

Support Vector Machines

Kernel Methods

- 1 Structural Risk Minimization
 - High Dimensional Spaces
 - Margins
 - Mathematical Formulation
- 2 Kernels
 - Bypassing High-Dimensional Computations
 - Re-Formulation of the Minimization Task
- 3 Support Vector Machines
 - Classification with Minimal Risk
 - Even Less Risk
 - Function Approximation

- 1 Structural Risk Minimization
 - High Dimensional Spaces
 - Margins
 - Mathematical Formulation
- 2 Kernels
 - Bypassing High-Dimensional Computations
 - Re-Formulation of the Minimization Task
- 3 Support Vector Machines
 - Classification with Minimal Risk
 - Even Less Risk
 - Function Approximation

Observation

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

Observation

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

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

Observation

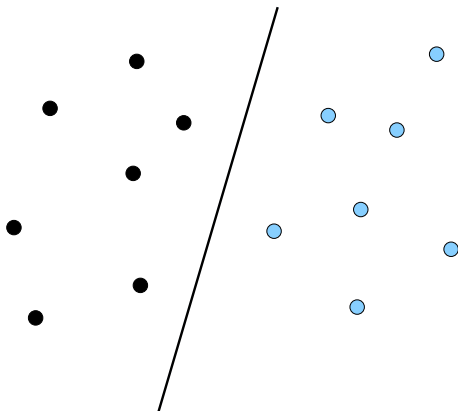
Almost everything becomes linearly separable when represented in high-dimensional spaces

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

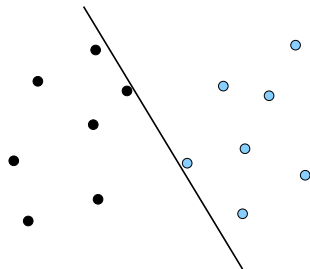
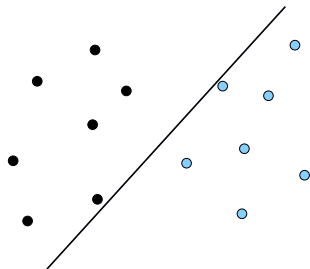
Two problems emerge

- 1 Many free parameters → bad generalization
- 2 Extensive computations

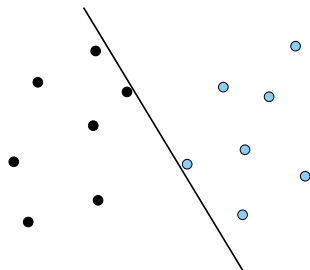
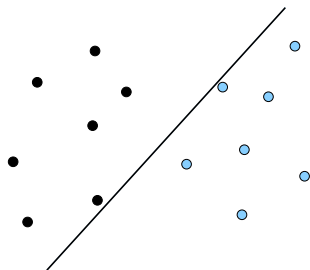
Linear Separation



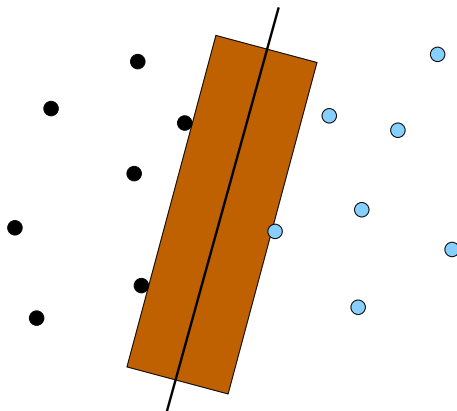
Many acceptable solutions



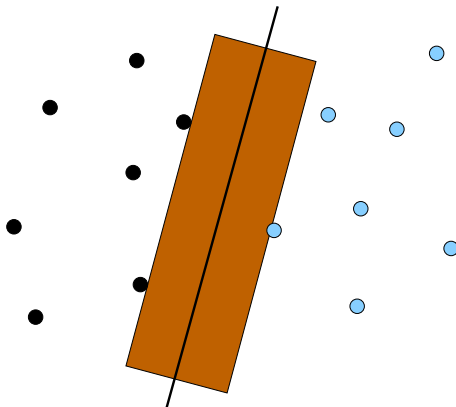
Many acceptable solutions \rightarrow bad generalization



Hyperplane with margins

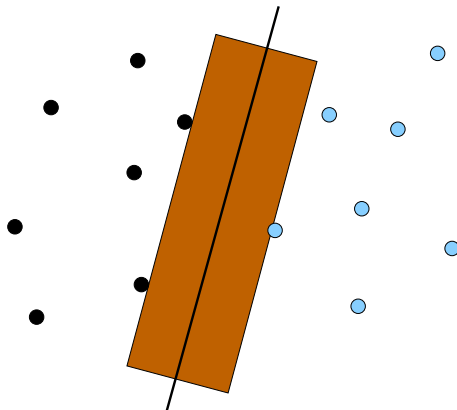


Hyperplane with margins



Less arbitrariness

Hyperplane with margins



Less arbitrariness → better generalization

- Wide margins reduces the VC-dimension

- Wide margins reduces the VC-dimension
- More limited classifier

- Wide margins reduces the VC-dimension
- More limited classifier
- Reduced risk for bad generalization

- Wide margins reduces the VC-dimension
- More limited classifier
- Reduced risk for bad generalization

Minimization of the structural risk \equiv maximization of the margin

- Wide margins reduces the VC-dimension
- More limited classifier
- 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 **generalize best**

Separating Hyperplane

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

Separating Hyperplane

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

Hyperplane with a margin

$$\vec{w}^T \vec{x} + b \geq 1 \quad \text{when } d = 1$$

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

Separating Hyperplane

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

Hyperplane with a margin

$$\vec{w}^T \vec{x} + b \geq 1 \quad \text{when } d = 1$$

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

Margin width $\frac{2}{\vec{w}^T \vec{w}}$

Separating Hyperplane

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

Hyperplane with a margin

$$\vec{w}^T \vec{x} + b \geq 1 \quad \text{when } d = 1$$

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

Margin width $\frac{2}{\vec{w}^T \vec{w}}$

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

Best Separating Hyperplane

Minimize

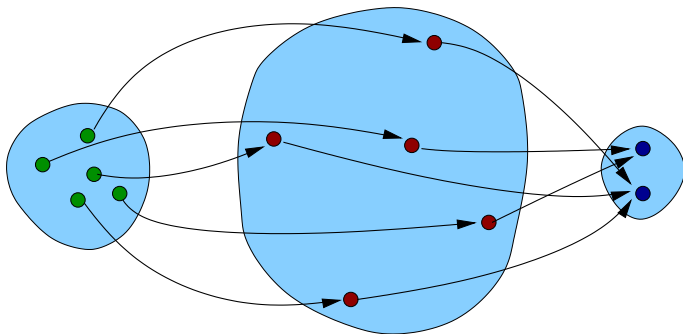
$$\frac{1}{2} \vec{w}^T \vec{w}$$

Constraints

$$d_i(\vec{w}^T \vec{x}_i + b) \geq 1 \quad \forall i$$

- 1 Structural Risk Minimization
 - High Dimensional Spaces
 - Margins
 - Mathematical Formulation
- 2 Kernels
 - Bypassing High-Dimensional Computations
 - Re-Formulation of the Minimization Task
- 3 Support Vector Machines
 - Classification with Minimal Risk
 - Even Less Risk
 - Function Approximation

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



Idea behind Kernels

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

Idea behind Kernels

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

- **Condition:** The only thing done to items in the high-dimensional space is to compute *scalar products* between pairs of items

Idea behind Kernels

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

- **Condition:** The only thing done to items in the high-dimensional space is to compute *scalar products* between pairs of items
- Common in ANN

Idea behind Kernels

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

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

Example

Example

Points in 2D

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

Example

Points in 2D

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

Transformation to 4D

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

Example

Points in 2D

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

$$\phi(\vec{x})^T \cdot \phi(\vec{y})$$

Transformation to 4D

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

Example

Points in 2D

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

Transformation to 4D

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

$$\phi(\vec{x})^T \cdot \phi(\vec{y}) = x_1^3y_1^3 + 3x_1^2y_1^2x_2y_2 + 3x_1y_1x_2^2y_2^2 + x_2^3y_2^3$$

Example

Points in 2D

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

Transformation to 4D

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

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

Example

Points in 2D

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

Transformation to 4D

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

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

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

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

Constraints

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

Structural Risk Minimization

Minimize

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

Constraints

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

Lagrange's Multiplier Method

Structural Risk Minimization

Minimize

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

Constraints

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

Lagranges Multiplier Method

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

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

Structural Risk Minimization

Minimize

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

Constraints

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

Lagranges Multiplier Method

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

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

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

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

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

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

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

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

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

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

$$\frac{\partial L}{\partial b} = 0$$

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

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

$$\frac{\partial L}{\partial b} = 0 \implies \sum_i \alpha_i d_i = 0$$

Use

$$\vec{w} = \sum_i \alpha_i d_i \phi(\vec{x}_i) \quad \sum_i \alpha_i d_i = 0$$

to eliminate \vec{w}

Use

$$\vec{w} = \sum_i \alpha_i d_i \phi(\vec{x}_i) \quad \sum_i \alpha_i d_i = 0$$

to eliminate \vec{w}

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

Use

$$\vec{w} = \sum_i \alpha_i d_i \phi(\vec{x}_i) \quad \sum_i \alpha_i d_i = 0$$

to eliminate \vec{w}

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

$$\begin{aligned} L &= \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j d_i d_j \phi(\vec{x}_i)^T \phi(\vec{x}_j) \\ &\quad - \sum_{i,j} \alpha_i \alpha_j d_i d_j \phi(\vec{x}_i)^T \phi(\vec{x}_j) - b \cdot 0 + \sum_i \alpha_i \end{aligned}$$

Use

$$\vec{w} = \sum_i \alpha_i d_i \phi(\vec{x}_i) \quad \sum_i \alpha_i d_i = 0$$

to eliminate \vec{w}

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

$$\begin{aligned} L &= \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j d_i d_j \phi(\vec{x}_i)^T \phi(\vec{x}_j) \\ &\quad - \sum_{i,j} \alpha_i \alpha_j d_i d_j \phi(\vec{x}_i)^T \phi(\vec{x}_j) - b \cdot 0 + \sum_i \alpha_i \end{aligned}$$

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

The Dual Problem

Maximize

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

Under the constraints

$$\sum_i \alpha_i d_i = 0 \quad \alpha_i \geq 0 \quad \forall i$$

The Dual Problem

Maximize

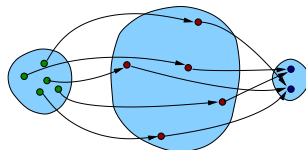
$$\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j d_i d_j \phi(\vec{x}_i)^T \phi(\vec{x}_j)$$

Under the constraints

$$\sum_i \alpha_i d_i = 0 \quad \alpha_i \geq 0 \quad \forall i$$

$\phi(\vec{x})$ and \vec{w} are never explicitly computed

- 1 Structural Risk Minimization
 - High Dimensional Spaces
 - Margins
 - Mathematical Formulation
- 2 Kernels
 - Bypassing High-Dimensional Computations
 - Re-Formulation of the Minimization Task
- 3 Support Vector Machines
 - Classification with Minimal Risk
 - Even Less Risk
 - Function Approximation



- 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

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

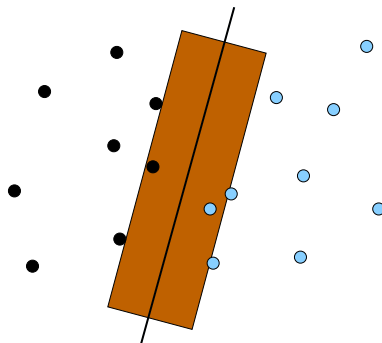
None-Separable Training Samples

None-Separable Training Samples

Allow for **Slack**

None-Separable Training Samples

Allow for **Slack**



Re-formulation of the minimization problem

Re-formulation of the minimization problem

Minimize

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

Constraints

$$d_i(\vec{w}^T \phi(\vec{x}_i) + b) \geq 1$$

Re-formulation of the minimization problem

Minimize

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

Constraints

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

Re-formulation of the minimization problem

Minimize

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

Constraints

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

Maximization with Slack

Maximize

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

With constraints

$$\sum_i \alpha_i d_i = 0 \quad 0 \leq \alpha_i \quad \forall i$$

Maximization with Slack

Maximize

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

With constraints

$$\sum_i \alpha_i d_i = 0 \quad 0 \leq \alpha_i \leq C \quad \forall i$$

Maximization with Slack

Maximize

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

With constraints

$$\sum_i \alpha_i d_i = 0 \quad 0 \leq \alpha_i \leq C \quad \forall i$$

Otherwise, everything remains as before

Support vector methods can also be used for function approximation

