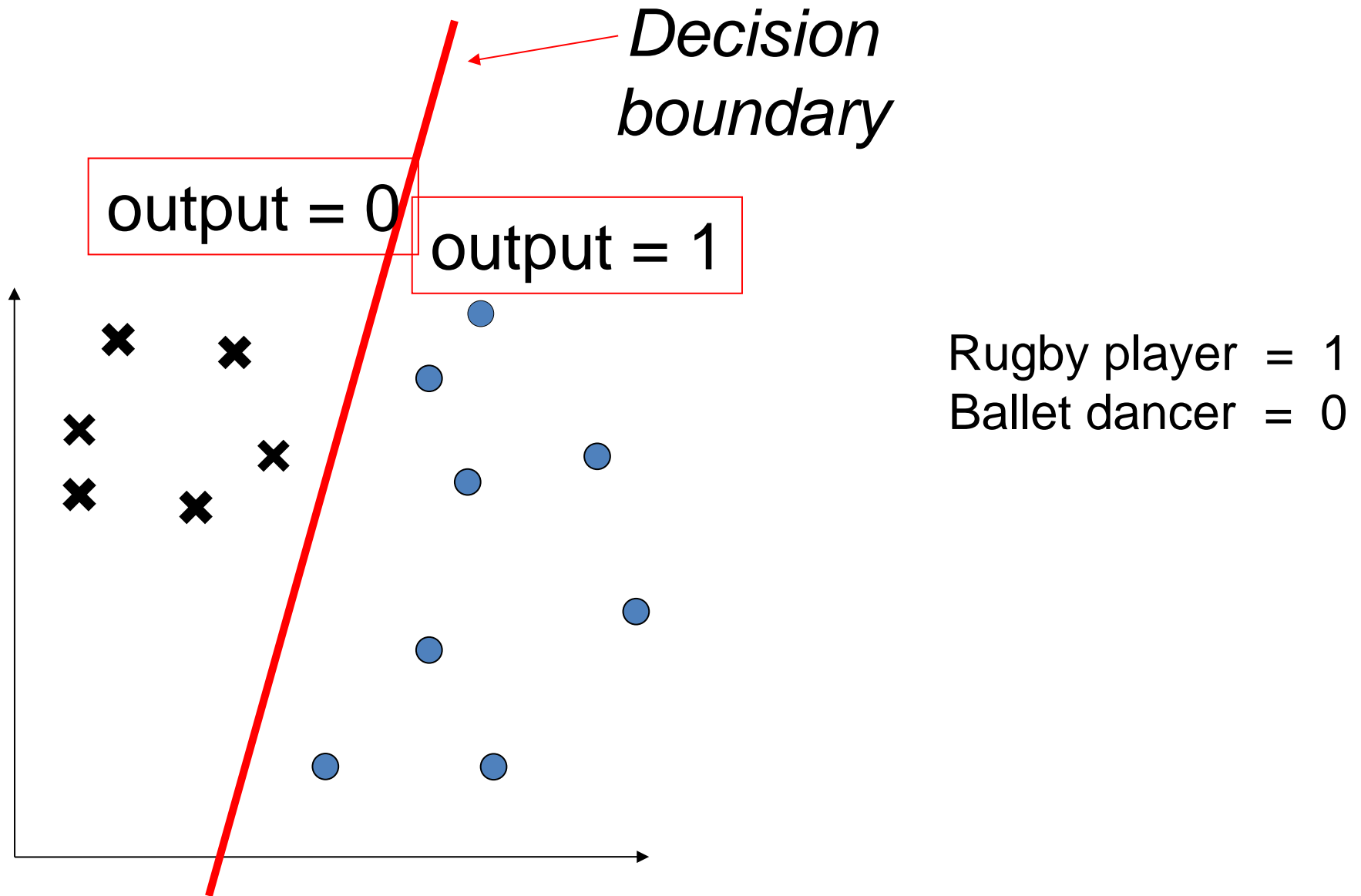


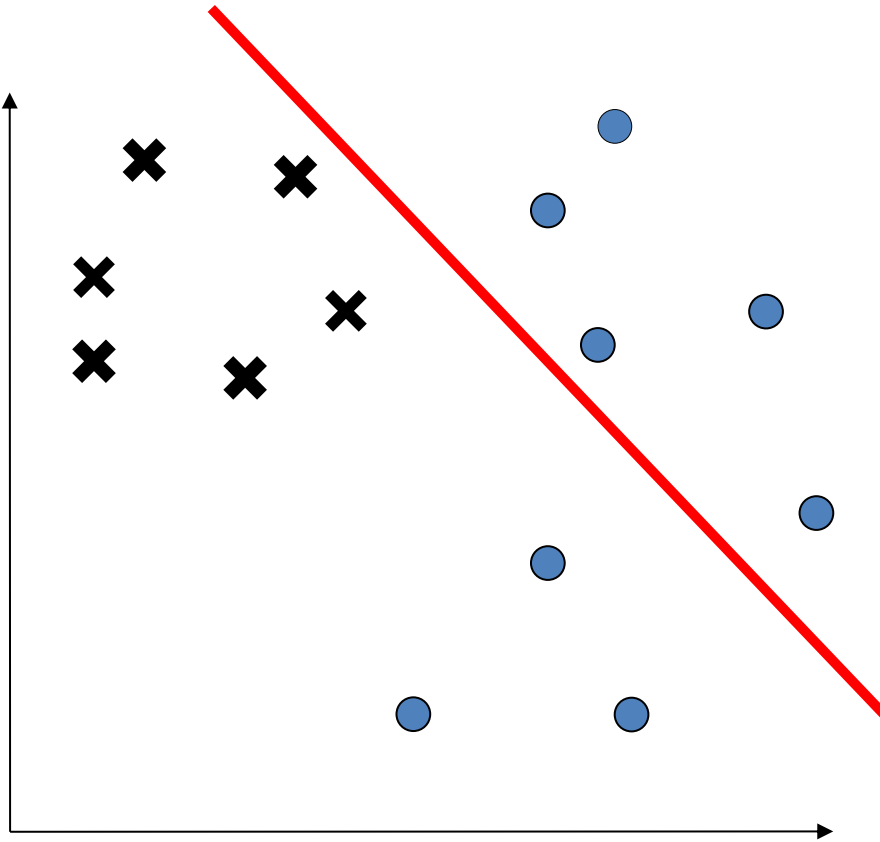
The Perceptron Decision Rule

$$\text{if } \left(\sum_{i=1}^M x_i w_i \right) > t \quad \text{then } \textit{output} = 1, \text{ else } \textit{output} = 0$$



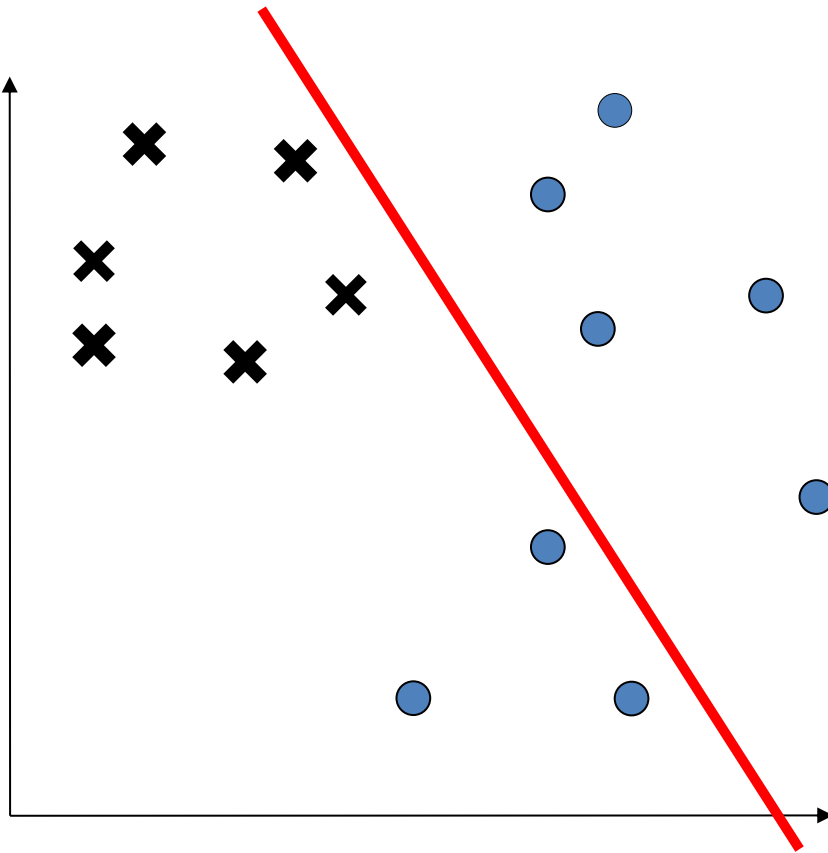
$$\text{if } \left(\sum_{i=1}^M x_i w_i \right) > t \quad \text{then } \textit{output} = 1, \text{ else } \textit{output} = 0$$





Is this a good decision boundary?

$$\text{if } \left(\sum_{i=1}^M x_i w_i \right) > t \quad \text{then } \textit{output} = 1, \text{ else } \textit{output} = 0$$

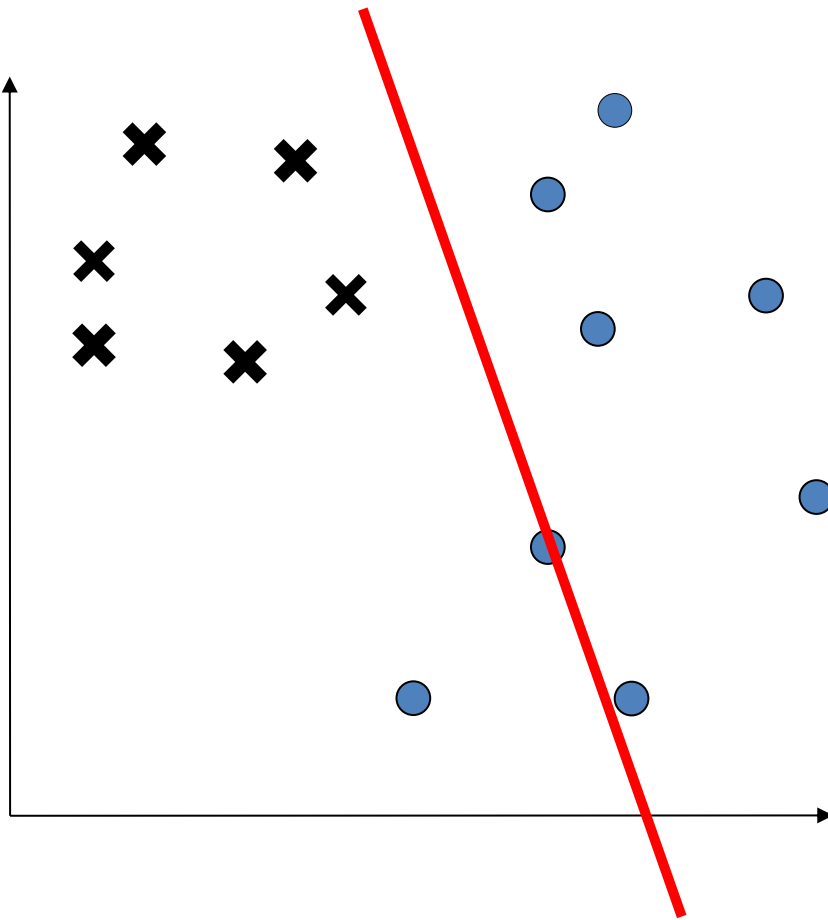


$$w_1 = 1.0$$

$$w_2 = 0.2$$

$$t = 0.05$$

$$\text{if } \left(\sum_{i=1}^M x_i w_i \right) > t \quad \text{then } \textit{output} = 1, \text{ else } \textit{output} = 0$$

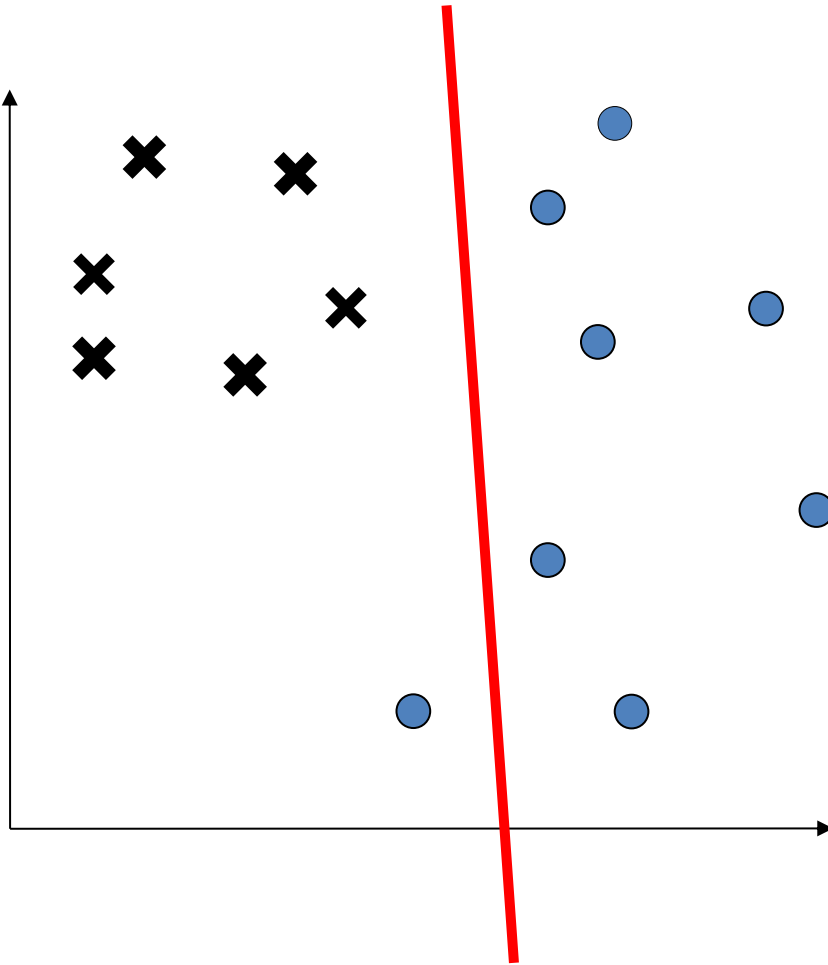


$$w_1 = 2.1$$

$$w_2 = 0.2$$

$$t = 0.05$$

$$\text{if } \left(\sum_{i=1}^M x_i w_i \right) > t \quad \text{then } \textit{output} = 1, \text{ else } \textit{output} = 0$$

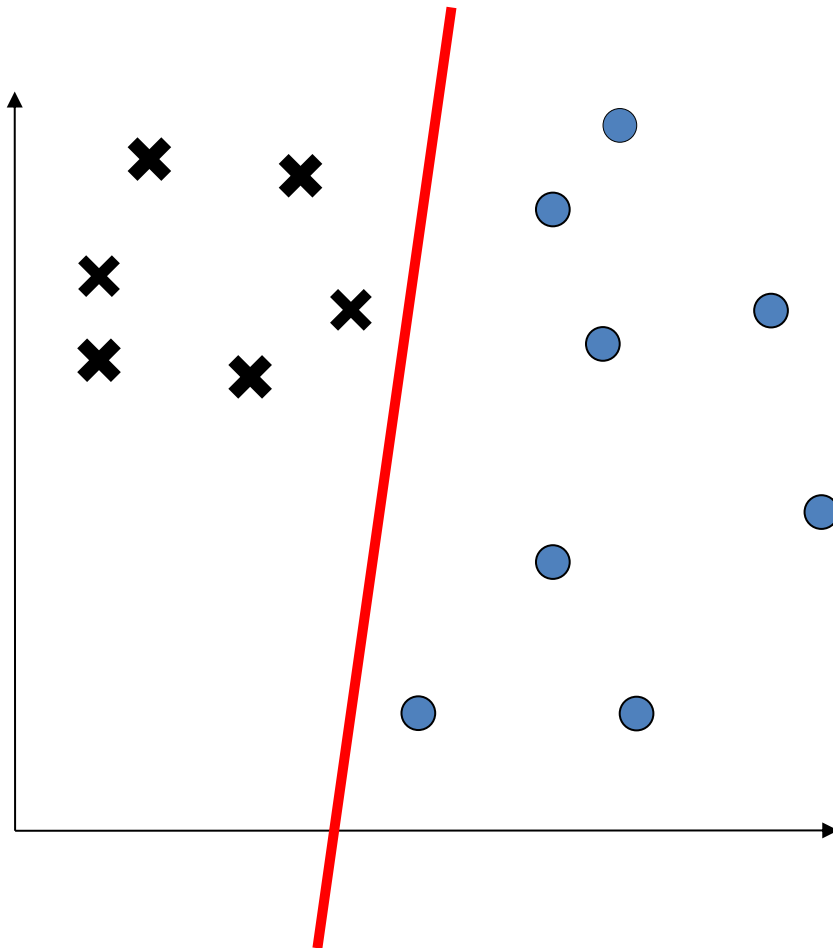


$$w_1 = 1.9$$

$$w_2 = 0.02$$

$$t = 0.05$$

$$\text{if } \left(\sum_{i=1}^M x_i w_i \right) > t \quad \text{then } \textit{output} = 1, \text{ else } \textit{output} = 0$$



$$w_1 = 0.8$$

$$w_2 = -0.03$$

$$t = 0.05$$

Changing the weights/threshold makes the decision boundary move.

Pointless / impossible to do it by hand – only ok for simple 2-D case.

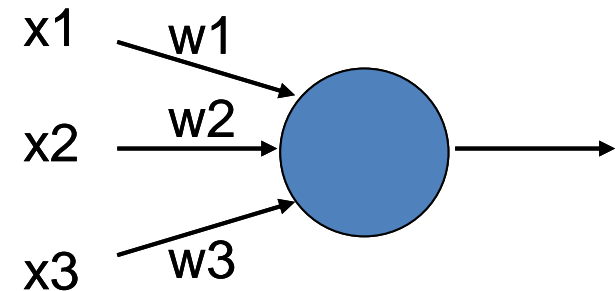
We need an algorithm....

$$x = [1.0, 0.5, 2.0]$$

$$w = [0.2, 0.5, 0.5]$$

$$t = 1.0$$

$$a = \sum_{i=1}^M x_i w_i$$



Q1. What is the activation, a , of the neuron?

$$a = \sum_{i=1}^M x_i w_i = (1.0 \times 0.2) + (0.5 \times 0.5) + (2.0 \times 0.5) = 1.45$$

Q2. Does the neuron fire?

if (activation > threshold) output=1 else output=0

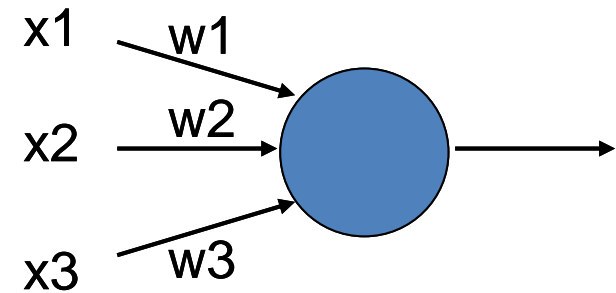
.... So yes, it fires.

$$x = [1.0, 0.5, 2.0]$$

$$w = [0.2, 0.5, 0.5]$$

$$t = 1.0$$

$$a = \sum_{i=1}^M x_i w_i$$



Q3. What if we set threshold at 0.5 and weight #3 to zero?

$$a = \sum_{i=1}^M x_i w_i = (1.0 \times 0.2) + (0.5 \times 0.5) + (2.0 \times 0.0) = 0.45$$

if (activation > threshold) output=1 else output=0

.... So no, it does not fire..

The Perceptron

Model

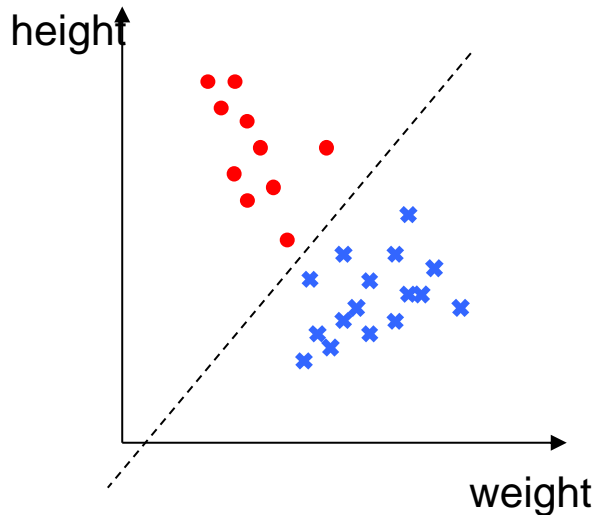
$$\text{if } \sum_{i=1}^d w_i x_i > t \text{ then } \hat{y} = 1 \text{ else } \hat{y} = 0 \quad \left\{ \begin{array}{l} \text{"player"} = 1 \\ \text{"dancer"} = 0 \end{array} \right.$$

Error function

Number of mistakes (a.k.a. classification error)

Learning algo.

??? need to optimise the w and t values...



Perceptron Learning Rule

$$\text{new weight} = \text{old weight} + 0.1 \times (\text{trueLabel} - \text{output}) \times \text{input}$$

update

What weight updates do these cases produce?

if... (**target** = 0, **output** = 0) then **update** = ?

if... (**target** = 0, **output** = 1) then **update** = ?

if... (**target** = 1, **output** = 0) then **update** = ?

if... (**target** = 1, **output** = 1) then **update** = ?

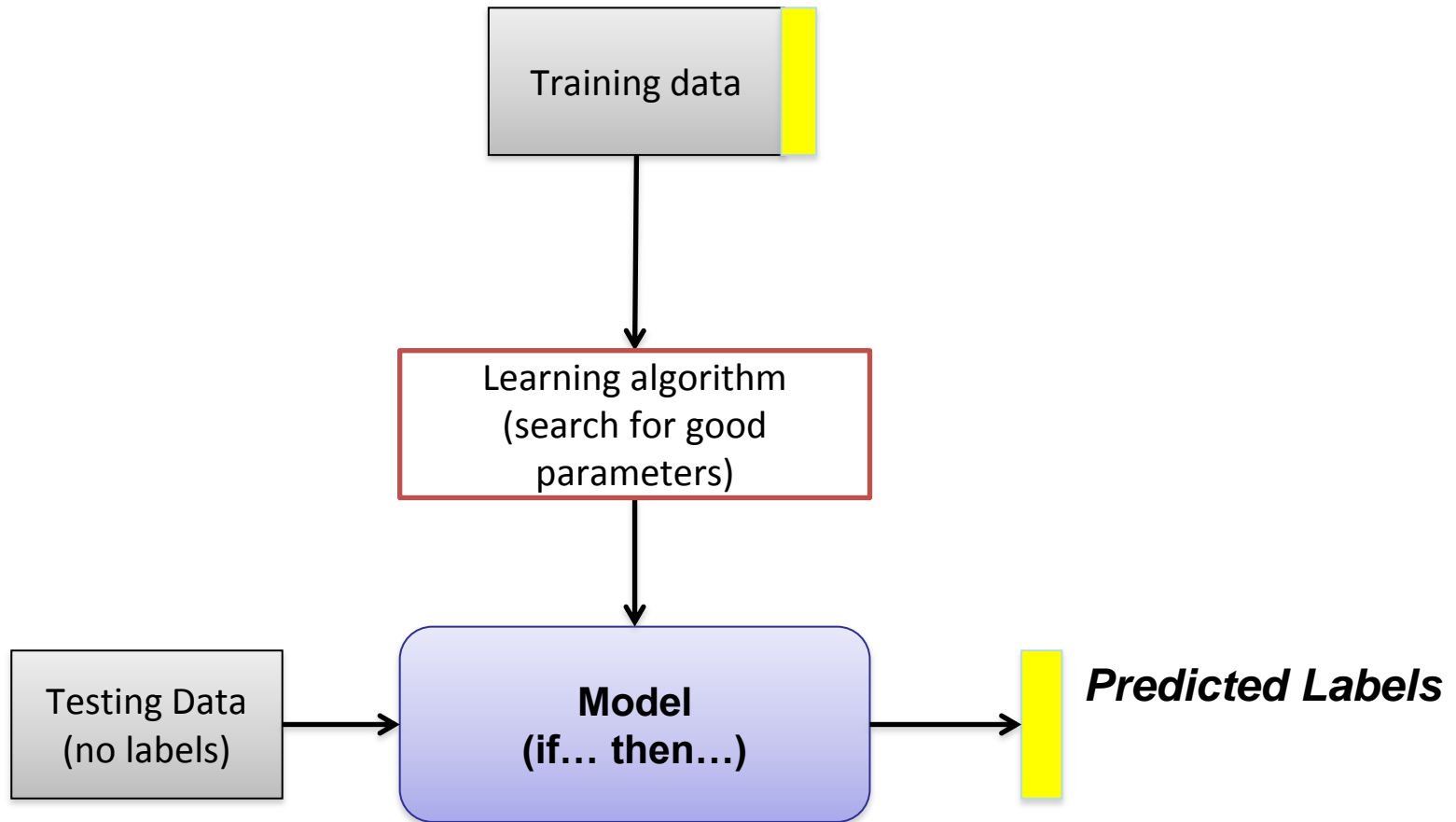
Learning algorithm for the Perceptron

```
initialise weights to random numbers in range -1 to +1
for n = 1 to NUM_ITERATIONS
  for each training example (x,y)
    calculate activation
    for each weight
      update weight by learning rule
    end
  end
end
end
```

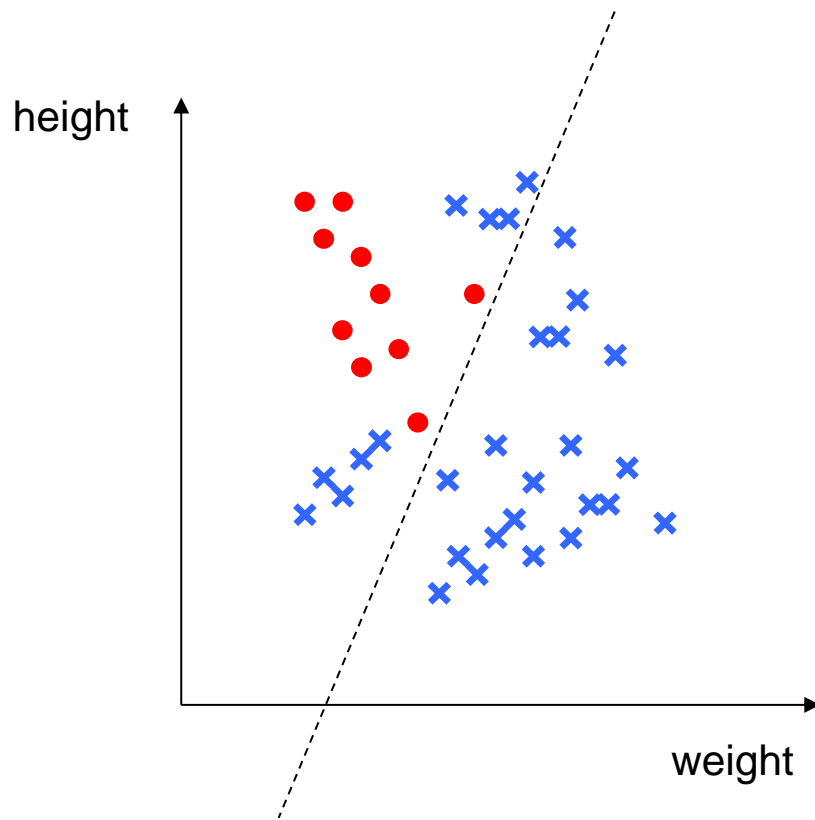
Perceptron convergence theorem:

If the data is linearly separable, then application of the Perceptron learning rule will find a separating decision boundary, within a finite number of iterations

Supervised Learning Pipeline for Perceptron



New data.... “non-linearly separable”



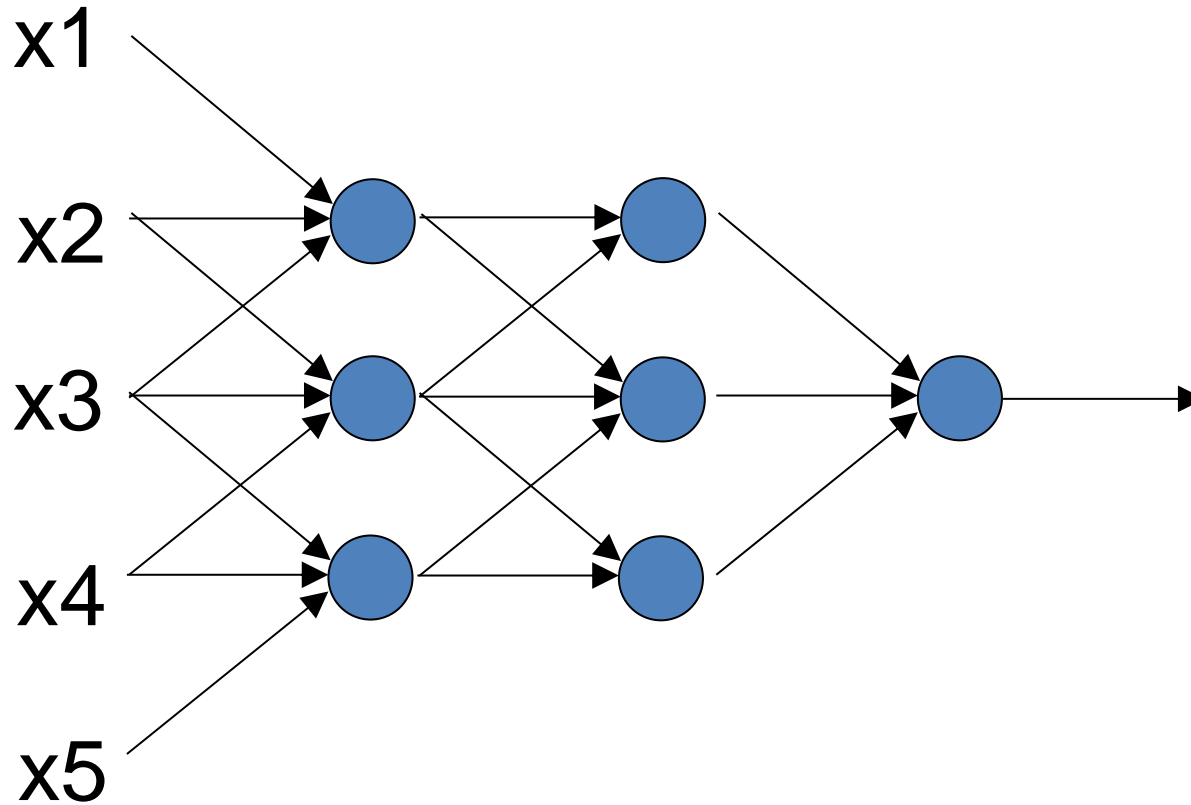
Our model does not match
the problem!

(AGAIN!)

if $\sum_{i=1}^d w_i x_i > t$ then "player" else "dancer"

Many mistakes!

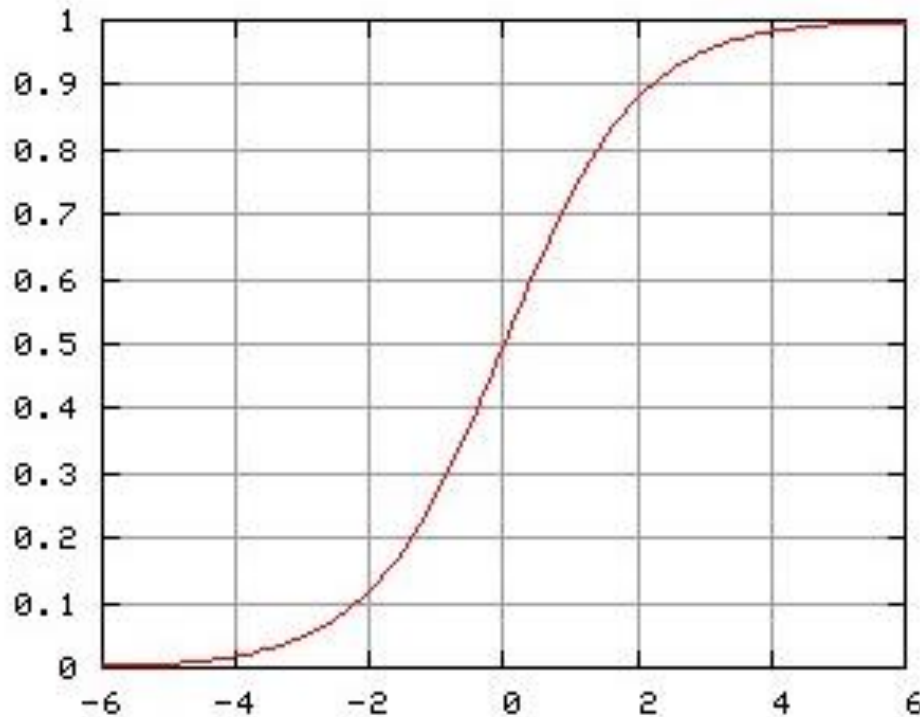
One Approach: Multilayer Perceptron



Another Approach: Sigmoid activation – no more thresholds needed ☺

~~if $\sum_{i=1}^d w_i x_i > t$ then $\hat{y} = 1$ else $\hat{y} = 0$~~

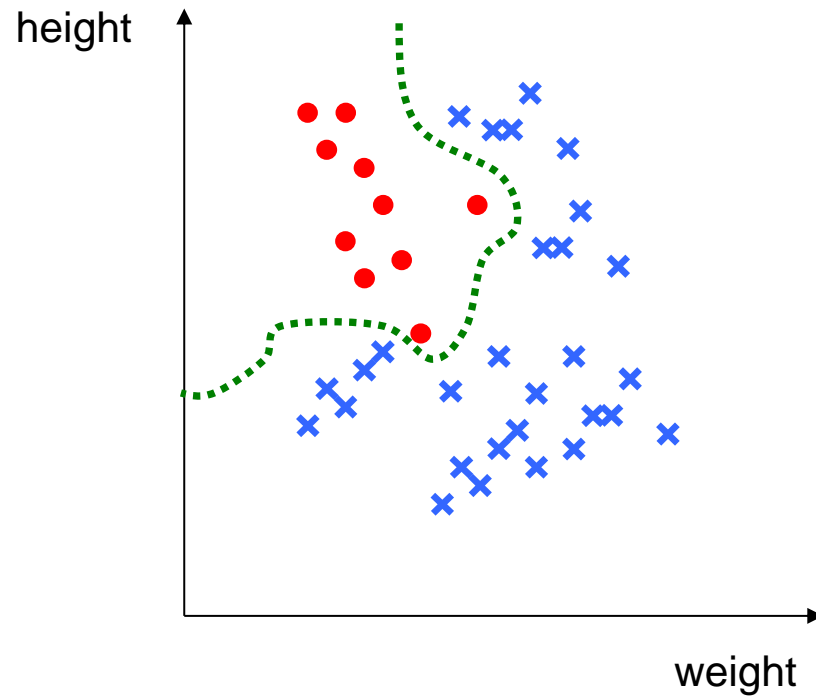
$$a = \frac{1}{1 + \exp\left(-\sum_{i=1}^d w_i x_i\right)}$$



activation level

$$\sum_{i=1}^d w_i x_i$$

MLP decision boundary – nonlinear problems, solved!



Neural Networks - summary

Perceptrons are a (simple) emulation of a neuron.

Layering perceptrons gives you... a multilayer perceptron.

An MLP is one type of neural network – there are others.

An MLP with sigmoid activation functions can solve highly nonlinear problems.

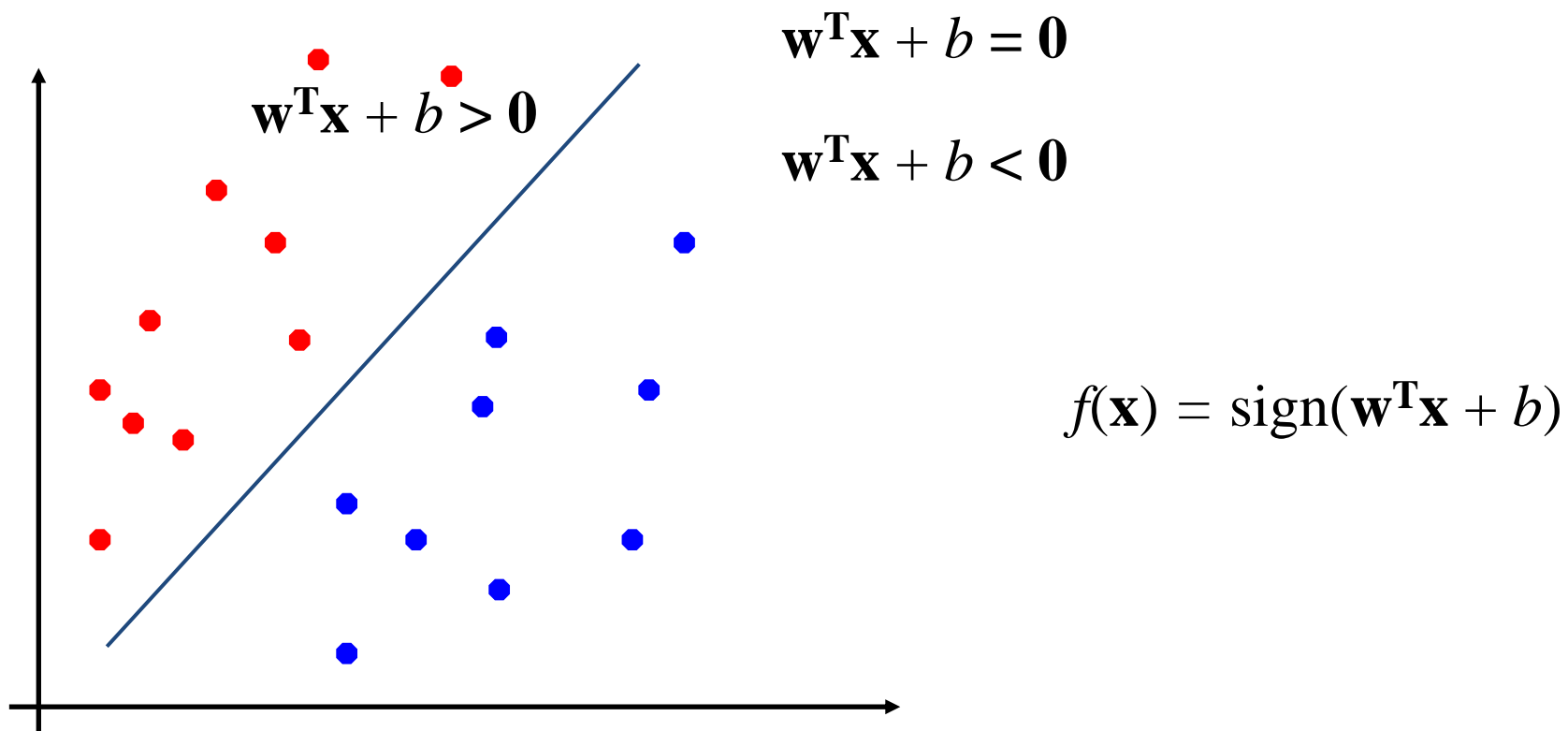
Downside – we cannot use the simple perceptron learning algorithm.

Instead we have the “backpropagation” algorithm.

We will cover this later.

Perceptron Revisited: Linear Separators

- Binary classification can be viewed as the task of separating classes in feature space:



SEE SLIDES FOR SVM