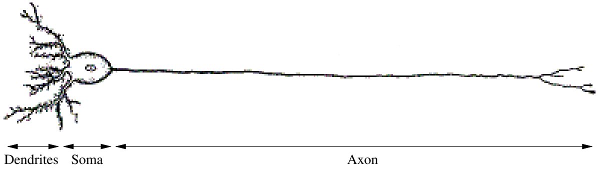
Classification with Separating Hyperplanes

Concept Learning

- Concept Learning:
  - Supervised learning of Boolean-valued functions
  - Learn from positive and negative examples to classify (yes/no) correctly
- Examples of concepts
  - Concrete things: “Dog”, “Mammal”, “Vehicle”, ...
  - Abstract: “Criminal offence”, “Critical thinking”, ...
- Input is an array of attribute values

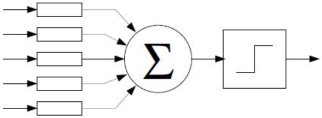


Neuron caricature, "artificial neuron"

- Weighted input signals
- Summing
- Thresholded output

Artificial Neuron

What can a single ”artificial neuron” compute?



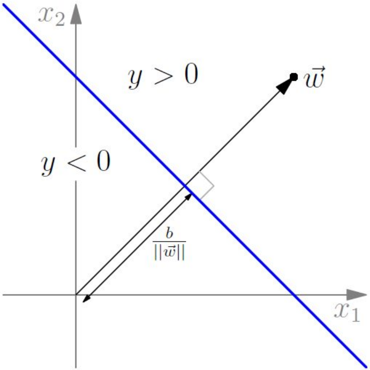
- $\vec{x}$ Input in vector format
- $\vec{w}$ Weights in vector format
- b Threshold
- y Output (True/False, encoded as $+1/-1$)

$$y = \operatorname{sign}\left(\sum_i x_i w_i - b\right)$$

Artificial Neuron

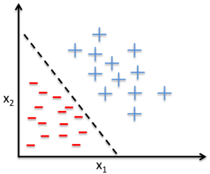
$$y = \mathrm{sign}\left(\sum_i x_i w_i - b\right)$$

Geometrical interpretation

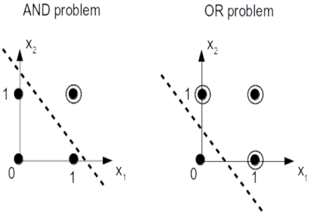


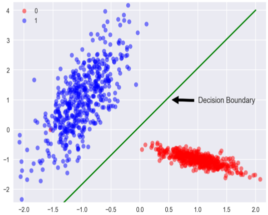
Common trick: treat the variable threshold (b) as an extra weight

Some examples of neuronal learning for classification

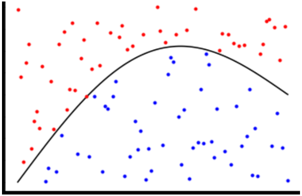


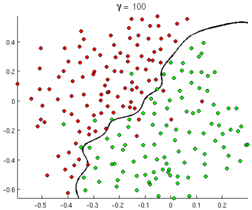
Example of a linear decision boundary for binary classification.

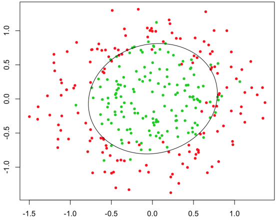




But these ...? How to do that?







Training a linear separator

What does learning mean here?

Learning means finding the best weights w_i

Two good algorithms exist:

- Perceptron Learning
- Delta Rule

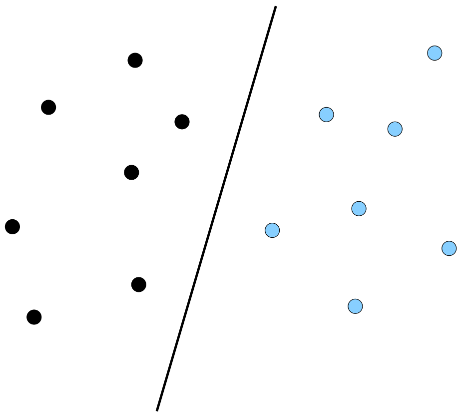
Perceptron Learning [binary output]

- Incremental learning
- Weights only change when the output is wrong
- Update rule: $w_i \leftarrow w_i + \eta(t - o)x_i$
- Always converges if the problem is solvable

Delta Rule (LMS-rule) [continuous output]

- Incremental learning
- Weights always change
- $w_i \leftarrow w_i + \eta(t - \vec{w}^T \vec{x})x_i$
- Converges only in the mean
- Will find an optimal solution even if the problem can not be fully solved

Linear Separation

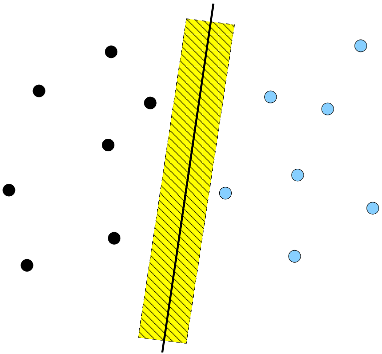


Many acceptable solutions → bad generalization

- Works well for all training data, but creates **Structural Risk**
- Future data samples might get mis-classified

1. Linear separation
2. Structural Risk Minimization
3. Support Vector Machines
4. Kernels
5. Non-separable Classes

Hyperplane with margins
Training data points are *at least* a distance d from the plane



Less arbitrariness $\rightarrow$ better generalization

- Wide margins restrict the possible hyperplanes to choose from
- Less risk to choose a bad hyperplane by accident
- Reduced risk for bad generalization

Minimization of the structural risk $\equiv$ maximization of the margin

Out of all hyperplanes which solve the problem,
the one with widest margin will probably generalize best

Mathematical Formulation

- Separating Hyperplane

$$\vec{w}^T \vec{x} = 0$$

- Hyperplane with a margin

$$\begin{aligned} \vec{w}^T \vec{x} &\geq 1 \quad \text{when } t = 1 \quad \text{(i.e. a positive target)} \\ \vec{w}^T \vec{x} &\leq -1 \quad \text{when } t = -1 \quad \text{(i.e. a negative target)} \end{aligned}$$

- Combined

$$t \vec{w}^T \vec{x} \geq 1$$

How wide is the margin?

1. Select two points, $\vec{p}$ and $\vec{q}$, on the two margins:

$$\vec{w}^T\vec{p} = 1 \qquad \vec{w}^T\vec{q} = -1$$

2. Distance between $\vec{p}$ and $\vec{q}$ *along* $\vec{w}$:

$$2d = \frac{\vec{w}^T}{||\vec{w}||}(\vec{p} - \vec{q})$$

3. Simplify:

$$2d = \frac{\vec{w}^T\vec{p} - \vec{w}^T\vec{q}}{||\vec{w}||} = \frac{1-(-1)}{||\vec{w}||} = \frac{2}{||\vec{w}||}$$

Maximal margin corresponds to minimal length of the weight vector

Best Separating Hyperplane

Minimize

$$\vec{w}^T \vec{w}$$

Constraints

$$t_i \vec{w}^T \vec{x}_i \geq 1 \qquad \forall i$$

1 Linear separation

2 Structural Risk Minimization

3 Support Vector Machines

4 Kernels

5 Non-separable Classes

Observation

Almost everything becomes linearly separable
when represented in high-dimensional spaces

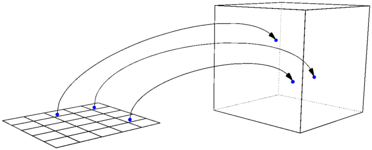
"Ordinary" low-dimensional data can be "scattered" into a high-dimensional space.

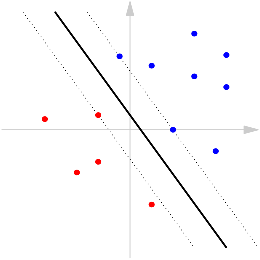
Two problems emerge

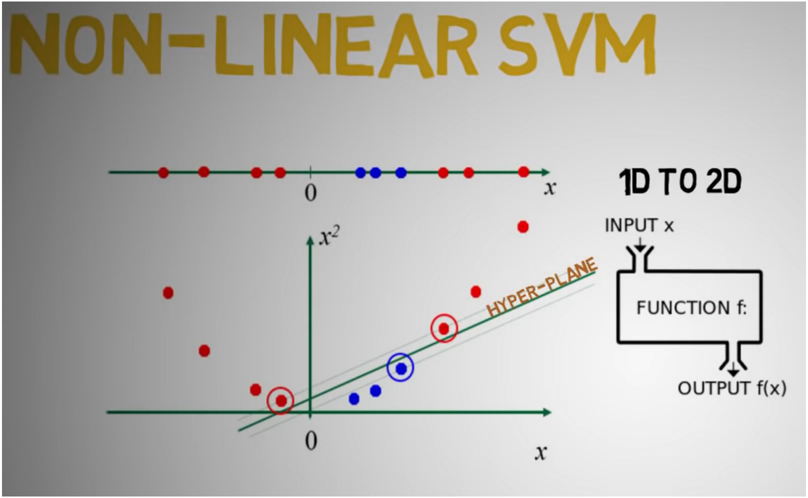
1. Many free parameters $\rightarrow$ bad generalization
2. Extensive computations

Support Vector Machines

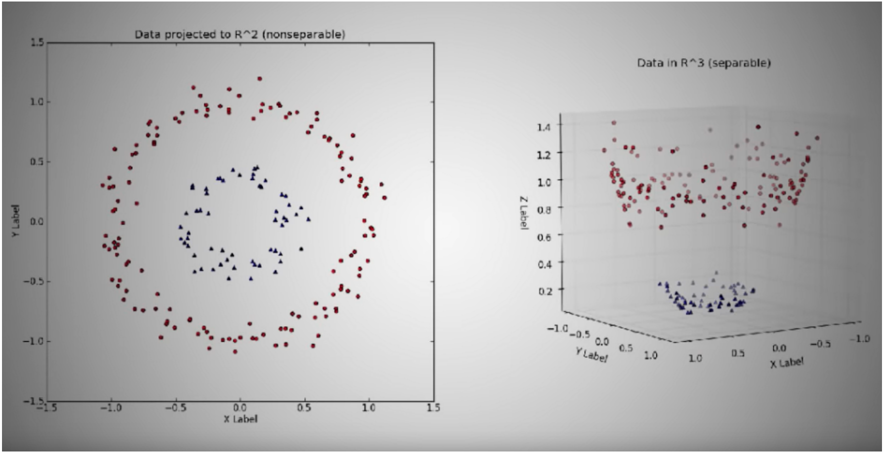
- Transform the input to a suitable high-dimensional space
- Choose the unique separating hyperplane that has maximal margins






NON-LINEAR SVM
0
x
1D TO 2D
INPUT x
x^2
HYPER-PLANE
FUNCTION f:
OUTPUT f(x)
0
x

2D-3D example



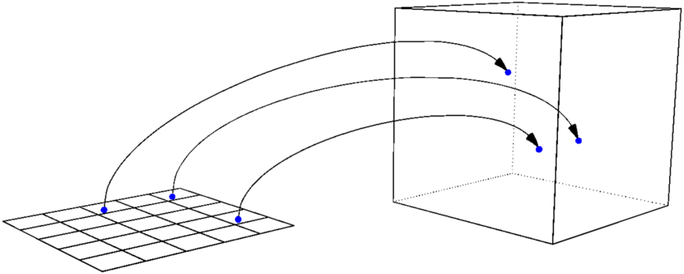
great, but computationally expensive

Support Vector Machines

- Advantages
  - Very good generalization
  - Works well even with few training samples
  - Fast classification
- Disadvantages
  - Non-local weight calculation
  - Hard to implement efficiently

What is the "correct" mapping to high dimensional spaces to use?

Kernels: Only *pretend* that we transform the input data into a high-dimensional feature space!



Idea behind Kernels

Utilize the advantages of a high-dimensional space
without actually representing anything high-dimensional

- Condition: The only operation done in the high-dimensional space is to compute *scalar products* between pairs of items
- Trick: The high-dimensional scalar product is computed using the original (low-dimensional) representation

Example

Points in 2D

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Transformation to 4D

$$\phi(\vec{x}) = \begin{bmatrix} x_1^3 \\ \sqrt{3}x_1^2 x_2 \\ \sqrt{3}x_1 x_2^2 \\ x_2^3 \end{bmatrix}$$

$$\begin{aligned} \phi(\vec{x})^T \cdot \phi(\vec{y}) &= x_1^3 y_1^3 + 3x_1^2 y_1^2 x_2 y_2 + 3x_1 y_1 x_2^2 y_2^2 + x_2^3 y_2^3 \\ &= (x_1 y_1 + x_2 y_2)^3 \\ &= (\vec{x}^T \cdot \vec{y})^3 \\ &= \mathcal{K}(\vec{x}, \vec{y}) \end{aligned}$$

Common Kernels

Polynomials

$$\mathcal{K}(\vec{x}, \vec{y}) = (\vec{x}^T \vec{y} + 1)^p$$

Radial Bases

$$\mathcal{K}(\vec{x}, \vec{y}) = e^{-\frac{1}{2\rho^2}||\vec{x}-\vec{y}||^2}$$

Structural Risk Minimization

Minimize

$$\vec{w}^T\vec{w}$$

Constraints

$$t_i\vec{w}^T\vec{x}_i \geq 1 \qquad \forall i$$

- Non-linear transformation ϕ of input $\vec{x}$

New formulation

Minimize

$$\frac{1}{2}\vec{w}^T\vec{w}$$

Constraints

$$t_i\vec{w}^T\phi(\vec{x}_i) \geq 1 \qquad \forall i$$

Structural Risk Minimization

Minimize

$$\frac{1}{2}\vec{w}^T\vec{w}$$

Constraints

$$t_i\vec{w}^T\phi(\vec{x}_i) \geq 1 \qquad \forall i$$

Lagranges Multiplier Method

$$L = \frac{1}{2}\vec{w}^T\vec{w} - \sum_i \alpha_i \left[t_i\vec{w}^T\phi(\vec{x}_i) - 1\right]$$

Minimize w.r.t. $\vec{w}$, maximize w.r.t. $\alpha_i \geq 0$

$$\frac{\partial L}{\partial \vec{w}} = 0$$

$$L = \frac{1}{2}\vec{w}^T\vec{w} - \sum_i \alpha_i \left[t_i \vec{w}^T \phi(\vec{x}_i) - 1 \right]$$

$$\frac{\partial L}{\partial \vec{w}} = 0 \implies \vec{w} - \sum_i \alpha_i t_i \phi(\vec{x}_i) = 0$$

$$\vec{w} = \sum_i \alpha_i t_i \phi(\vec{x}_i)$$

Use

$$\vec{w} = \sum_i \alpha_i t_i \phi(\vec{x}_i)$$

to eliminate $\vec{w}$

$$L = \frac{1}{2}\vec{w}^T\vec{w} - \sum_i \alpha_i \left[t_i \vec{w}^T \phi(\vec{x}_i) - 1 \right]$$

$$L = \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j t_i t_j \phi(\vec{x}_i)^T\phi(\vec{x}_j) - \sum_{i,j} \alpha_i\alpha_j t_i t_j \phi(\vec{x}_i)^T\phi(\vec{x}_j) + \sum_i \alpha_i$$

$$L = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j t_i t_j \phi(\vec{x}_i)^T\phi(\vec{x}_j)$$

The Dual Problem

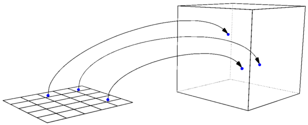
Maximize

$$\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j t_i t_j \phi(\vec{x}_i)^T \phi(\vec{x}_j)$$

Under the constraints

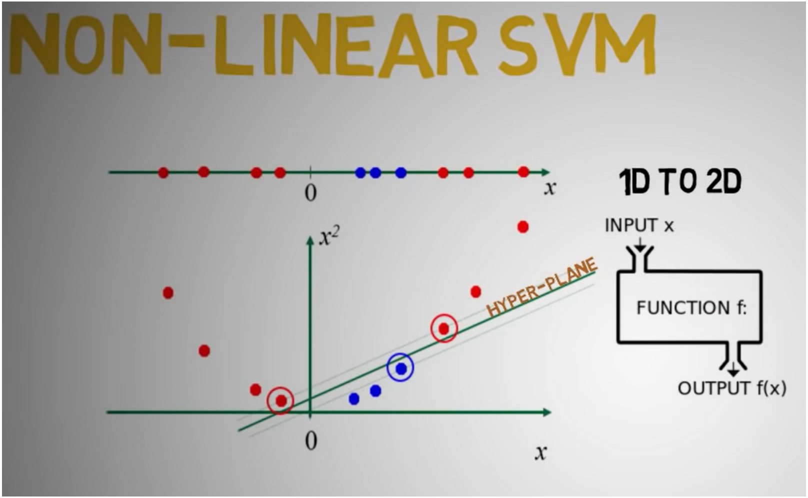
$$\alpha_i \geq 0 \quad \forall i$$

- $\vec{w}$ has disappeared
- $\phi(\vec{x})$ only appear in scalar product pairs



1. Choose a suitable kernel function
2. Compute α_i (solve the maximization problem)
3. $\vec{x}_i$ corresponding to $\alpha_i \neq 0$ are called support vectors
4. Classify new data points via

$$\sum_i \alpha_i t_i \mathcal{K}(\vec{x}, \vec{x}_i) > 0$$

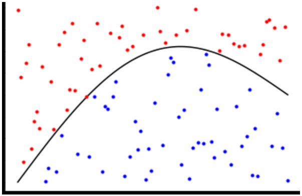

NON-LINEAR SVM
0
x
1D TO 2D
INPUT x
x^2
HYPER-PLANE
FUNCTION f:
OUTPUT f(x)
0
x

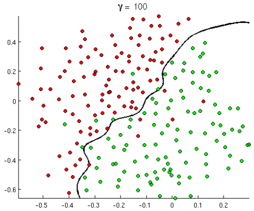
non-separable classes?

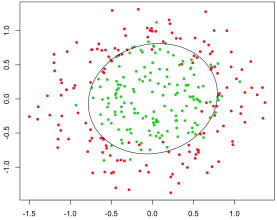
all classes are separable, but is this what we want?

generalization / specialization tradeoff

instead introduce ... slack ...

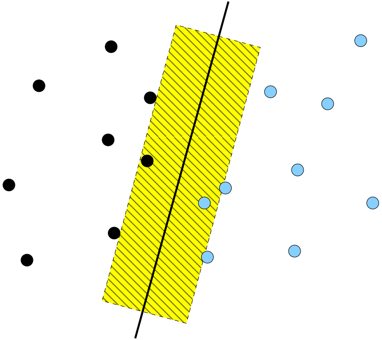






None-Separable Training Samples

Allow for Slack



Re-formulation of the minimization problem

Minimize

$$\frac{1}{2}\vec{w}^T\vec{w} + C\sum_i \xi_i$$

Constraints

$$t_i \vec{w}^T \phi(\vec{x}_i) \geq 1 - \xi_i$$

ξ_i are called *slack variables*

Dual Formulation with Slack

Maximize

$$\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j \phi(\vec{x}_i)^T \phi(\vec{x}_j)$$

With constraints

$$0 \leq \alpha_i \leq C \quad \forall i$$

Otherwise, everything remains as before