Lecture 16: The Perceptron

Artificial Intelligence
CS-GY-6613

Julian Togelius
julian.togelius@nyu.edu

Types of learning

- **Supervised learning**
  Learning to predict or classify labels based on labeled input data

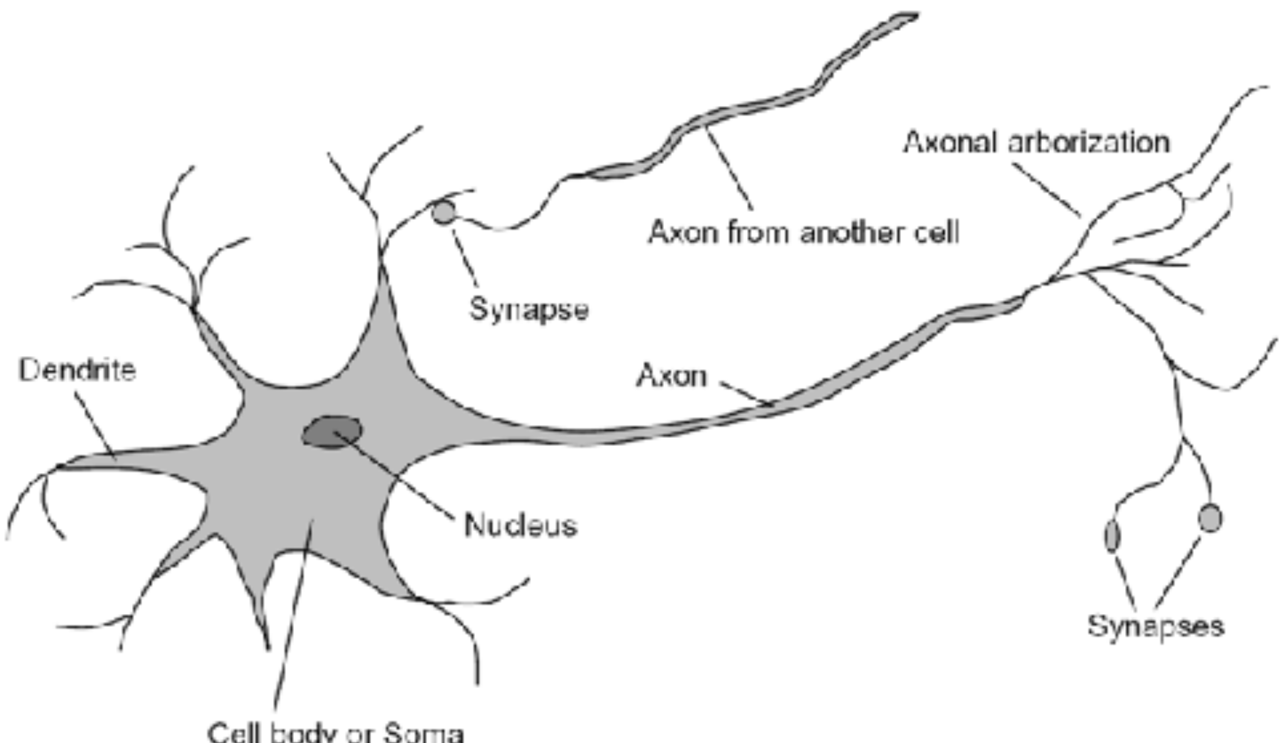
- **Unsupervised learning**
  Finding patterns in unlabeled data

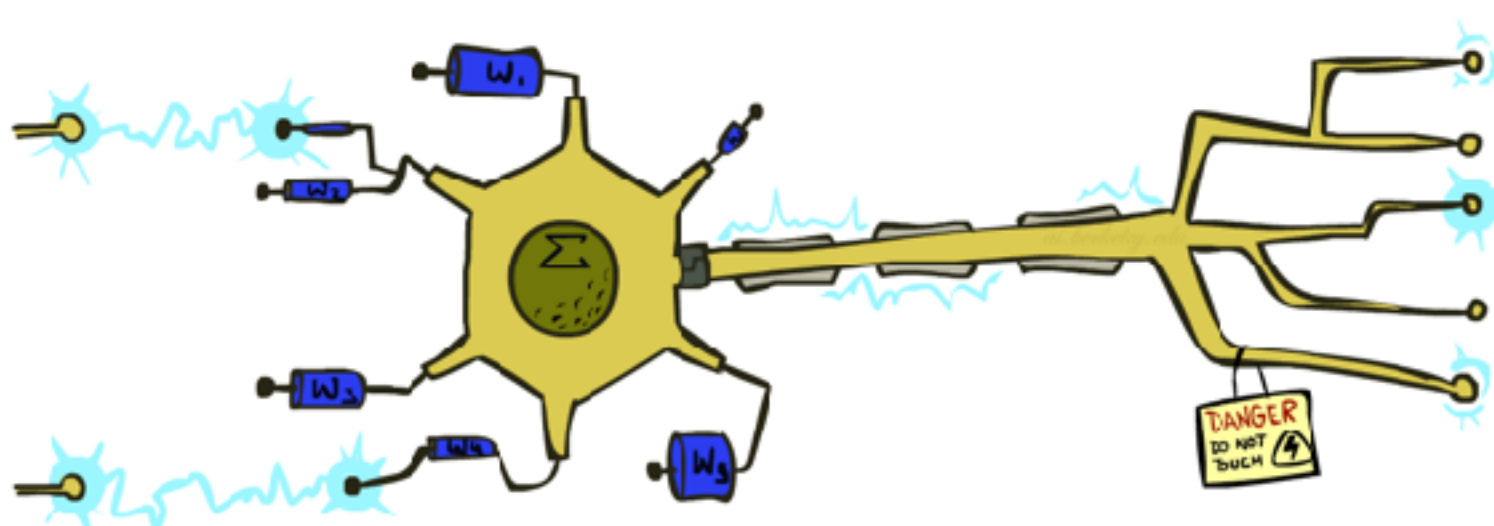
- **Reinforcement learning**
  Learning well-performing behavior from state observations and rewards

Perceptrons

- Early attempt at “neural networks” - copying neurophysiology to produce a learning machine
- Very simple and fast learning algorithm for linearly separable problems
- The basis for many more advanced neural network variants, such as Multilayer Perceptrons (and, by extension, deep networks)

Very loose biological analogy



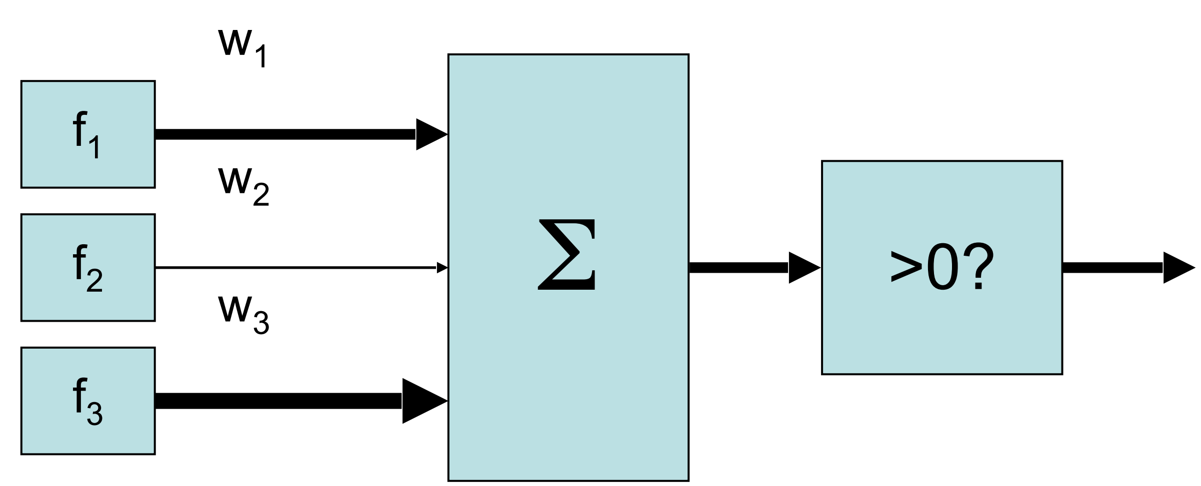


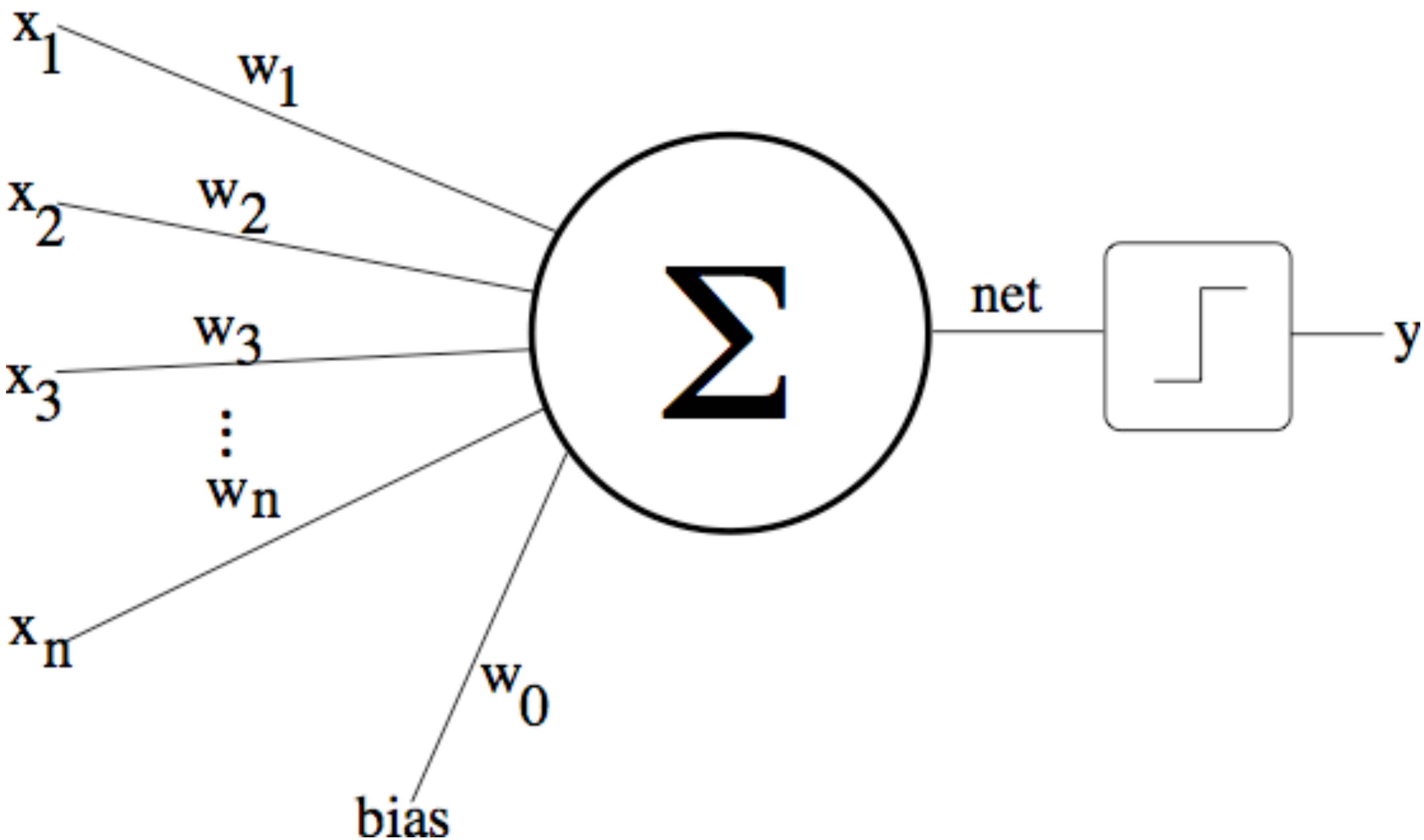
Perceptrons are linear classifiers

- Inputs are feature values
- Each feature has a weight
- Sum is the activation

$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

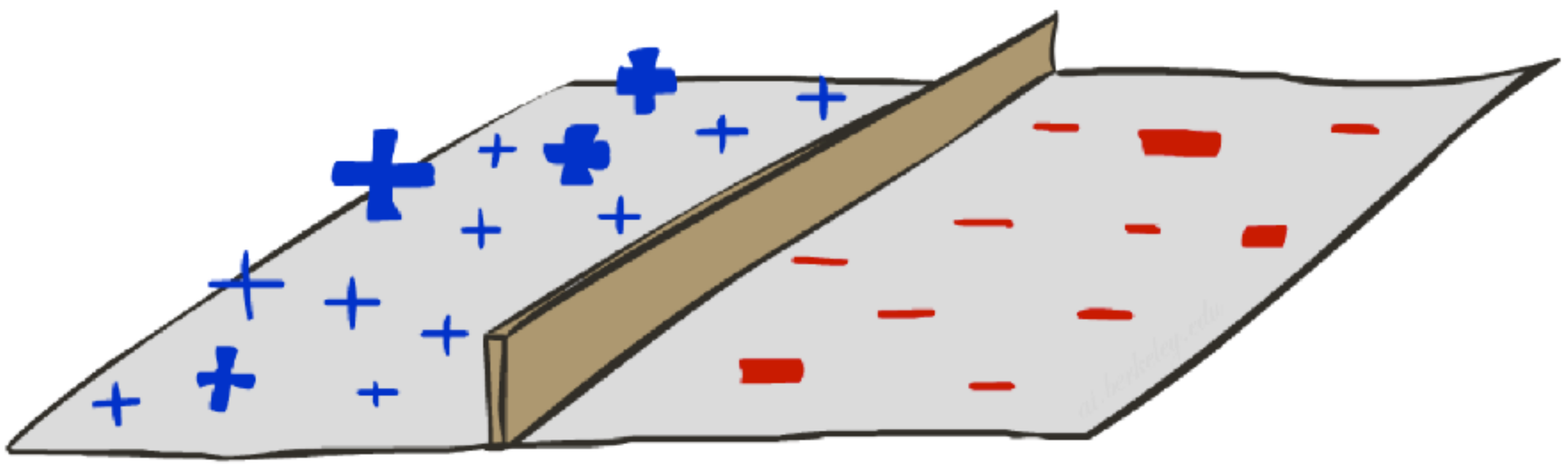
- If the activation is:
  - Positive, output +1
  - Negative, output -1





$$f(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0 & \text{else} \end{cases}$$

Decision surface

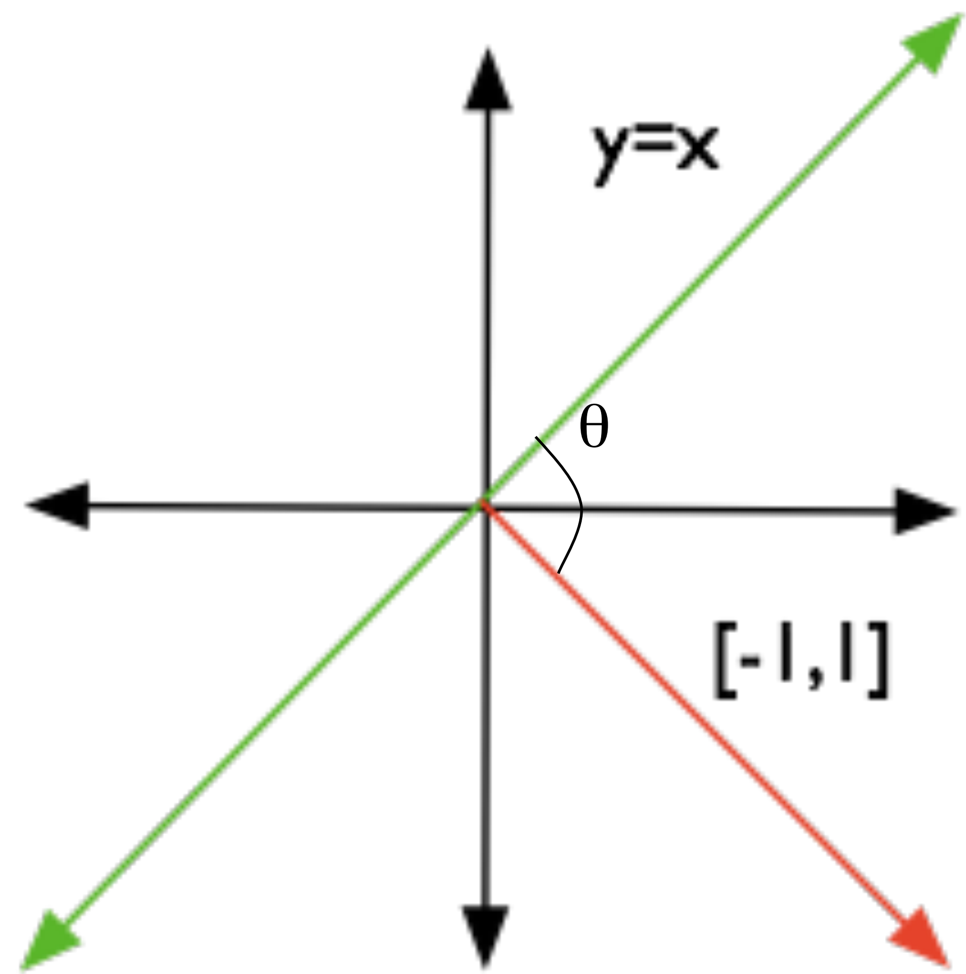


Decision surface

$$y = x$$
$$0 = x - y$$
$$0 = [1, -1]\left[\begin{array}{c} x \\ y \end{array}\right]$$

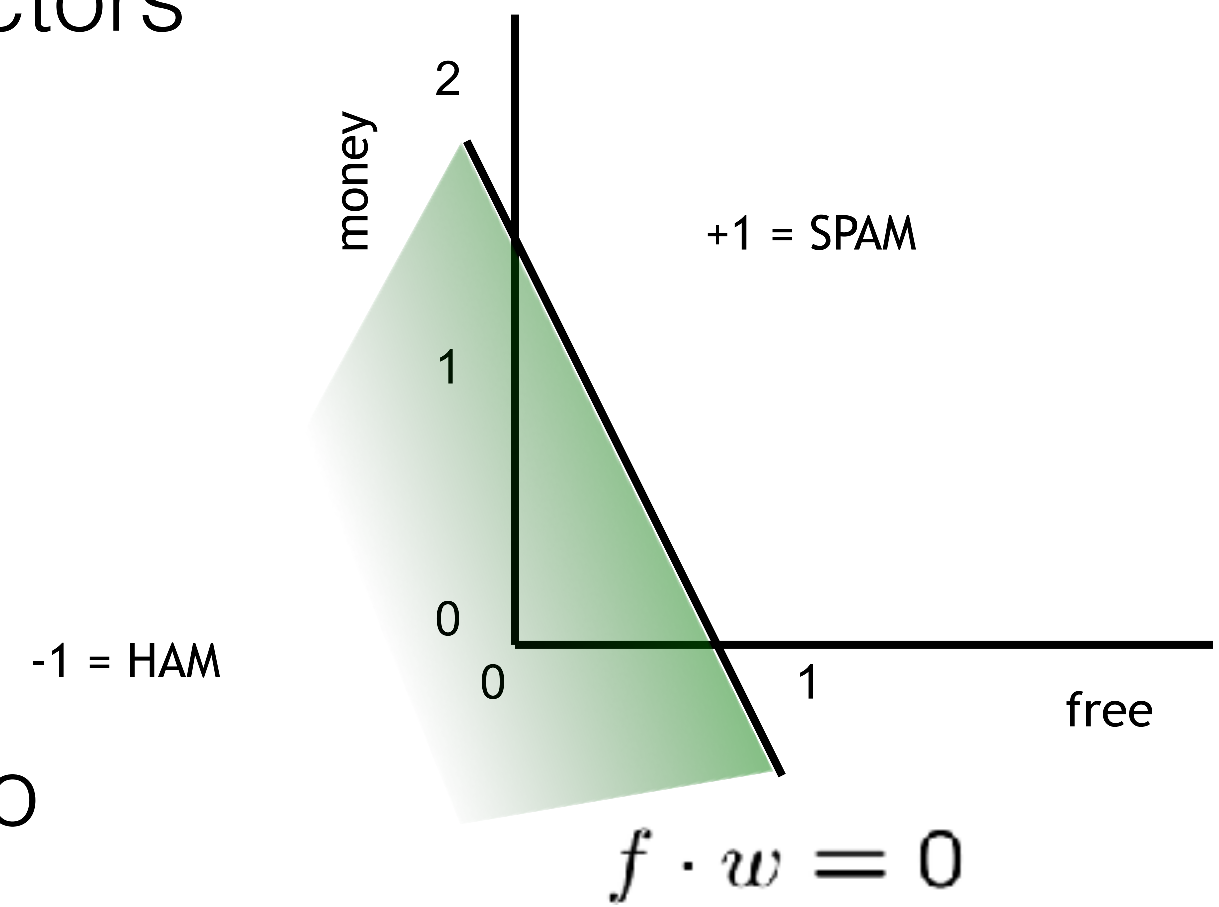
In general a **hyperplane** is defined by

$$0 = \vec{w} \cdot \vec{x}$$



Model usage

- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to Y=+1
  - Other corresponds to Y=-1



Model training

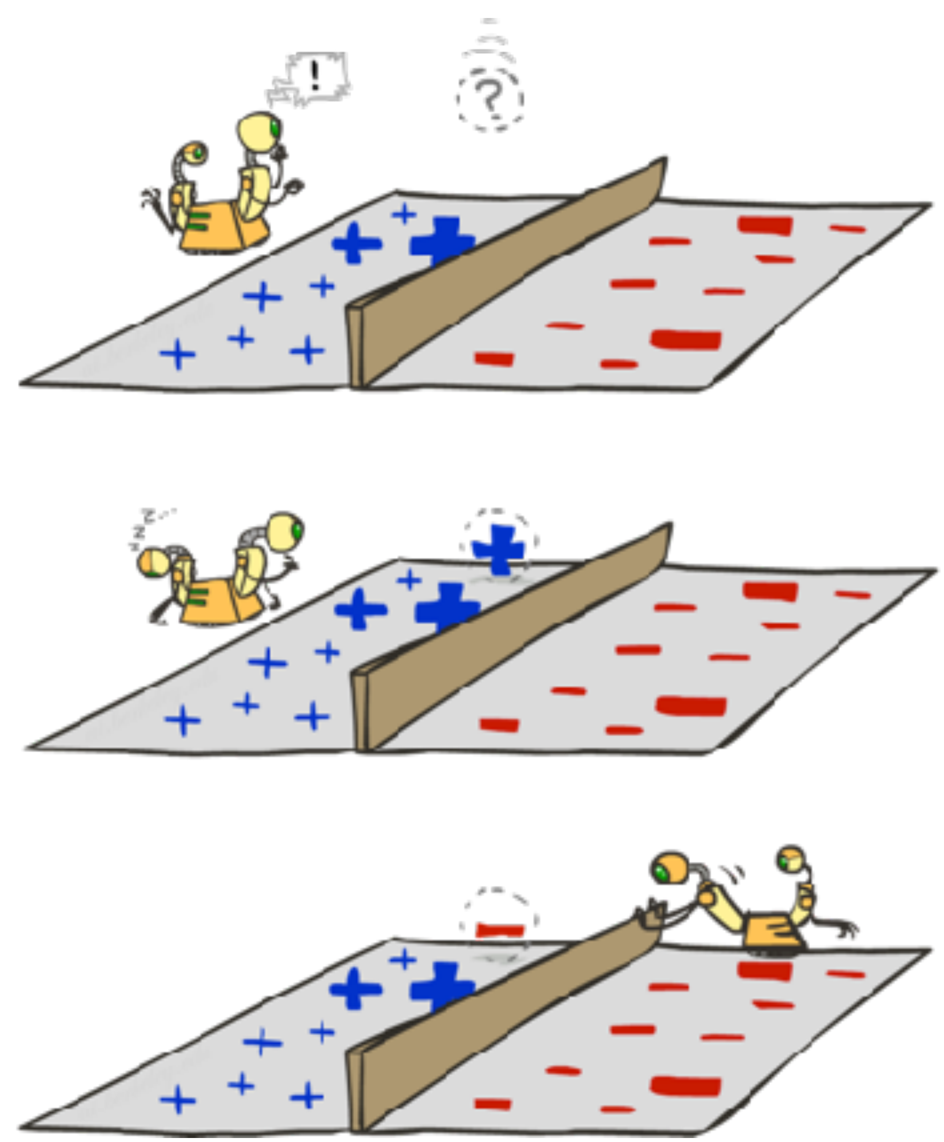
Algorithm

- Start with random weights. For each training instance:

- Classify with current weights

- If correct (i.e., y=y*), no change!

- If wrong: adjust the weight vector



Algorithm

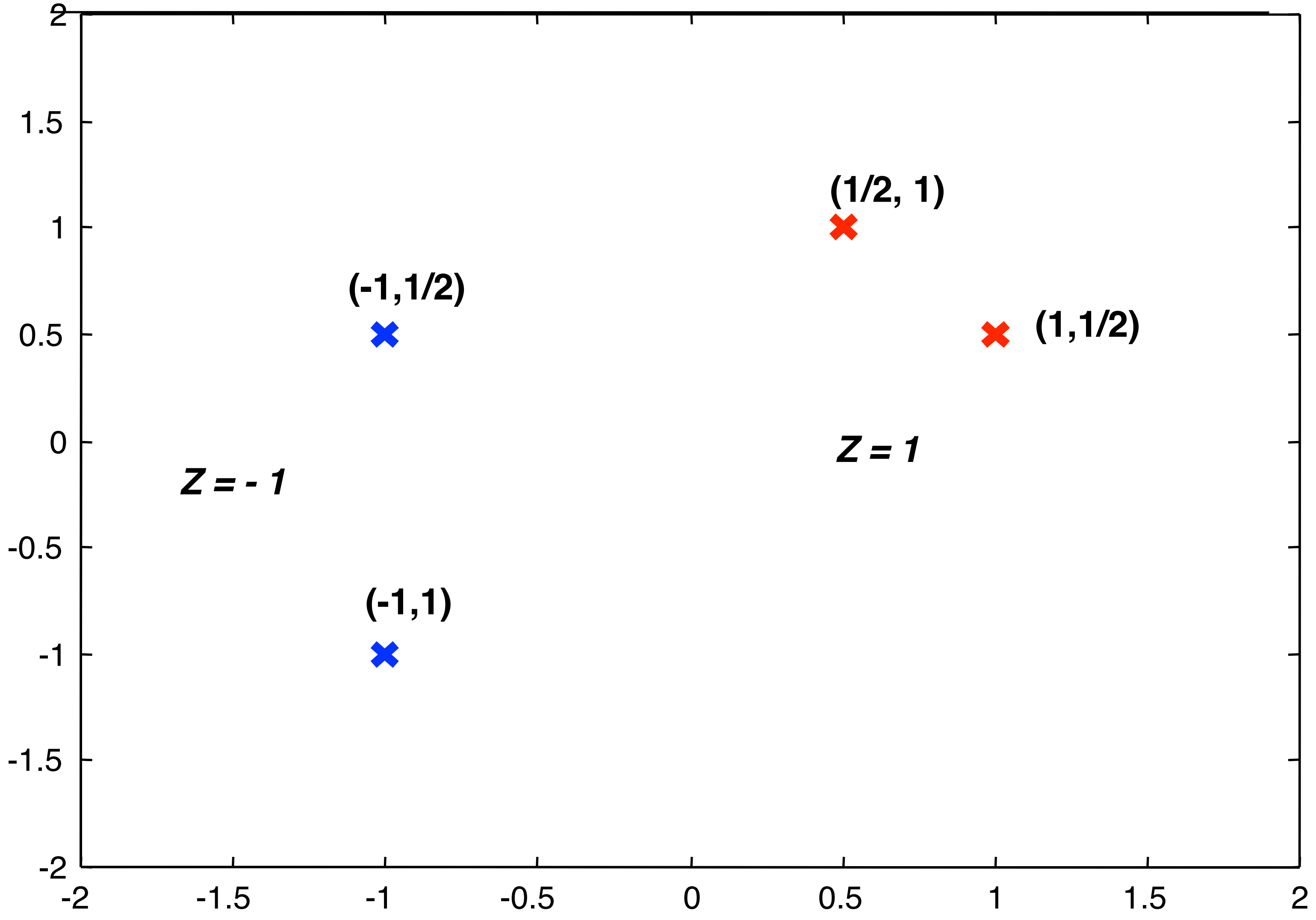
Input : list of n training examples $(x_0, d_0)...(x_n, d_n)$
where $\forall i : d_i \in \{+1, -1\}$
Output : classifying hyperplane w

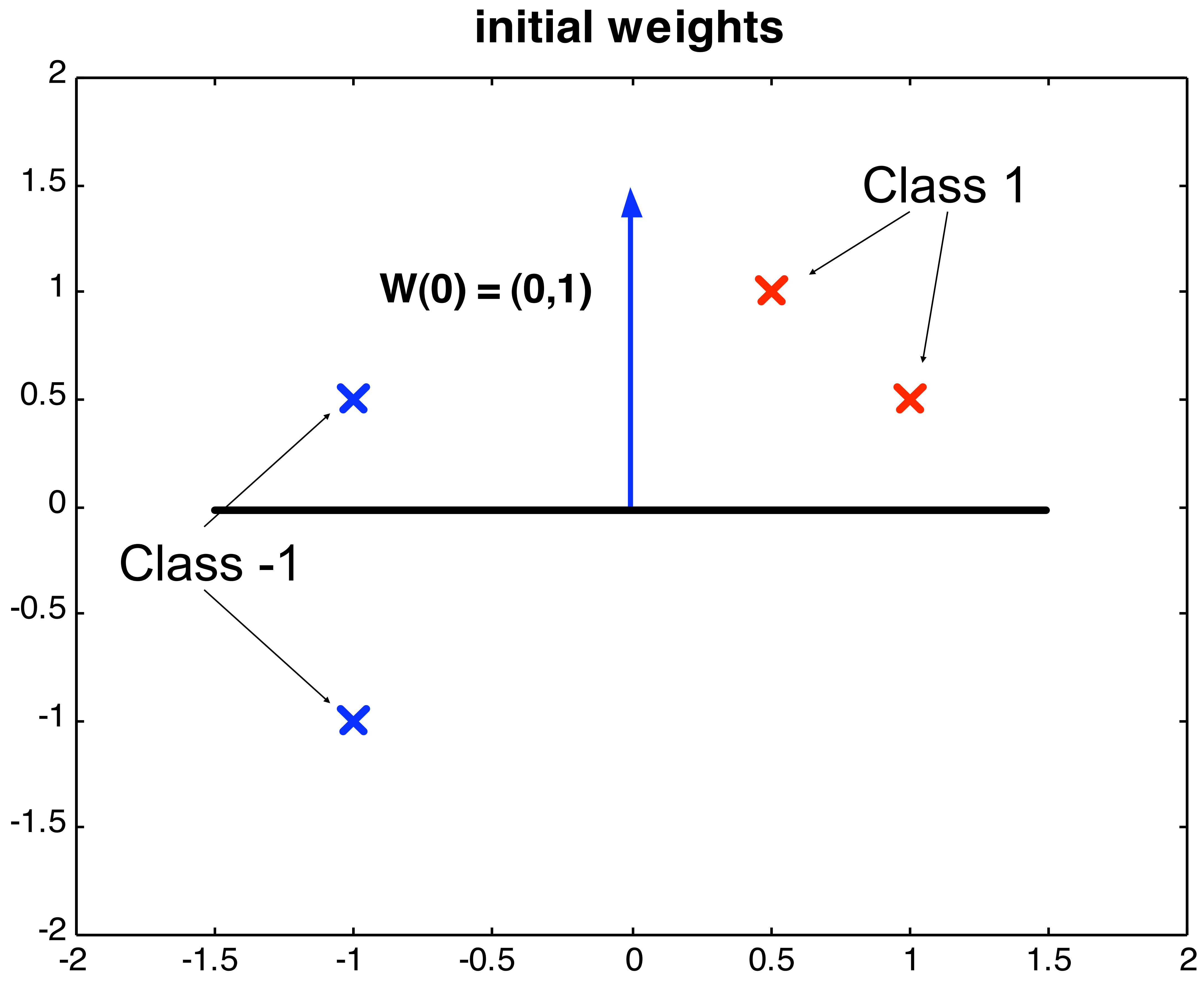
Algorithm :
Randomly initialize w;
While makes errors on training set **do**
    for $(x_i d_i)$ **do**
        let $y_i = sign(w \cdot x_i)$;
        if $y_i \neq d_i$ **then**
            $w \leftarrow w + \eta d_i x_i$;
        end
    end
end

x and w are vectors;
i is the instance index

A simple example

4 linearly separable points





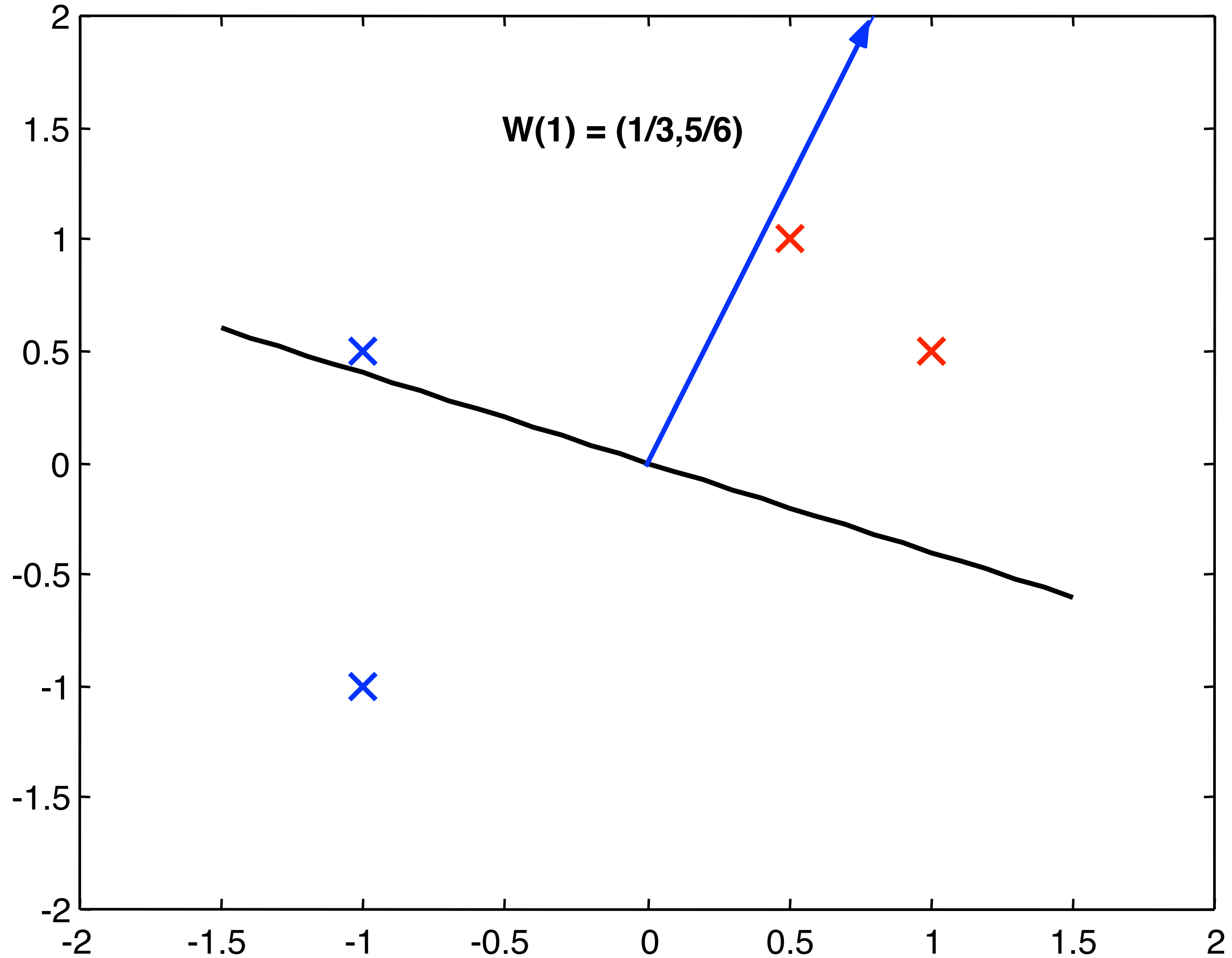
initial weights
W(0) = (0,1)
Class 1
Class -1
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

Updating Weights

Upper left point (-1,1/2) is wrongly classified

$$
\begin{aligned}
&x = (-1, 1/2) \\
&d = -1 \\
&\eta = 1/3, w(0) = (0,1) \\
&w(1) \leftarrow w(0) + \eta d x \\
&w(1) = (0,1) + 1/3 * (-1) * (-1, 1/2) \\
&\qquad\;\; = (0,1) + 1/3 * (1, -1/2) \\
&\qquad\;\; = (1/3, 5/6)
\end{aligned}
$$

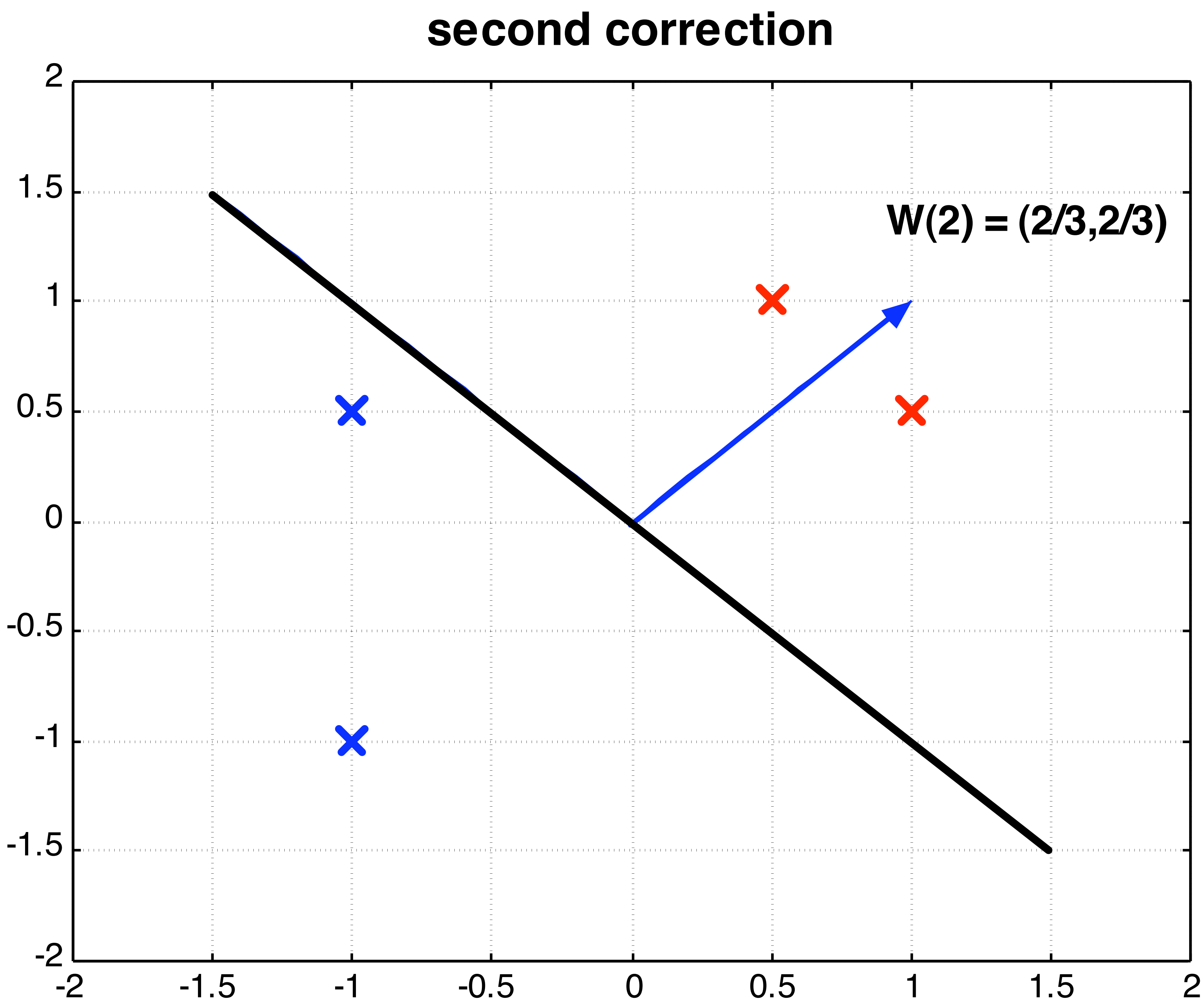
first correction



Updating Weights, Ctd

Upper left point is still wrongly classified

$$
\begin{aligned}
&x = (-1, 1/2) \\
&d = -1 \\
&w(2) \leftarrow w(1) + \eta d x \\
&w(2) = (1/3, 5/6) + 1/3 * (-1) * (-1, 1/2) \\
&\qquad = (1/3, 5/6) + 1/3 * (1, -1/2) \\
&\qquad = (2/3, 2/3)
\end{aligned}
$$


second correction
W(2) = (2/3,2/3)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

If we have multiple classes

- A weight vector for each class:

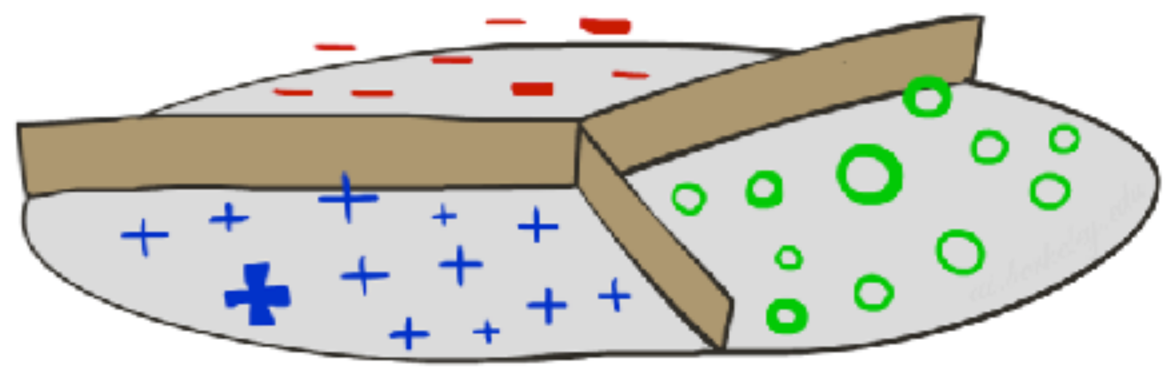
$$w_y$$

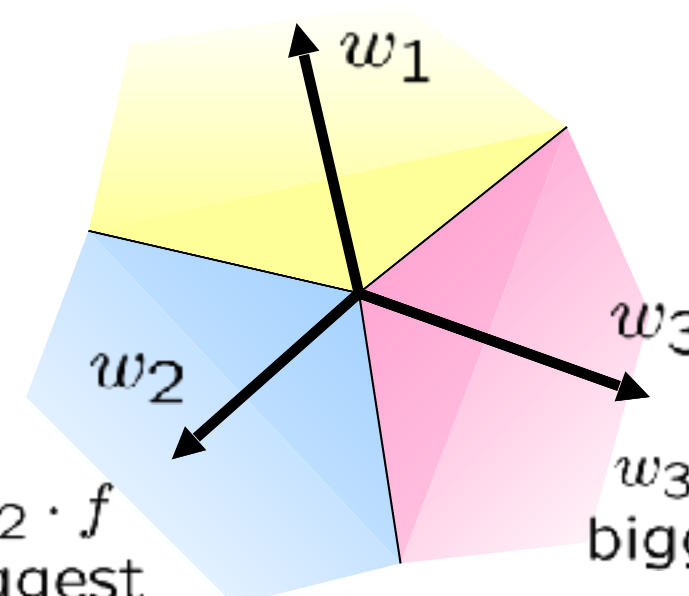
- Score (activation) of a class y:

$$w_y \cdot f(x)$$

- Prediction highest score wins

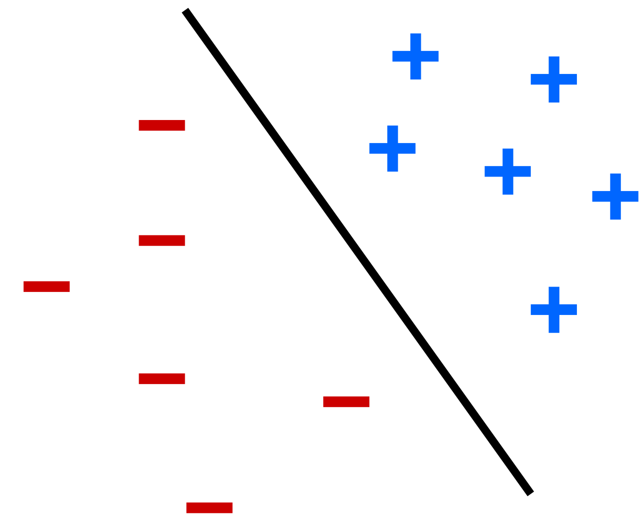
$$y = \underset{y}{\text{arg max}} \; w_y \cdot f(x)$$



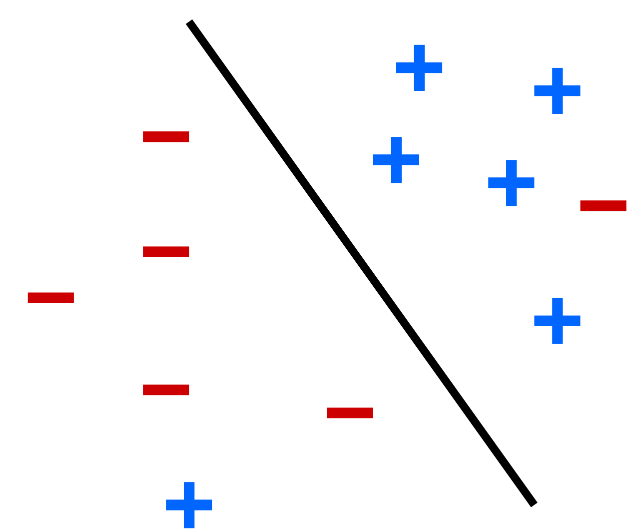


Properties of Perceptrons

- Separability: true if some parameters get the training set perfectly correct

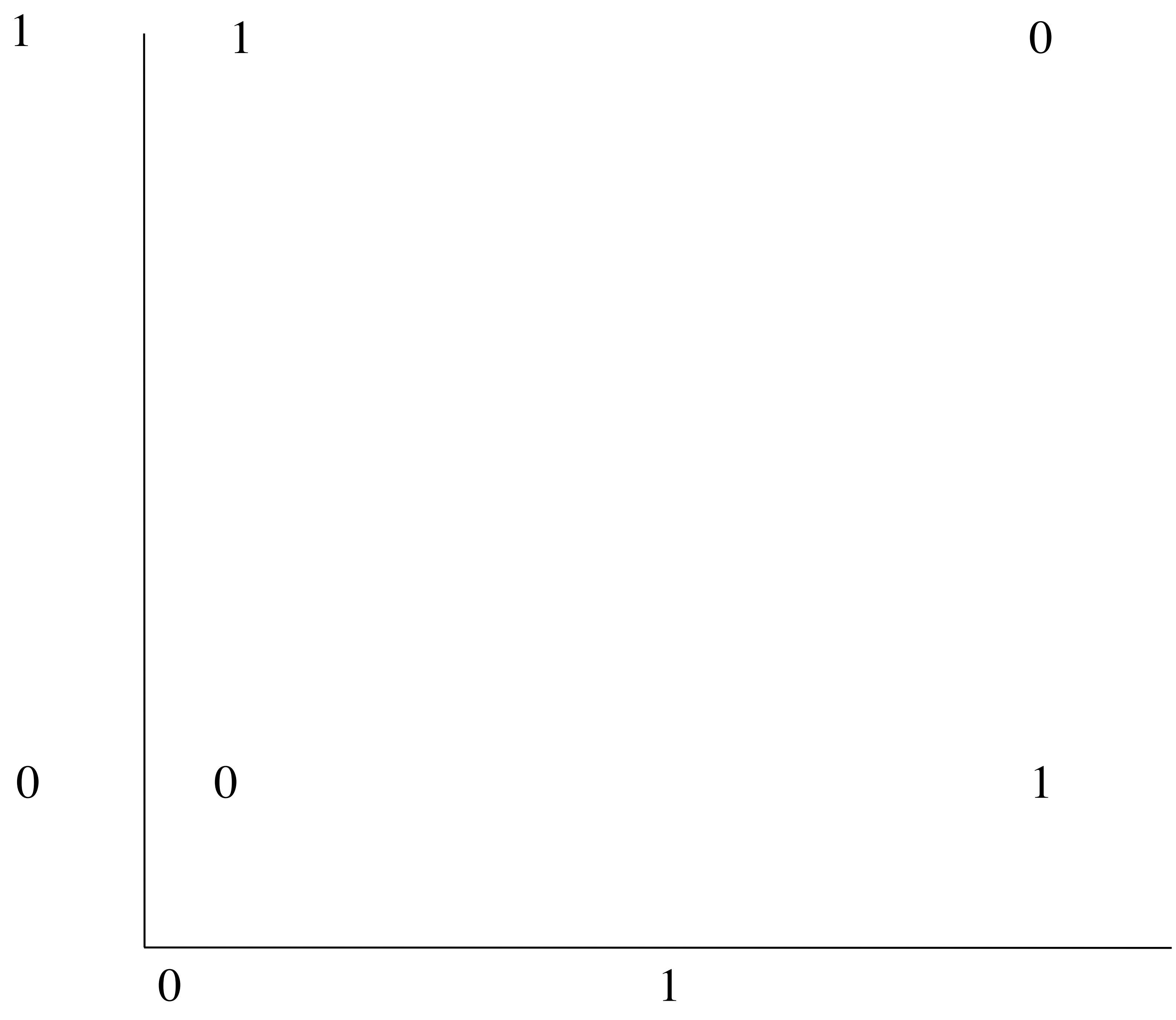


- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the margin or degree of separability

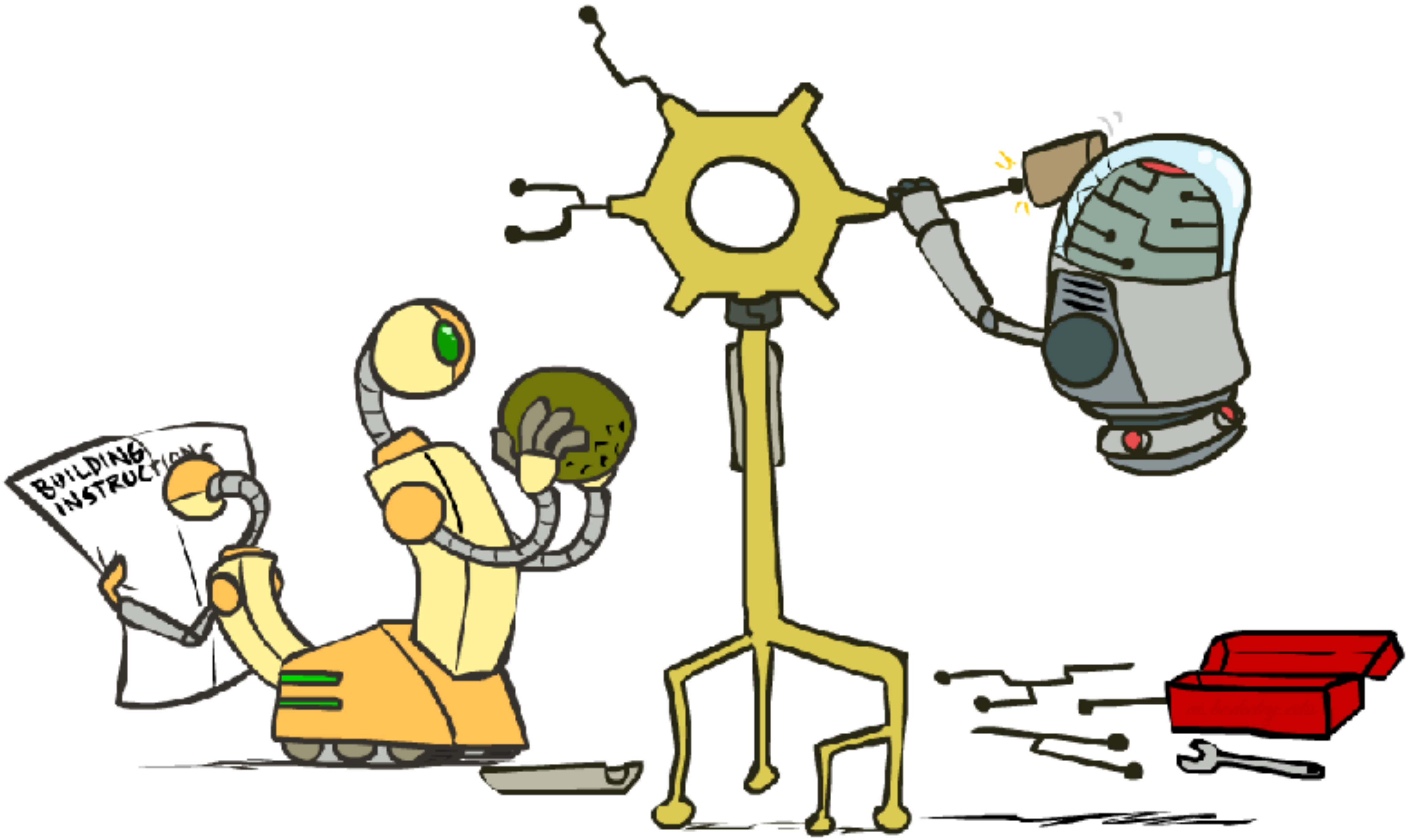


$$\text{mistakes} < \frac{k}{\delta^2}$$

Problem: learn this!

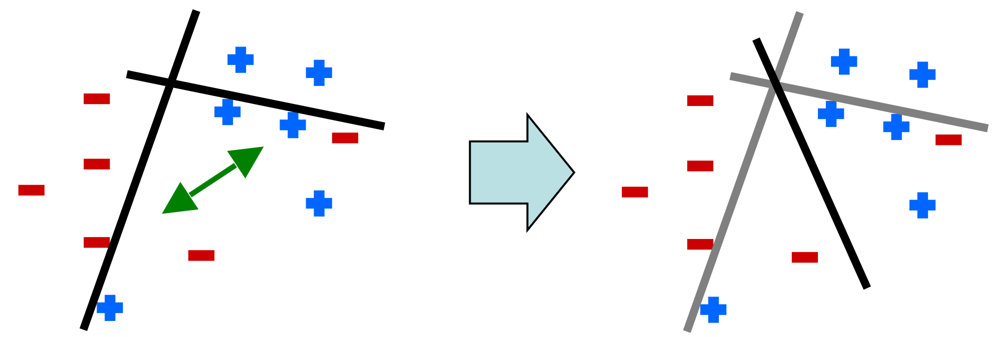


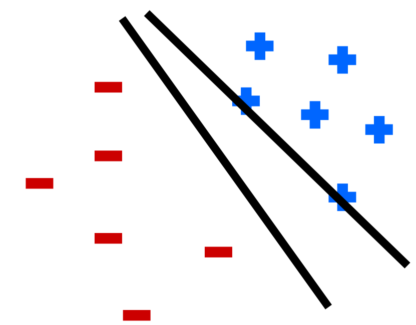
Improving the perceptron

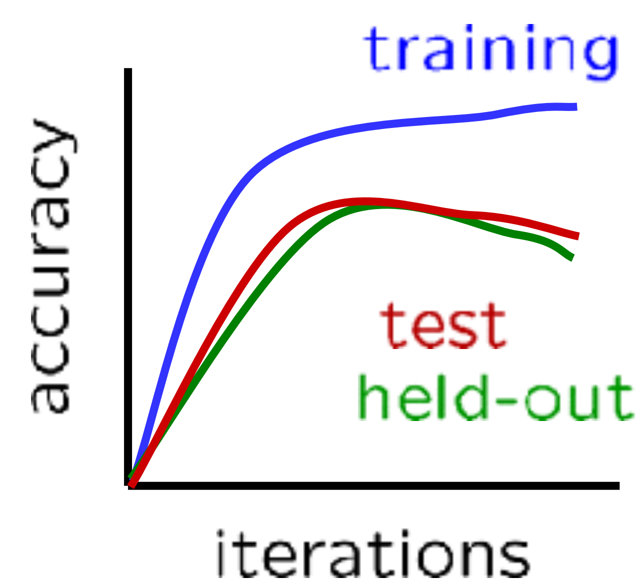


Perceptron problems

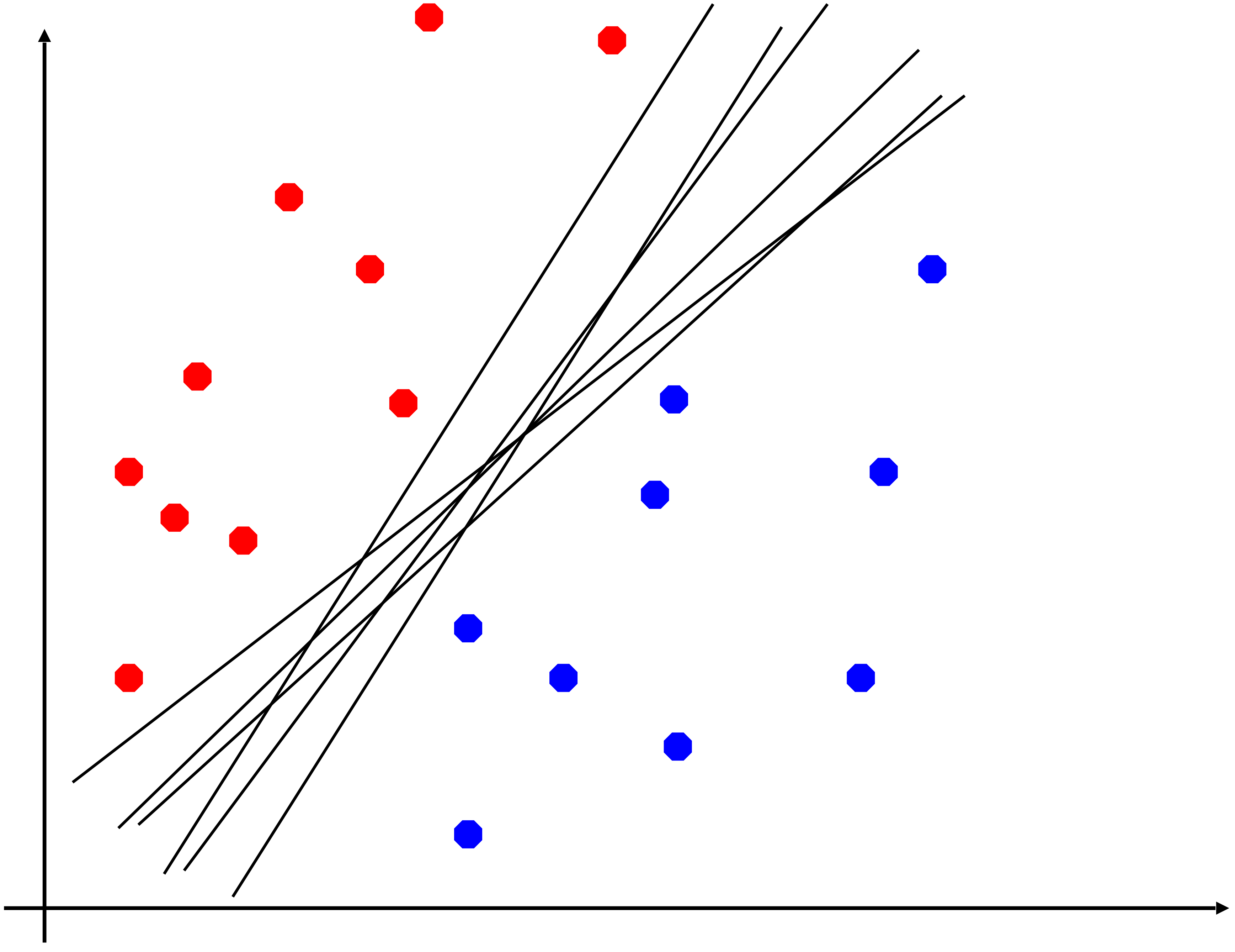
- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)
- Mediocre generalization: finds a "barely" separating solution
- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining is a kind of overfitting





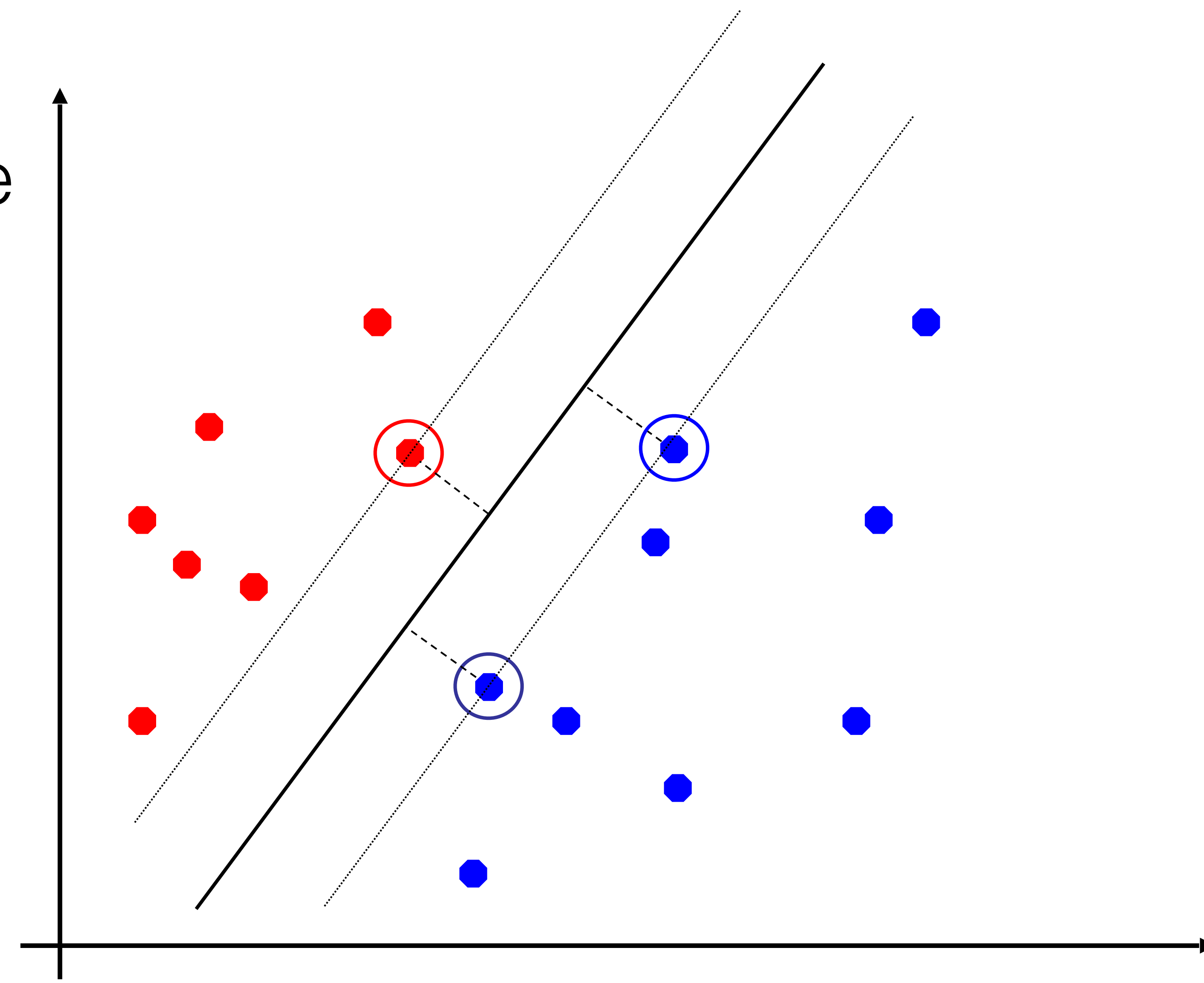


Which of these separators is optimal?

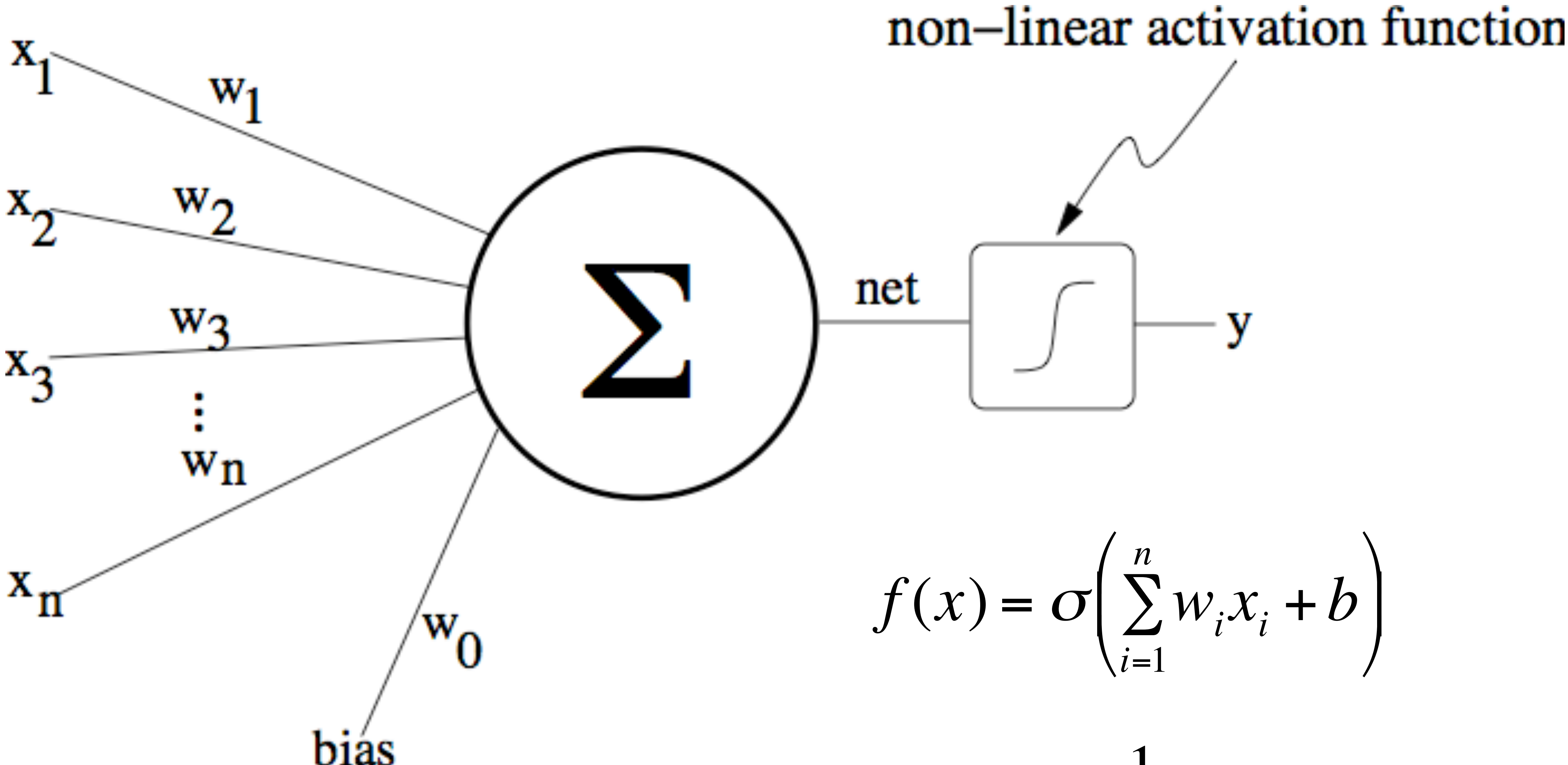


Support vector machines

- Maximizing the margin: good according to intuition, theory, practice
- Only support vectors matter; other training examples are ignorable
- Support vector machines (SVMs) find the separator with max margin



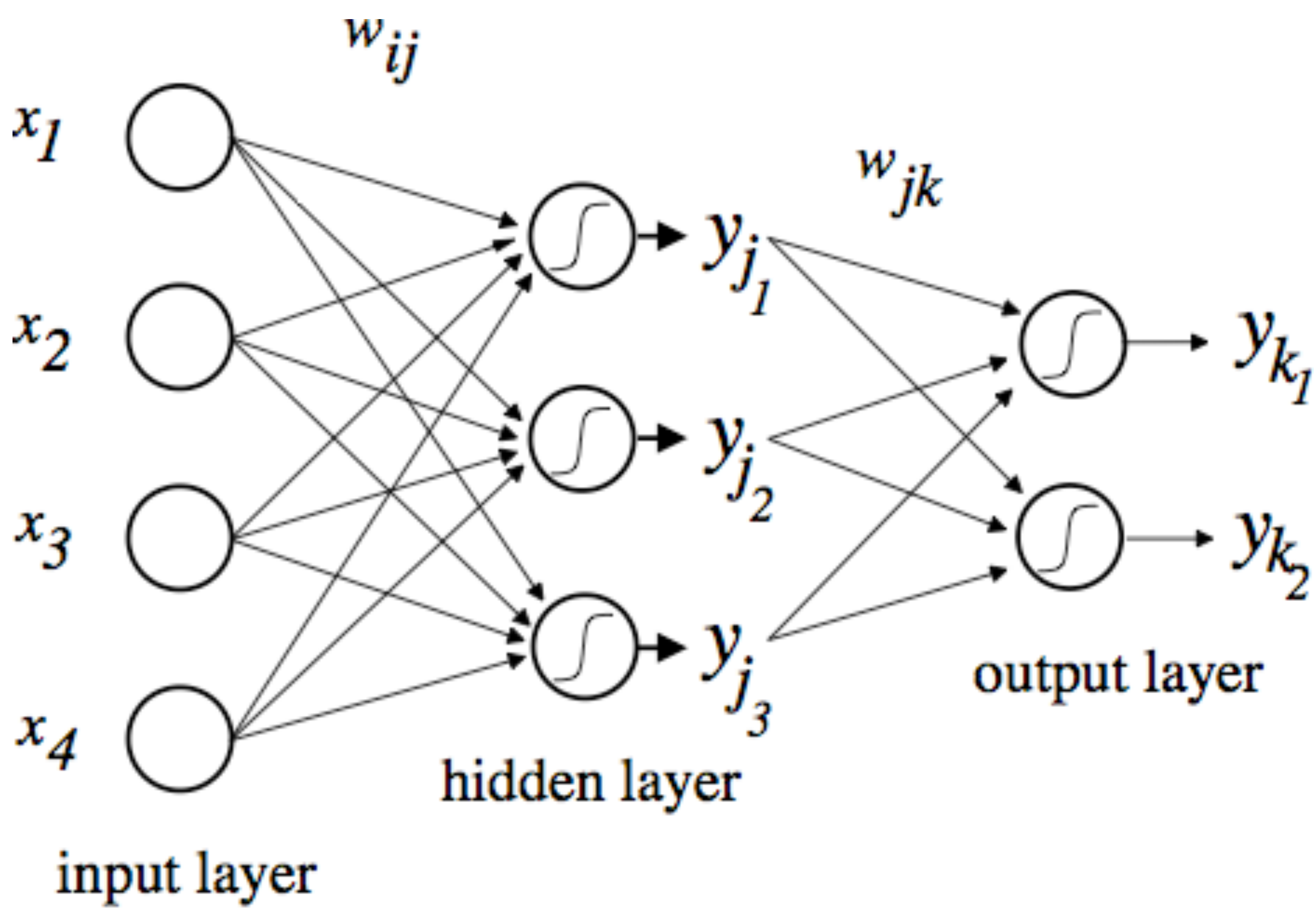
Non-Linear Neuron



$$f(x) = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

$$\sigma(net) = \frac{1}{1 + e^{-net}}$$

Multi-layer Perceptron (MLP)



Backpropagation

- Forward Pass: present training input pattern to network and activate network to produce output (can also do in batch: present all patterns in succession)
- Backward Pass: calculate error gradient and update weights starting at output layer and then going back