Support Vector Machines

Classification with Separating Hyperplanes

Concept Learning with Linear separation

Structural Risk Minimization

Support Vector Machines

Kernels

Non-separable Classes

Concept Learning with Linear separation

Structural Risk Minimization

Support Vector Machines

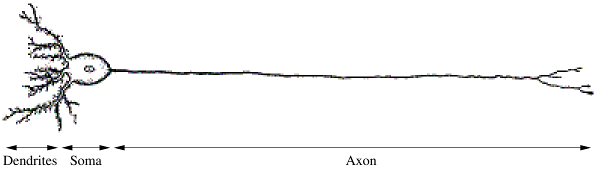
Kernels

Non-separable Classes

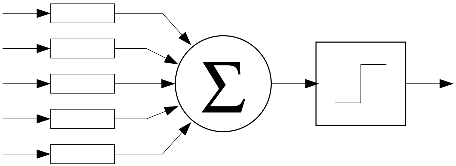
Concept Learning

- 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

Artificial Neural Networks



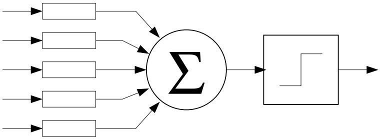
Neuron caricature, "artificial neuron"



- Weighted input signals
- Summing
- Thresholded output

Artificial Neuron

What can a single ”artificial neuron” compute?



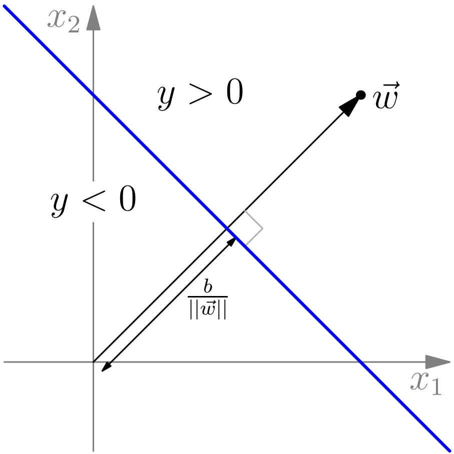
- $\vec{x}$ Input in vector format
- $\vec{w}$ Weights in vector format
- b Threshold
- y Output (True/False, encoded as $+1/-1$)

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

Artificial Neuron

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

Geometrical interpretation



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

Training a Linear Separator

What does learning mean here?

Learning means finding the best parameters: w_i, b
so that the actual output (y) becomes equal to the desired output (t)

Several efficient algorithms exist:

- Error correction
  Perceptron Learning
- Error/Loss minimization
  Delta Rule
  Logistic Regression

Training a Linear Separator

Perceptron Learning

- Incremental learning
- Weights only change when the output is wrong
- Update rule:

$$w_i \leftarrow w_i + \eta\delta x_i$$
$$b \leftarrow b - \eta\delta$$
$$\text{where } \delta = \frac{(t - y)}{2}$$

- Always converges if the problem is solvable

Training a Linear Separator

Delta Rule

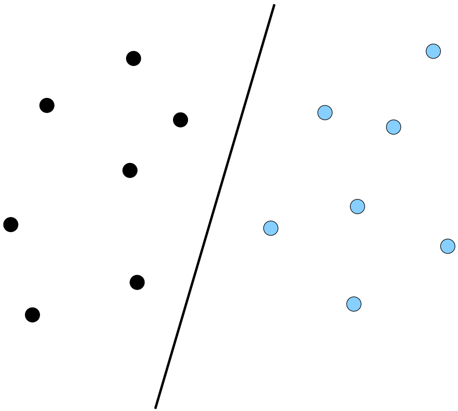
- Minimize the mismatch between the target and the output *before* thresholding
- Minimize $\sum_{k \in \mathcal{D}} ||t_k - \vec{w}^T \vec{x}_k + b||^2$
- Incremental version: Stochastic Gradient Descent
  For each sample: $\vec{w}, b \leftarrow \vec{w}, b - \eta \ \mathrm{grad}_{\vec{w},b} ||t - \vec{w}^T \vec{x} + b||^2$
-

$$w_i \leftarrow w_i + \eta \delta x_i$$

$$b \leftarrow b - \eta \delta$$

$$\text{where } \delta = t - \vec{w}^T \vec{x} + b$$

Linear Separation



Linear Separation

Many acceptable solutions $\rightarrow$ bad generalization

- Structural Risk

Concept Learning with Linear separation

Structural Risk Minimization

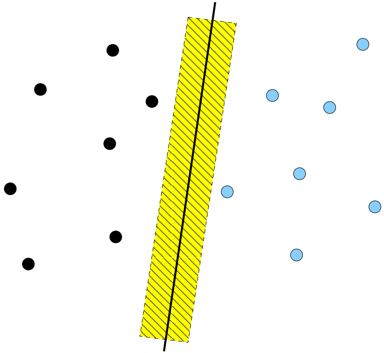
Support Vector Machines

Kernels

Non-separable Classes

Structural Risk Minimization

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



Less arbitrariness $\rightarrow$ better generalization

Structural Risk Minimization

- 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

Structural Risk Minimization

Mathematical Formulation

- Separating Hyperplane

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

- Hyperplane with a margin

$$\begin{aligned} \vec{w}^T\vec{x}_i - b &\geq 1 \qquad \text{when } t_i = 1 \\ \vec{w}^T\vec{x}_i - b &\leq -1 \qquad \text{when } t_i = -1 \end{aligned}$$

- Combined

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

Structural Risk Minimization

How wide is the margin?

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

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

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

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

3. Simplify:

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

Maximal margin corresponds to minimal length of the weight vector

Structural Risk Minimization

Best Separating Hyperplane

Minimize

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

Constraints

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

Support Vector Machines

Observation

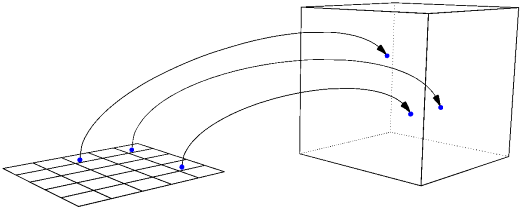
Almost everything becomes linearly separable
when represented in high-dimensional spaces

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

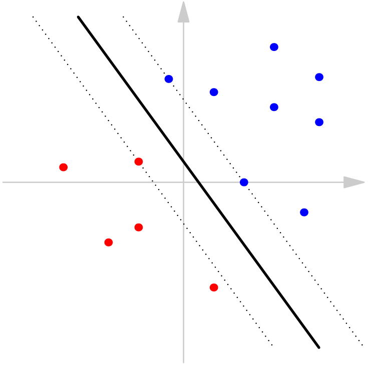
Two problems emerge

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

Support Vector Machines



1. Transform the input to a suitable high-dimensional space
2. Choose the unique separating hyperplane that has maximal margins
3. Classify new data using this hyperplane

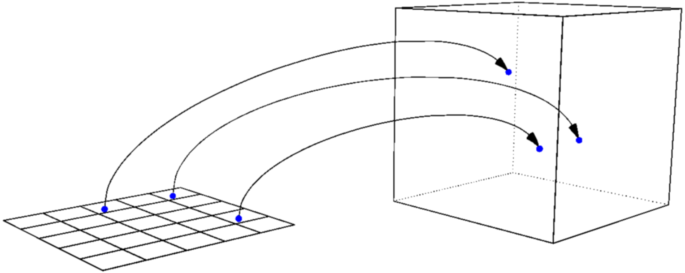


Support Vector Machines

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

Kernels

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



Kernels

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

Kernels

Example

Transformation to 4D

Points in 2D

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

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

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

Kernels

Common Kernels

Polynomials

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

Radial Bases

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

Kernels

Other possible Kernels

String kernels

$$\mathcal{K}(\text{"AATCCGCTAG"}, \text{"AACTCGAG"})$$

Graph kernels

$$\mathcal{K}(< n_1, e_1 >, < n_2, e_2 >)$$

Kernels

Structural Risk Minimization

Minimize

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

Constraints

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

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

New formulation

Minimize

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

Constraints

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

Structural Risk Minimization

Minimize

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

Constraints

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

Lagrange formulation with Karush Kuhn Tucker (KKT) multipliers: α_i

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

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

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

Kernels

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

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

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

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_i \alpha_i t_i = 0$$

Kernels

Use

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

to eliminate $\vec{w}$

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

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

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

Kernels

The Dual Problem

Maximize

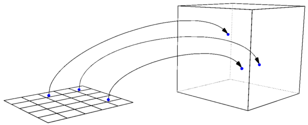
$$\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j t_i t_j \phi(\vec{x}_i)^T \phi(\vec{x}_j)$$

Under the constraints

$$\sum_i \alpha_i t_i = 0 \quad \text{and} \quad \alpha_i \geq 0 \quad \forall i$$

- $\vec{w}$ has disappeared
- $\phi(\vec{x})$ only appear in scalar product pairs $\Rightarrow$ we can use kernels

Kernels



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 the sign of the indicator function:

$$\mathrm{ind}(\vec{y}) = \sum_i \alpha_i t_i \mathcal{K}(\vec{y}, \vec{x}_i) - b$$

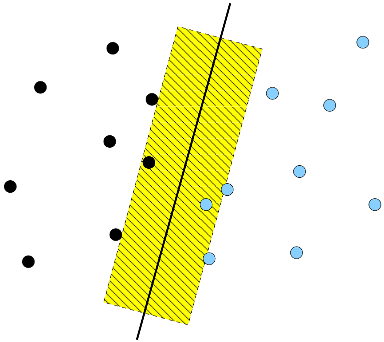
5. b can be calculated using any support vector x_k

$$\mathrm{ind}(\vec{x}_k) = t_k \quad \Rightarrow \quad b = \sum_i \alpha_i t_i \mathcal{K}(\vec{y}, \vec{x}_i) - t_k$$

Non-separable Classes

Non-Separable Training Samples

Allow for Slack



Non-separable Classes

Re-formulation of the minimization problem

Minimize

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

Constraints

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

ξ_i are called *slack variables*

Non-separable Classes

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

$$\sum_i \alpha_i t_i = 0 \quad \text{and} \quad 0 \leq \alpha_i \leq C \quad \forall i$$

Otherwise, everything remains as before