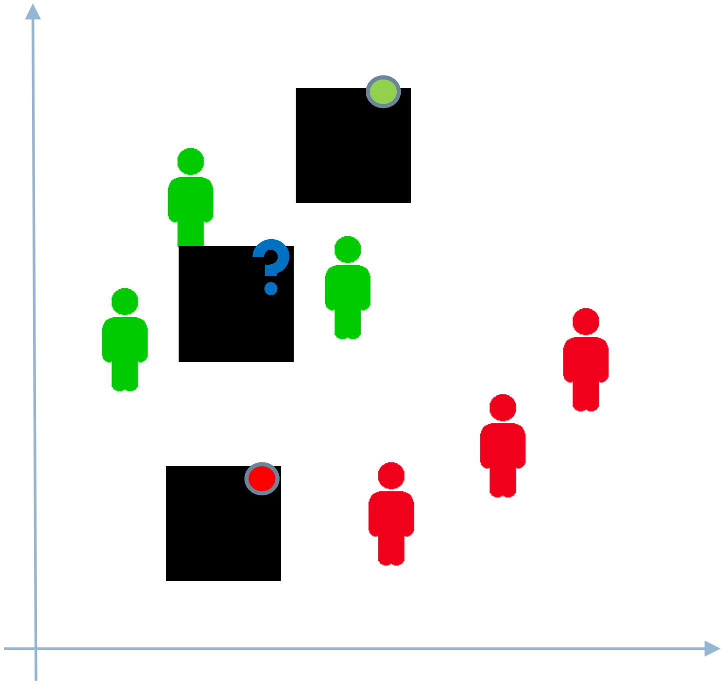
Advanced classification methods

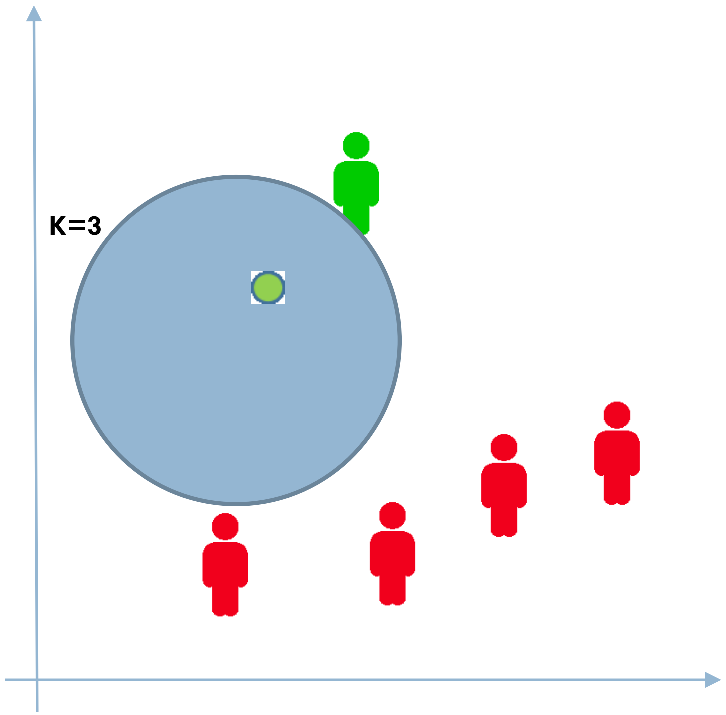
The most stupid classifier

- Rote learner
  - To classify object X, check if there is a labelled example in the training set identical to X
  - Yes → X has the same label
  - No → I don't know



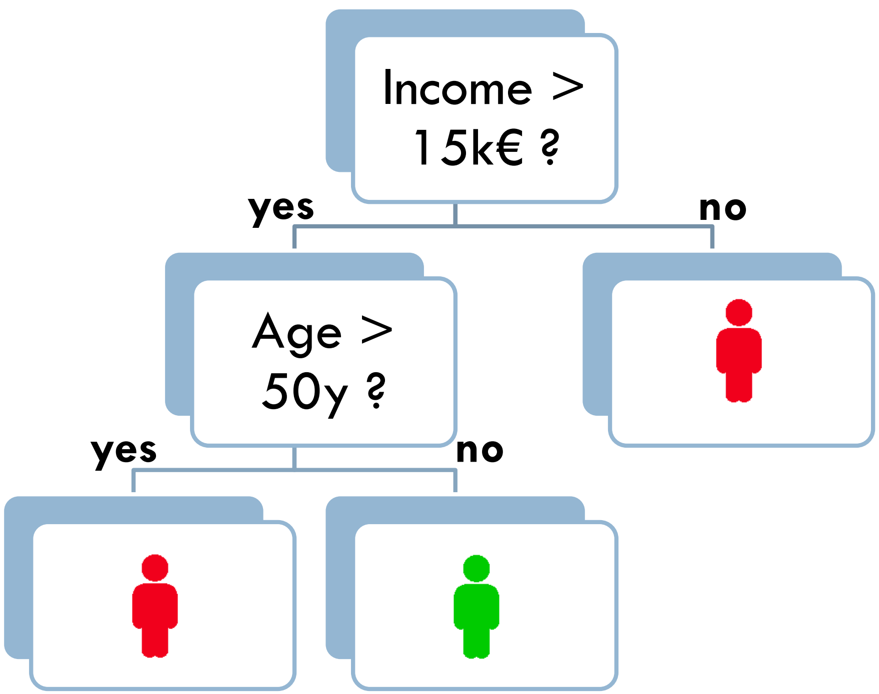
Classify by similarity

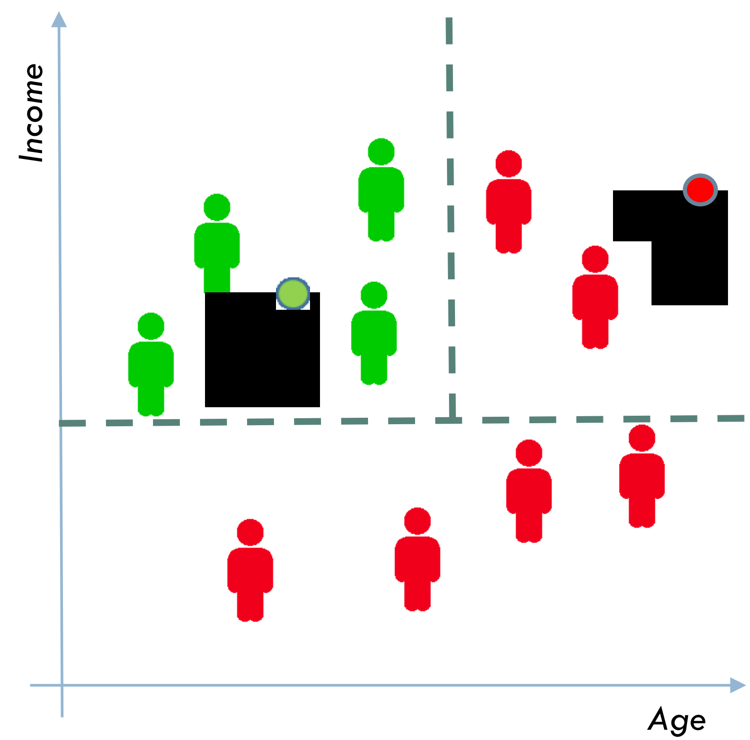
- K-Nearest Neighbors
  - Decide label based on K most similar examples



Build a model

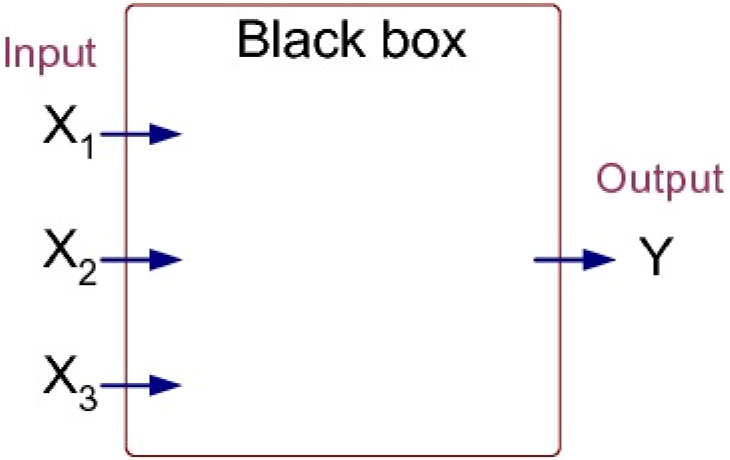
- Decision Trees
  - Cut space by lines orthogonal to the axes





Artificial Neural Networks (ANN)

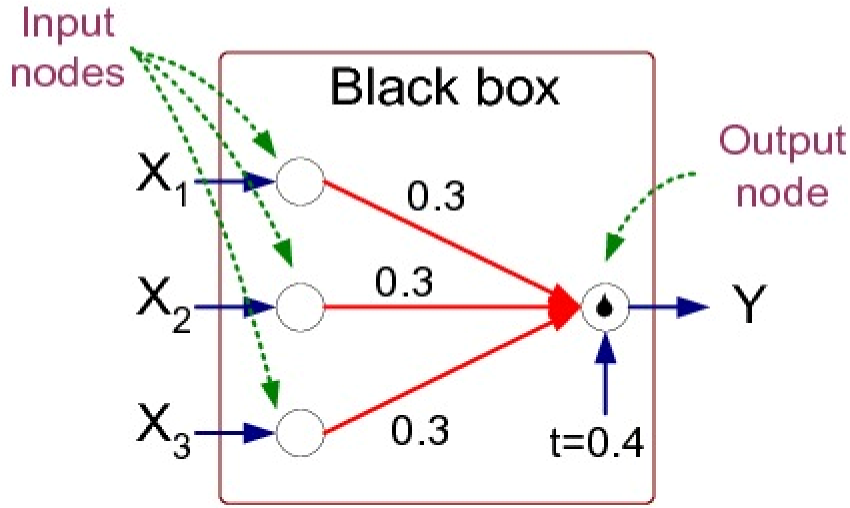
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Output Y is 1 if at least two of the three inputs are equal to 1.

Artificial Neural Networks (ANN)

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

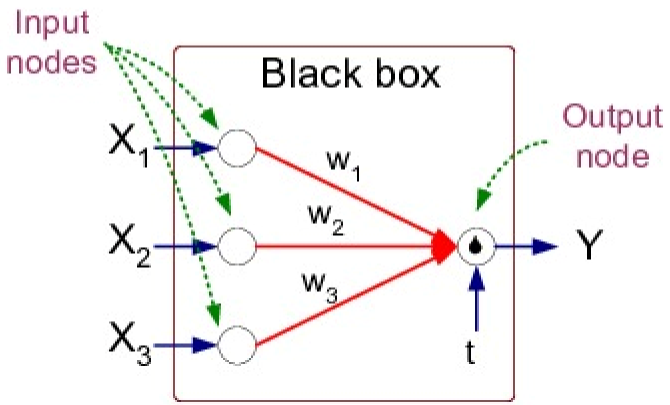
$$\text{where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Artificial Neural Networks (ANN)

- Model is an assembly of inter-connected nodes and weighted links

- Output node sums up each of its input value according to the weights of its links

- Compare output node against some threshold t



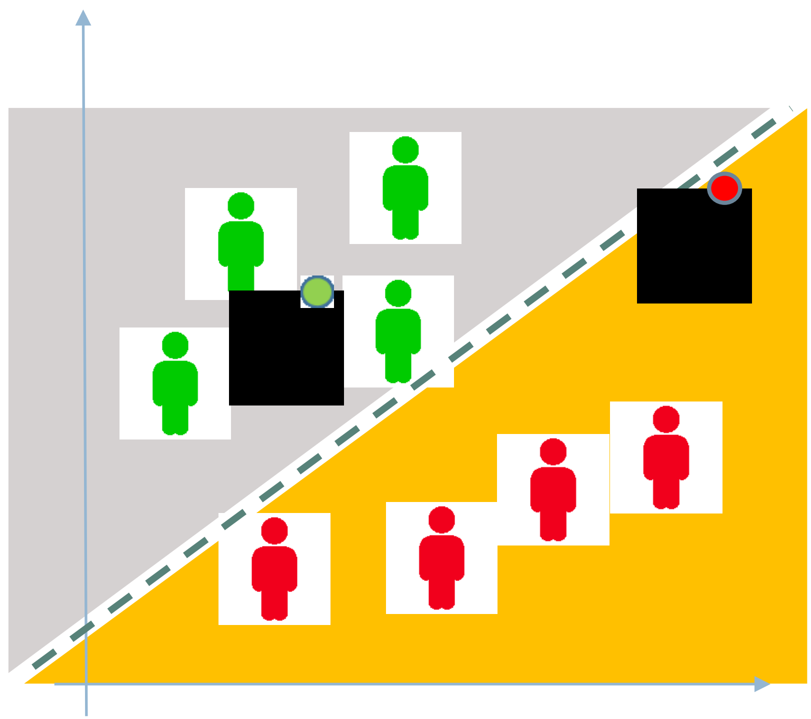
Perceptron Model

$$Y = I(\sum_i w_i X_i - t) \quad \text{or}$$

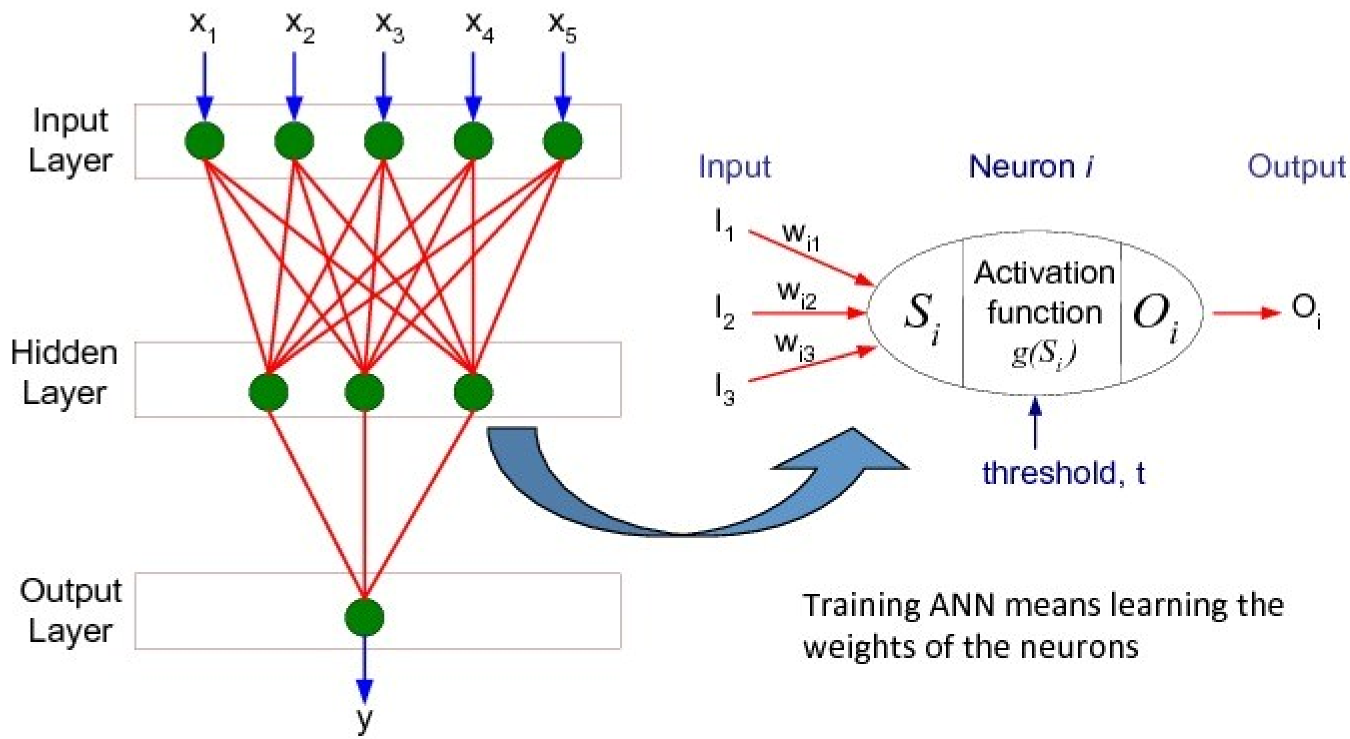
$$Y = sign(\sum_i w_i X_i - t)$$

Sample model on 2-D

- Linear separation line
  - General case: lines can be oblique

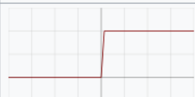
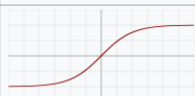
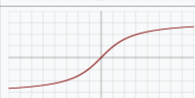


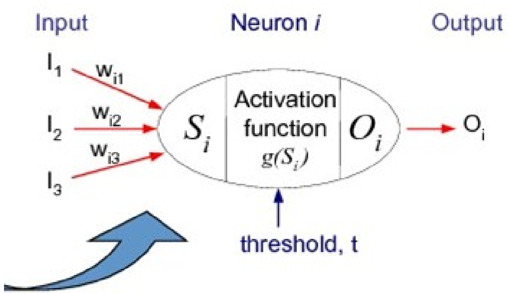
General Structure of ANN



Training ANN means learning the weights of the neurons

- Notice: activation function is fundamental !!!

Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$ [1]
TanH		$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$
ArcTan		$f(x) = \tan^{-1}(x)$



- With Identity (= no activation function) the ANN reduces to a simple perceptron
  - Proof: a linear sum of linear sums, is just another linear sum

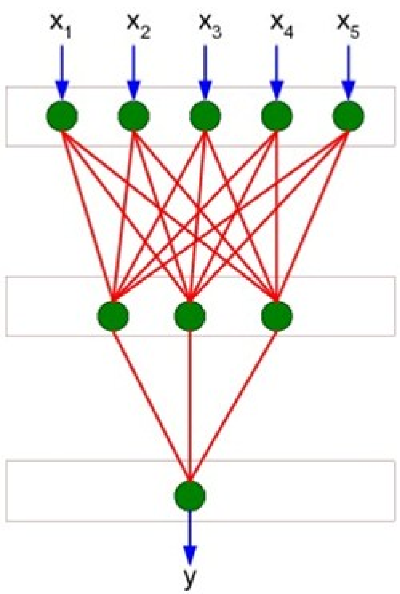
Algorithm for learning ANN

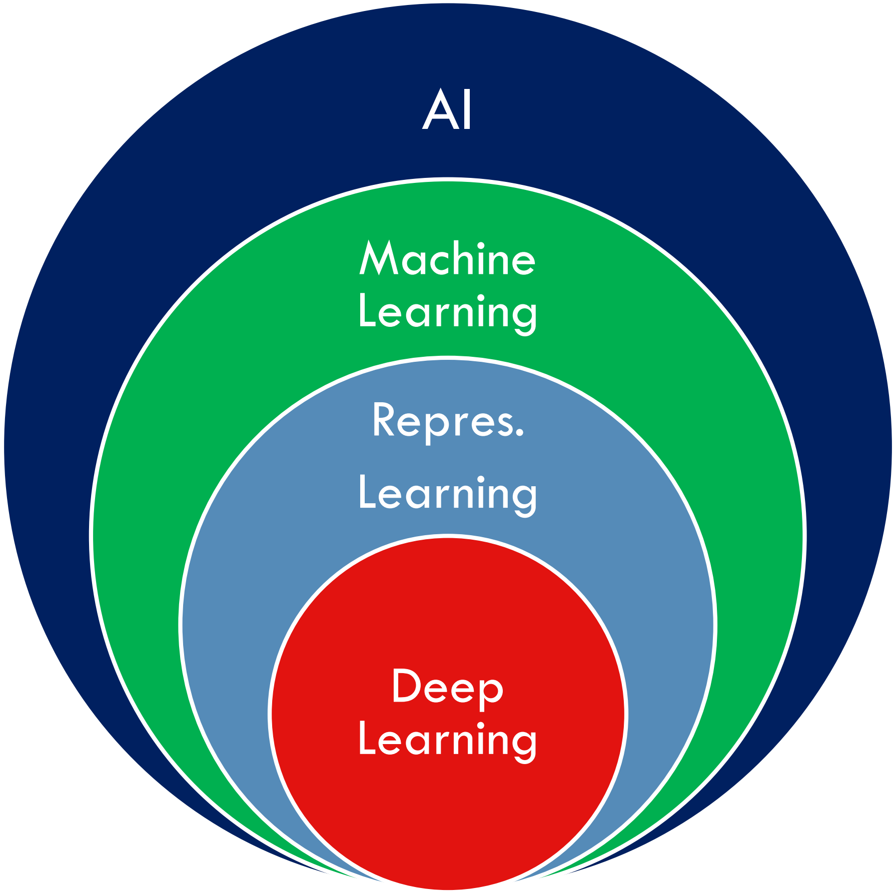
- Initialize the weights (w_0, w_1, …, w_k)

- Adjust the weights in such a way that the output of ANN is consistent with class labels of training examples
  - Objective function: $$E = \sum_i \left[Y_i - f(w_i, X_i) \right]^2$$
  - Find the weights w_i' s that minimize the above objective function
    - e.g., backpropagation algorithm (see lecture notes)

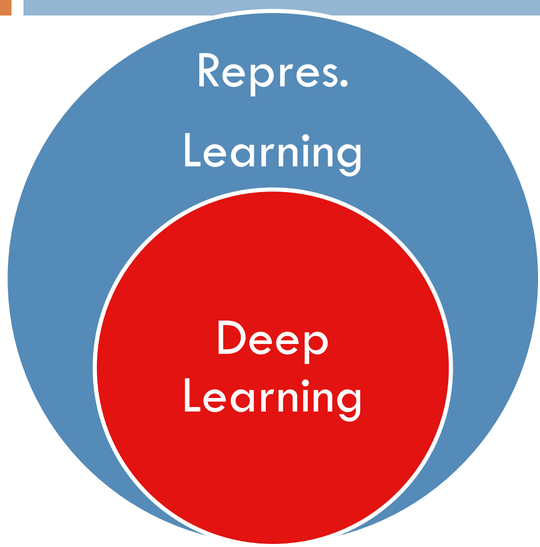
A quick look on Deep Learning

- Various approaches exist
- Basic examples equivalent to ANNs with several levels





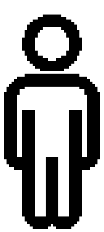
Deep learning



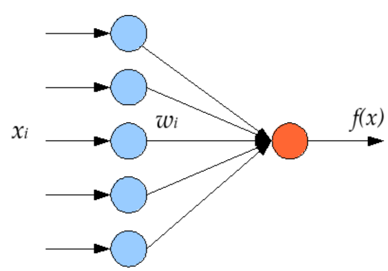
Representation learning methods that

- allow a machine to be fed with raw data and
- to automatically discover the representations needed for detection or classification.

Raw representation



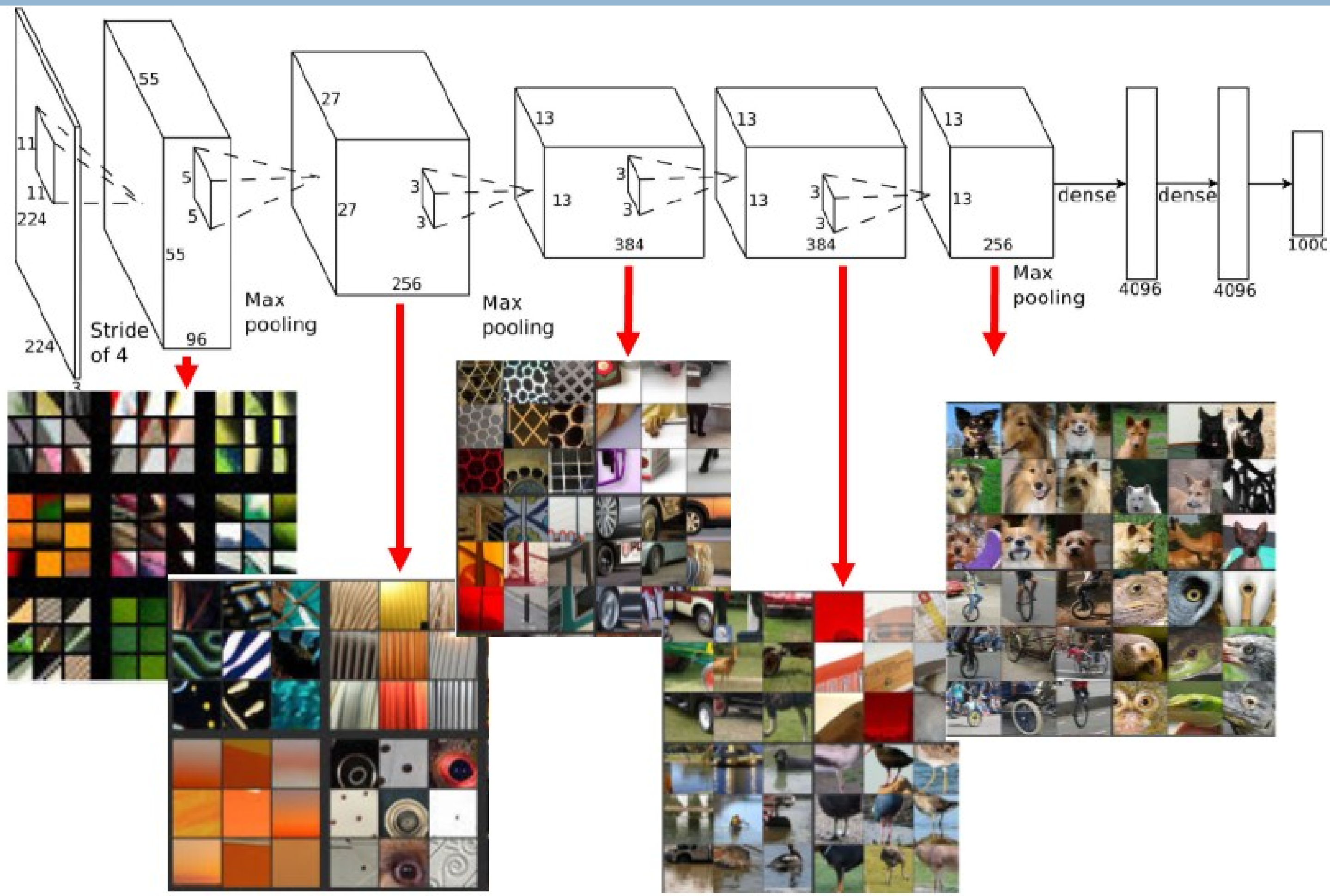
Age	35
Weight	65
Income	23 k€
Children	2
Likes sport	0.3
Likes reading	0.6
Education	high
…	…



Higher-level representation

Young parent	0.9
Fit sportsman	0.1
High-educated reader	0.8
Rich obese	0.0
…	…

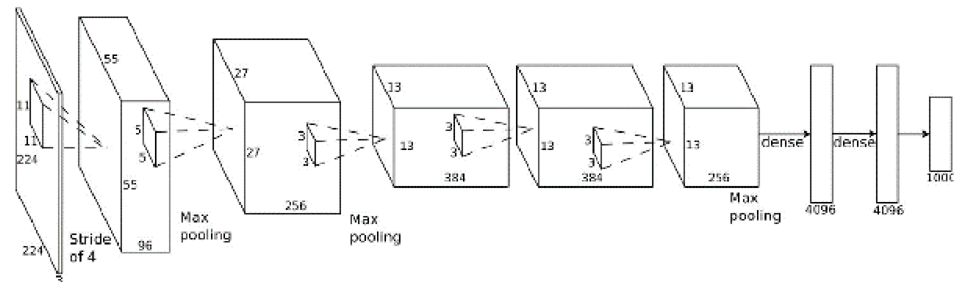
Multiple Levels Of Abstraction



Why now?

(Big) Data

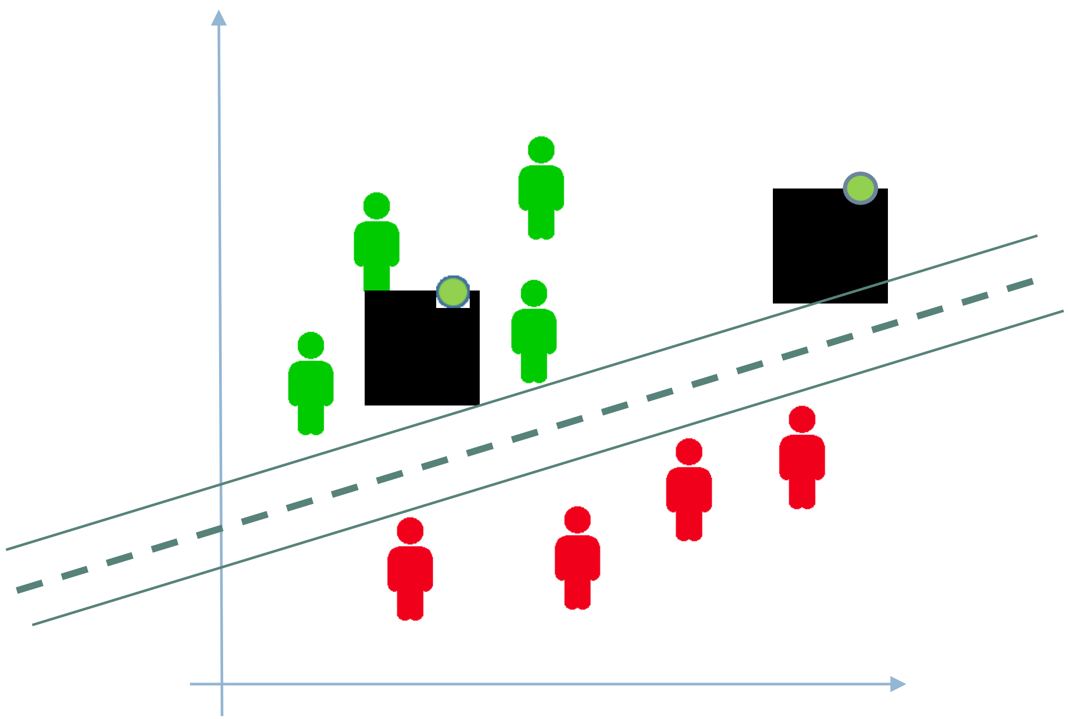
GPU



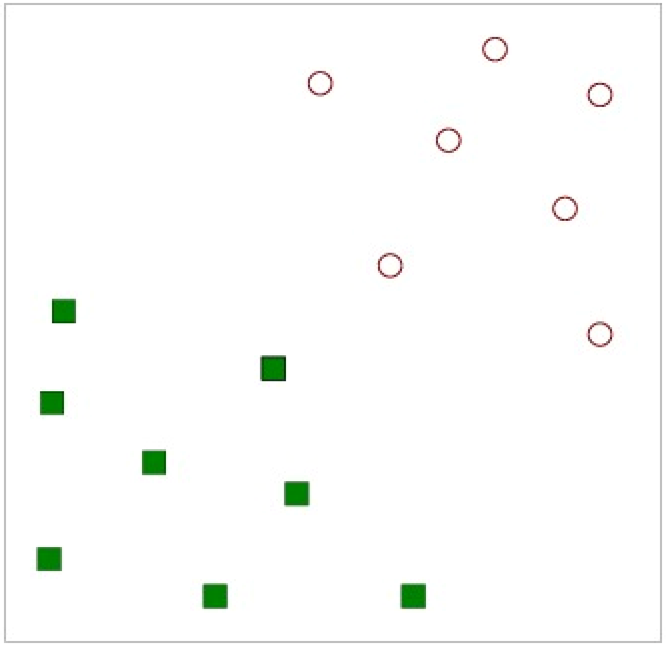
Theory

Support Vector Machines

- Support Vector Machine

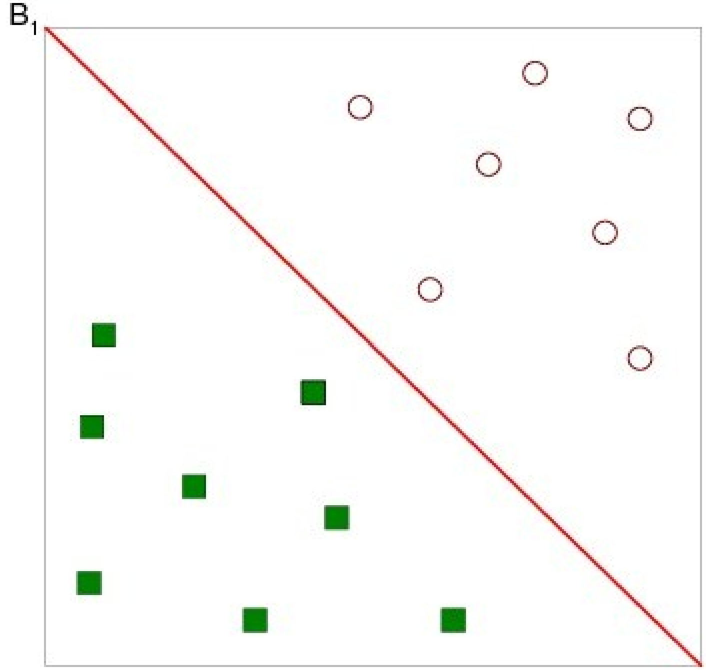


Support Vector Machines



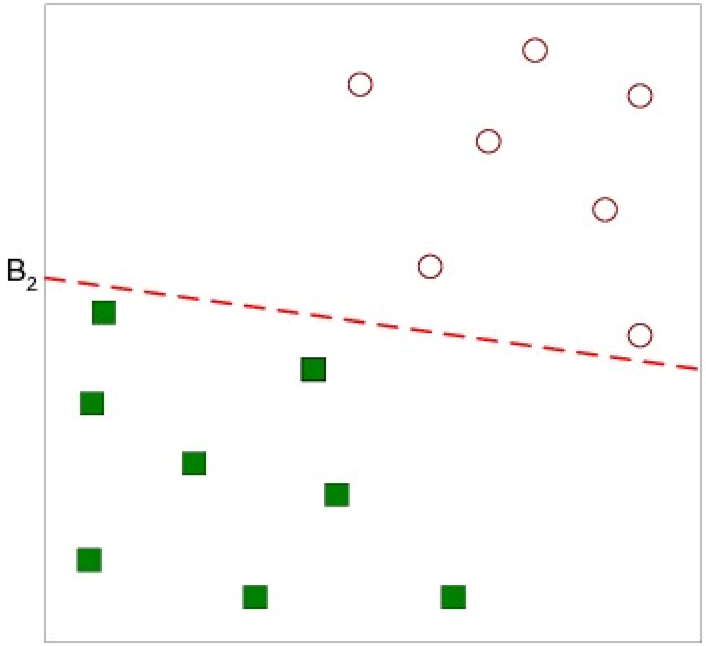
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



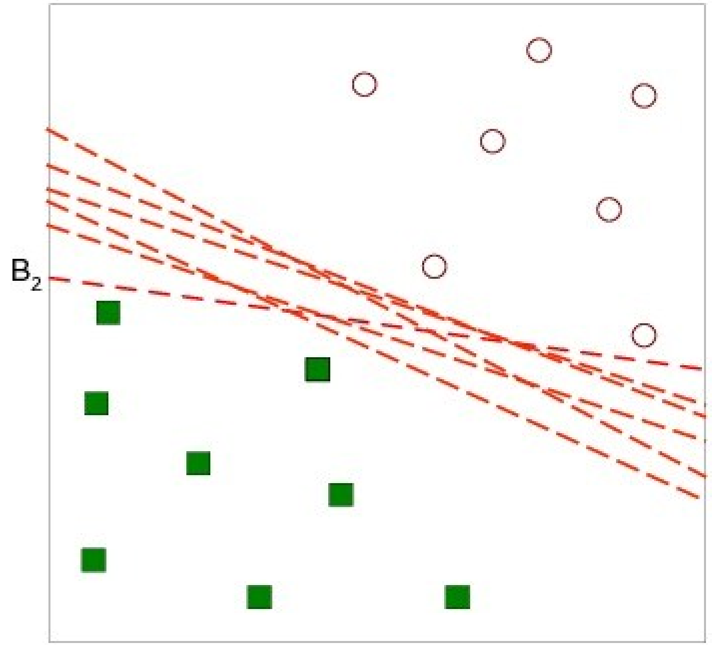
- One Possible Solution

Support Vector Machines



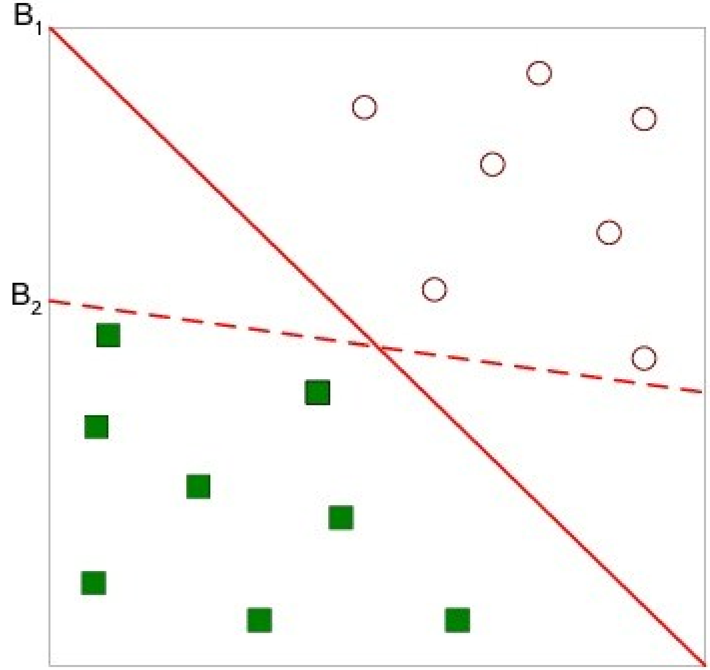
- Another possible solution

Support Vector Machines



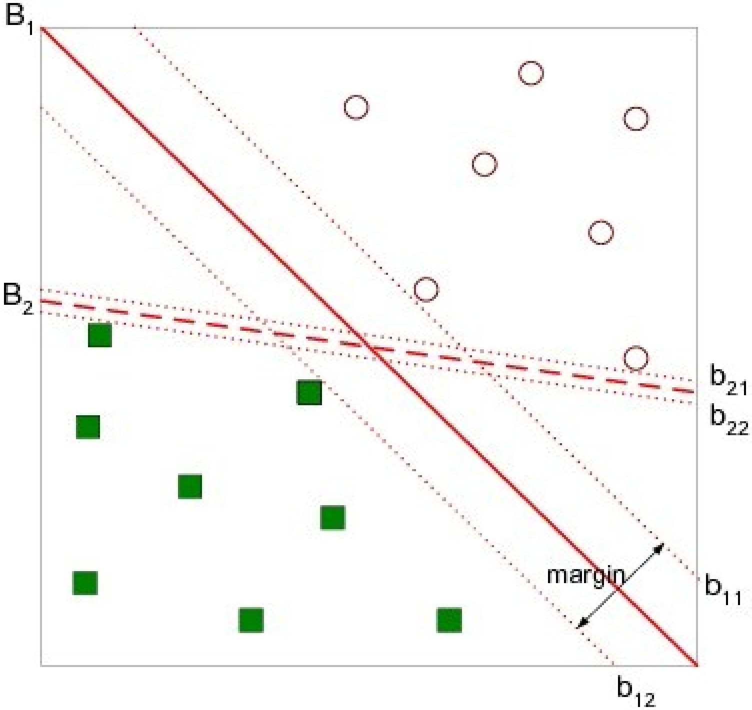
- Other possible solutions

Support Vector Machines



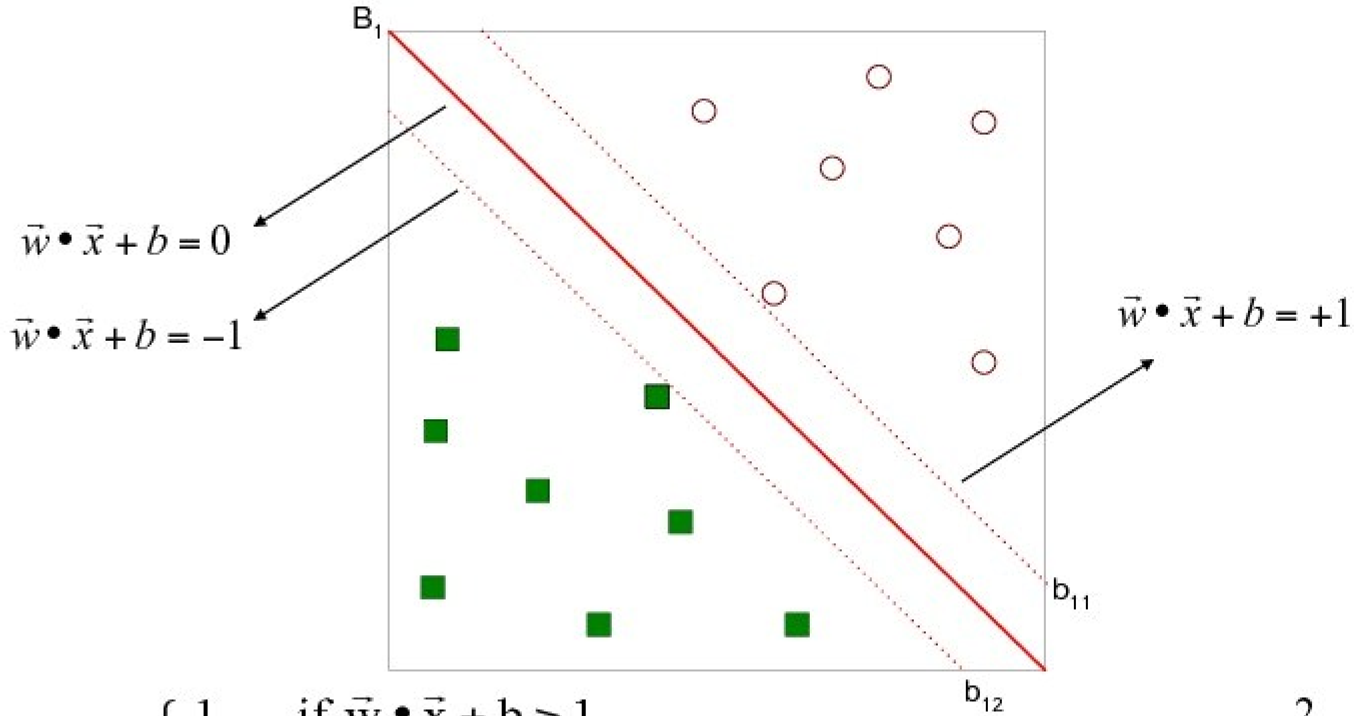
- Which one is better? B1 or B2?
- How do you define better?

Support Vector Machines



- Find hyperplane maximizes the margin => B1 is better than B2

Support Vector Machines



$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

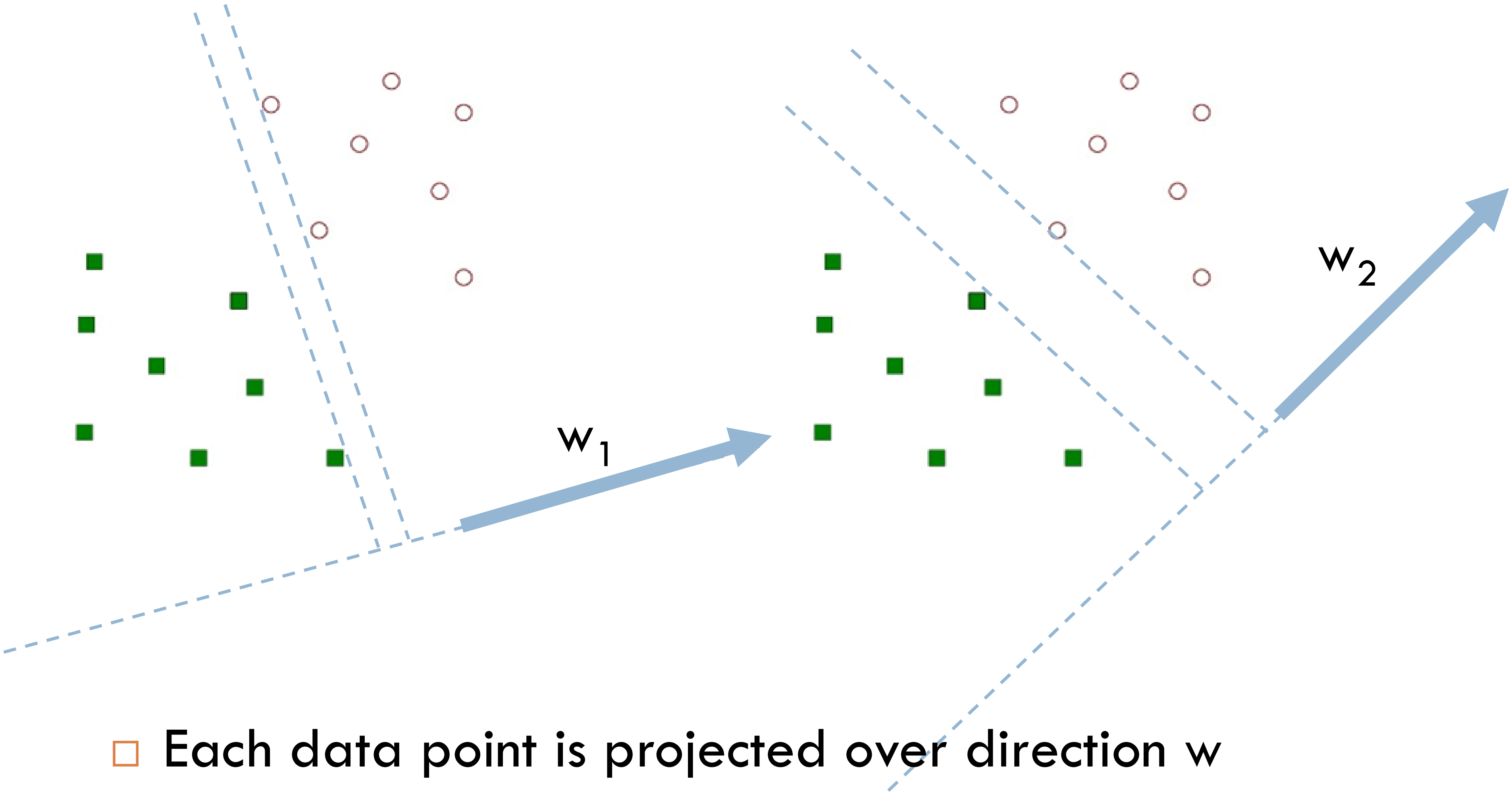
$$\text{Margin} = \frac{2}{\|\vec{w}\|^2}$$

x = data points (n-dimensional vectors)

w = direction of line (n-dimensional vector)

b = displacement of line

Example



- Each data point is projected over direction w
- Projections along w_2 have much larger margin than w_1

Support Vector Machines

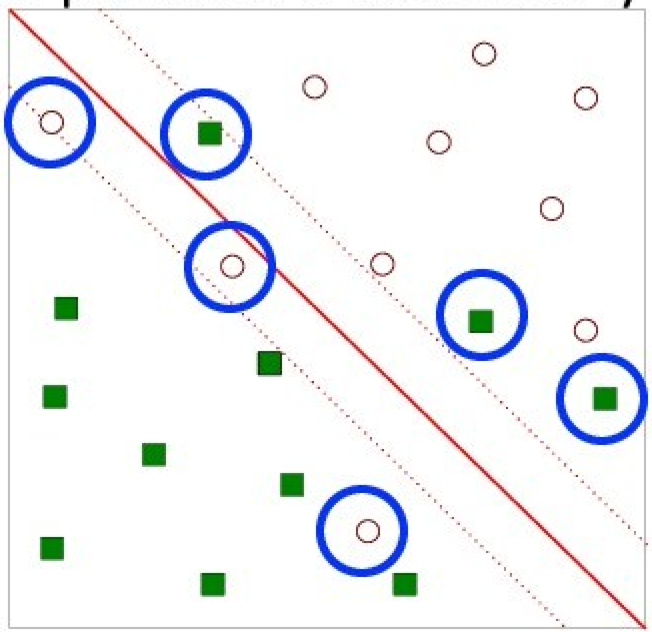
- We want to maximize: $\text{Margin} = \dfrac{2}{\|\vec{w}\|^2}$
  - Which is equivalent to minimizing: $L(w) = \dfrac{\|\vec{w}\|^2}{2}$
  - But subjected to the following constraints:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \ge 1 \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \le -1 \end{cases}$$

    - This is a constrained optimization problem
      - Numerical approaches to solve it (e.g., quadratic programming)

Support Vector Machines

- What if the problem is not linearly separable?



Support Vector Machines

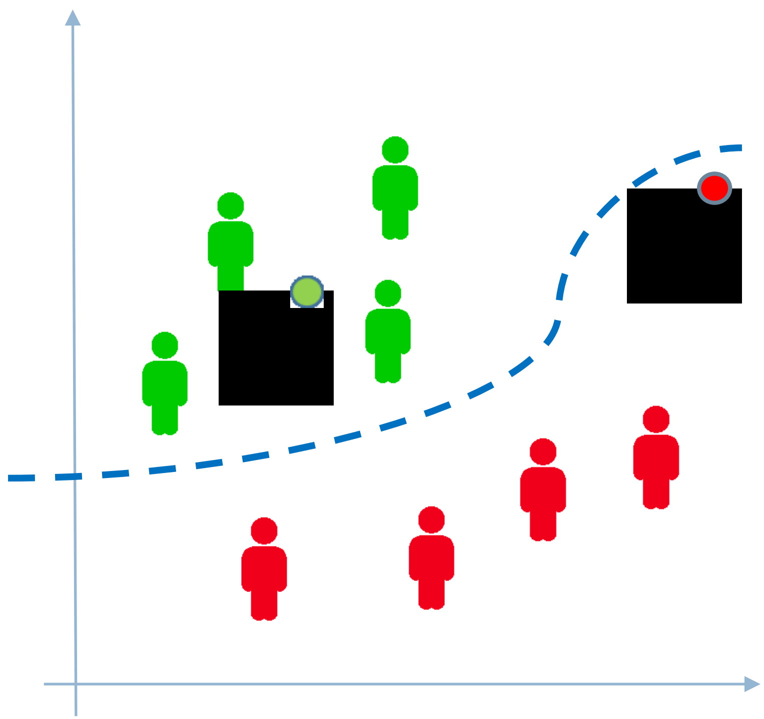
- What if the problem is not linearly separable?
  - Introduce slack variables
    - Need to minimize: $$L(w) = \frac{\|\vec{w}\|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k\right)$$
    - Subject to:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \geq 1 - \xi_\mathrm{i} \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \leq -1 + \xi_\mathrm{i} \end{cases}$$

- Basically, each point is "moved" by a a specific amount along the w direction
- The cost function "pays" for each extra movement

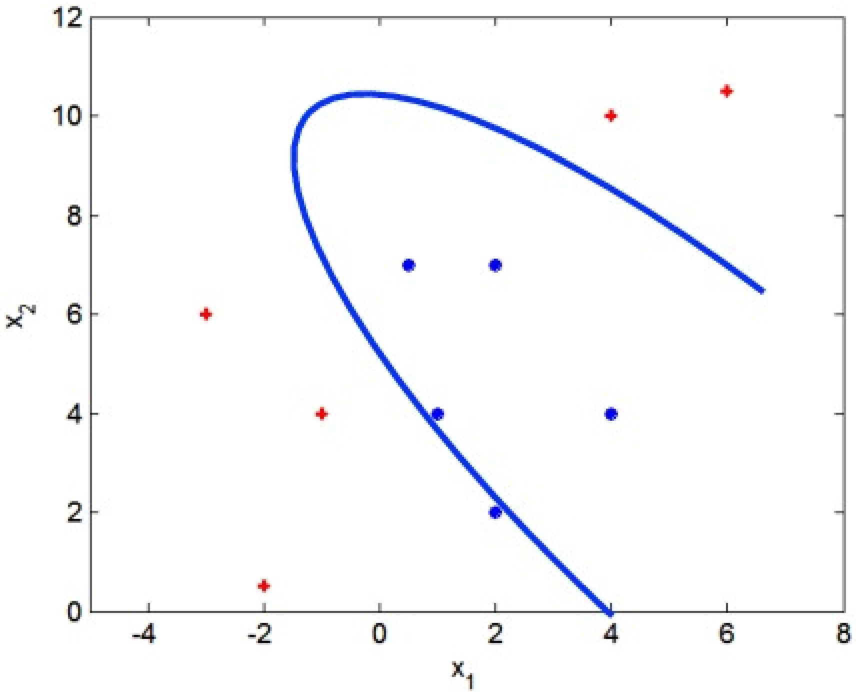
Support Vector Machines

- Non-linear separation line



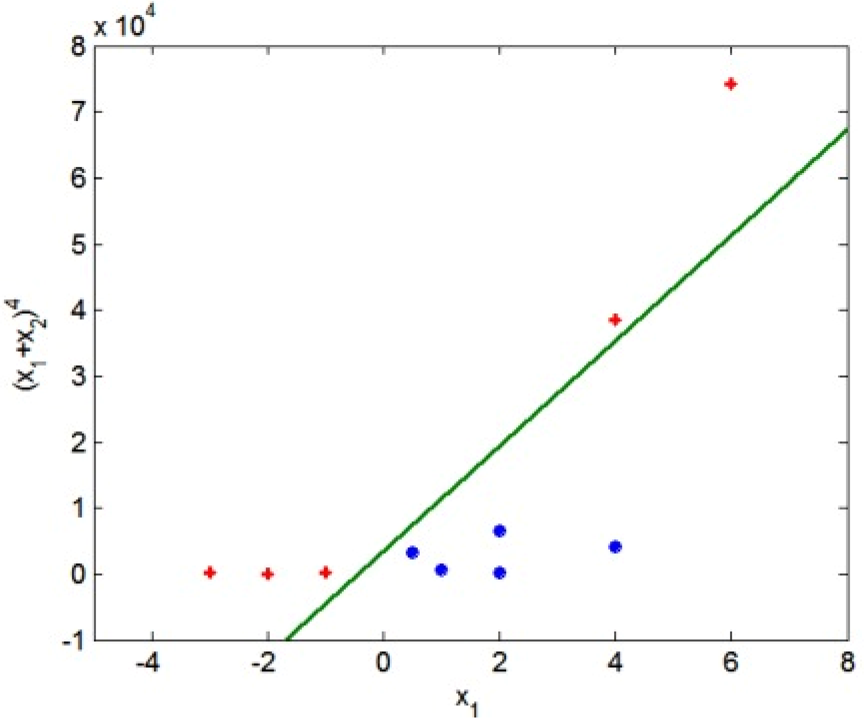
Nonlinear Support Vector Machines

- What if decision boundary is not linear?



Nonlinear Support Vector Machines

- Transform data into higher dimensional space



- Key problem: find the most appropriate set of extra dimensions
  - They are derived from original attributes
  - Most common: x^2, (x+y)^2, and other polynomials