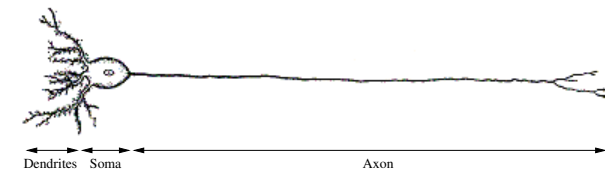


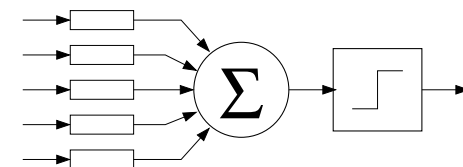
# Classification with Separating Hyperplanes

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

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

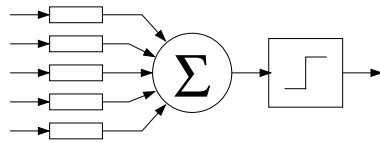


Neuron caricature, "artificial neuron"



- Weighted input signals
- Summing
- Thresholded output

What can a single "artificial neuron" compute?

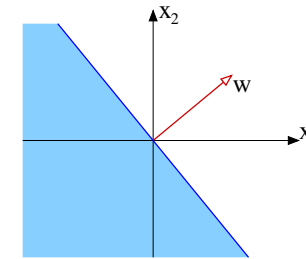


- $\vec{x}$  Input in vector format
- $\vec{w}$  Weights in vector format
- $o$  Output

$$o = \text{sign} \left( \sum_i x_i w_i \right)$$

$$o = \text{sign} \left( \sum_i x_i w_i \right)$$

Geometrical interpretation



Variable threshold  $\equiv$  Not anchored to origin  
 Common trick: treat the threshold as a weight

## 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

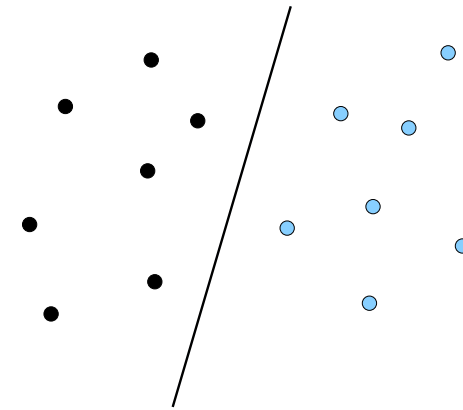
Perceptron Learning

- 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

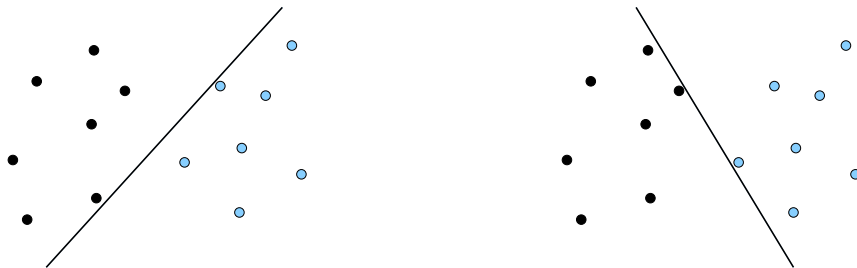
### Delta Rule (LMS-rule)

- 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  $\rightarrow$  bad generalization

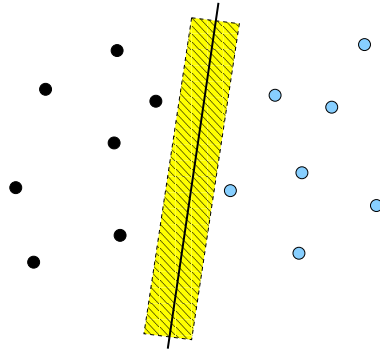


- Structural Risk

- 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 → 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

$$\vec{w}^T \vec{x} \geq 1 \quad \text{when } t = 1$$

$$\vec{w}^T \vec{x} \leq -1 \quad \text{when } t = -1$$

- 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:

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

- 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

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

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

## Support Vector Machines

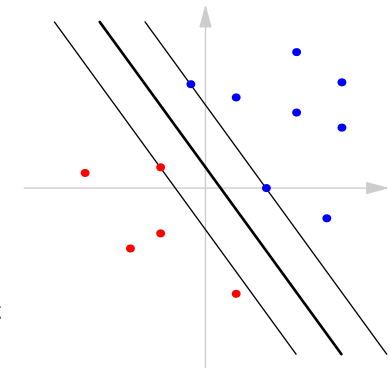
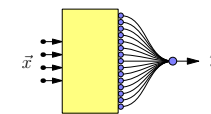
### Observation

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 → bad generalization
- 2 Extensive computations

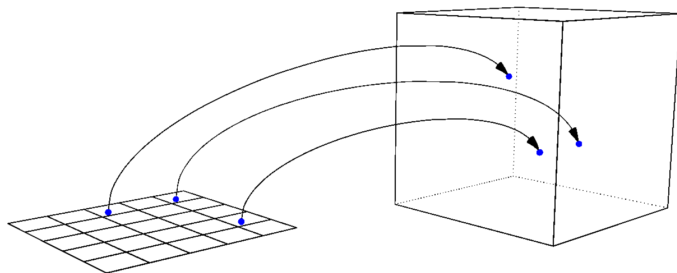


- Transform the input to a suitable high-dimensional space
- Choose the separation that has maximal margins

# 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

Transform input data non-linearly into a high-dimensional feature space



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

## 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 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

$$\phi(\vec{x}) = \begin{bmatrix} x_1^3 \\ \sqrt{3}x_1^2x_2 \\ \sqrt{3}x_1x_2^2 \\ x_2^3 \end{bmatrix}$$

$$\begin{aligned} \phi(\vec{x})^T \cdot \phi(\vec{y}) &= x_1^3y_1^3 + 3x_1^2y_1^2x_2y_2 + 3x_1y_1x_2^2y_2^2 + x_2^3y_2^3 \\ &= (x_1y_1 + x_2y_2)^3 \\ &= (\vec{x}^T \cdot \vec{y})^3 \\ &= \mathcal{K}(\vec{x}, \vec{y}) \end{aligned}$$

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

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

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

### New formulation

Minimize

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

Constraints

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

### Structural Risk Minimization

Minimize

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

Constraints

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

Lagranges Multiplier Method

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

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

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

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

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

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

Use

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

to eliminate  $\vec{w}$

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

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

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

### The Dual Problem

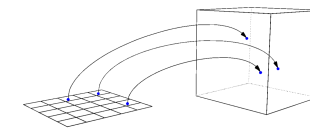
Maximize

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

Under the constraints

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

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



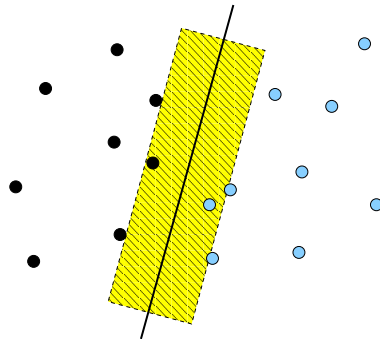
- 1 Choose a suitable kernel function
- 2 Compute  $\alpha_i$  (solve the maximization problem)
- 3  $\vec{x}_i$  corresponding to  $\alpha_i \neq 0$  are called **support vectors**
- 4 Classify new data points via

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



## None-Separable Training Samples

Allow for **Slack**



## Re-formulation of the minimization problem

Minimize

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

Constraints

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

$\xi_i$  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