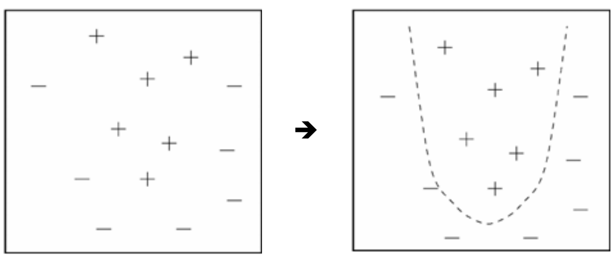
Support Vector Machines and Kernels

CS472/CS473 – Fall 2005

Outline

- **Transform a linear learner into a non-linear learner**
- **Kernels can make high-dimensional spaces tractable**
- **Kernels can make non-vectorial data tractable**

Non-Linear Problems



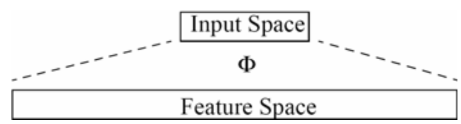
Problem:

- some tasks have non-linear structure
- no hyperplane is sufficiently accurate

How can SVMs learn non-linear classification rules?

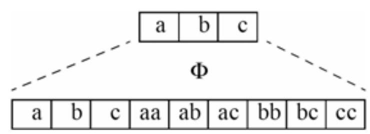
Extending the Hypothesis Space

Idea: add more features



➔ Learn linear rule in feature space.

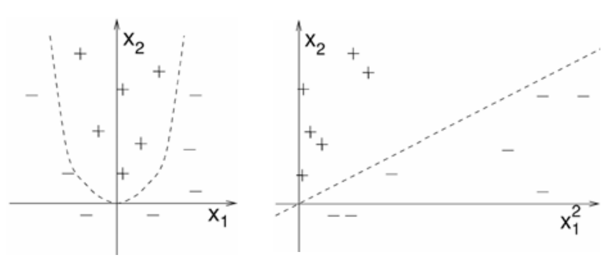
Example:



➔ The separating hyperplane in feature space is degree two polynomial in input space.

Example

- **Input Space:** $\vec{x} = (x_1, x_2)$ **(2 attributes)**
- **Feature Space:** $\Phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1)$ **(6 attributes)**



Dual (Batch) Perceptron Algorithm

Input: $S = ((\vec{x}_1, y_1), ..., (\vec{x}_n, y_n))$, $\vec{x}_i \in \Re^N$, $y_i \in \{-1, 1\}$, $I \in [1, 2, ..]$

Dual Algorithm:

- $\forall i \in [1..n] : \alpha_i = 0$
- repeat
  - FOR i=1 TO n
    - IF $y_i\left(\sum_{j=1}^{n} \alpha_j y_j (\vec{x}_j \cdot \vec{x}_i)\right) \leq 0$
      - $\alpha_i = \alpha_i + 1$
    - ENDIF
  - ENDFOR
- until I iterations reached

Primal Algorithm:

- $\vec{w} = \vec{0}$, $k = 0$
- repeat
  - FOR i=1 TO n
    - IF $y_i(\vec{w} \cdot \vec{x}_i) \leq 0$
      - $\vec{w} = \vec{w} + y_i \vec{x}_i$
    - ENDIF
  - ENDFOR
- until I iterations reached

Dual SVM Optimization Problem

- **Primal Optimization Problem**

$$\text{minimize: } P(\vec{w}, b, \vec{\xi}) = \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to: } \forall_{i=1}^{n}: y_i[\vec{w}\cdot\vec{x}_i + b] \geq 1 - \xi_i$$
$$\forall_{i=1}^{n}: \xi_i > 0$$

- **Dual Optimization Problem**

$$\text{maximize: } D(\vec{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\vec{x}_i \cdot \vec{x}_j)$$
$$\text{subject to: } \sum_{i=1}^{n} y_i \alpha_i = 0$$
$$\forall_{i=1}^{n}: 0 \leq \alpha_i \leq C$$

- **Theorem:** If w^* is the solution of the Primal and α^* is the solution of the Dual, then $\vec{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \vec{x}_i$

Kernels

Problem: Very many Parameters! Polynomials of degree p over N attributes in input space lead to attributes in feature space!

Solution: [Boser et al.] The dual OP depends only on inner products => Kernel Functions

$$K(\vec{a}, \vec{b}) = \Phi(\vec{a}) \cdot \Phi(\vec{b})$$

Example: For $\Phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1)$ calculating $K(\vec{a}, \vec{b}) = [\vec{a}\cdot\vec{b} + 1]^2$ computes inner product in feature space.

➔ no need to represent feature space explicitly.

SVM with Kernel

Training:

$$\text{maximize: } D(\vec{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j K(\vec{x}_i, \vec{x}_j)$$
$$\text{subject to: } \sum_{i=1}^{n} y_i \alpha_i = 0$$
$$\forall_{i=1}^{n}: 0 \leq \alpha_i \leq C$$

Classification:

$$h(\vec{x}) = sign\left(\left[\sum_{i=1}^{n}\alpha_i y_i \Phi(\vec{x}_i)\right]\cdot\Phi(\vec{x}) + b\right)$$
$$= sign\left(\sum_{i=1}^{n}\alpha_i y_i K(\vec{x}_i, \vec{x}) + b\right)$$

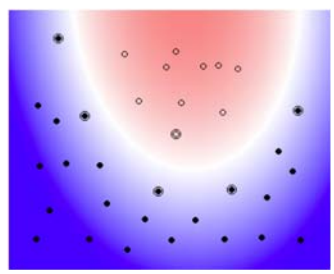
New hypotheses spaces through new Kernels:

- **Linear:** $K(\vec{a}, \vec{b}) = \vec{a}\cdot\vec{b}$
- **Polynomial:** $K(\vec{a}, \vec{b}) = [\vec{a}\cdot\vec{b} + 1]^d$
- **Radial Basis Function:** $K(\vec{a}, \vec{b}) = exp(-\gamma[\vec{a} - \vec{b}]^2)$
- **Sigmoid:**

Examples of Kernels

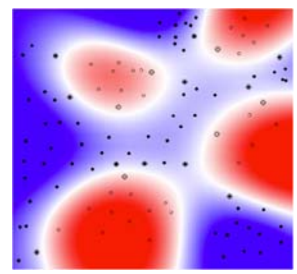
Polynomial

$K(\vec{a}, \vec{b}) = [\vec{a}\cdot\vec{b} + 1]^2$



Radial Basis Function

$K(\vec{a}, \vec{b}) = exp(-\gamma[\vec{a} - \vec{b}]^2)$



Kernels for Non-Vectorial Data

- **Applications with Non-Vectorial Input Data**
  → classify non-vectorial objects
  - Protein classification (x is string of amino acids)
  - Drug activity prediction (x is molecule structure)
  - Information extraction (x is sentence of words)
  - Etc.
- **Applications with Non-Vectorial Output Data**
  → predict non-vectorial objects
  - Natural Language Parsing (y is parse tree)
  - Noun-Phrase Co-reference Resolution (y is clustering)
  - Search engines (y is ranking)

➔ **Kernels can compute inner products efficiently!**

Properties of SVMs with Kernels

- **Expressiveness**
  - Can represent any boolean function (for appropriate choice of kernel)
  - Can represent any sufficiently "smooth" function to arbitrary accuracy (for appropriate choice of kernel)
- **Computational**
  - Objective function has no local optima (only one global)
  - Independent of dimensionality of feature space
- **Design decisions**
  - Kernel type and parameters
  - Value of C