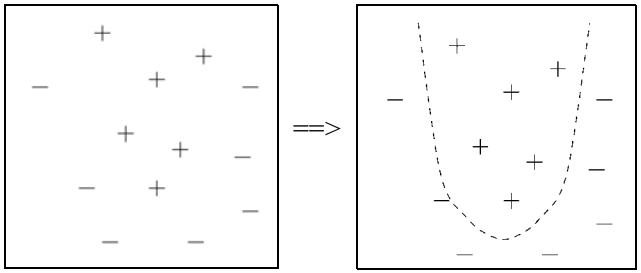
Kernels

CS678 Advanced Topics in Machine Learning
Thorsten Joachims
Spring 2003

Outline:

- A representation of the hyperplane in terms of the training examples.
- How to transform a linear learner into a non-linear learner!
- How can kernels can make high-dimensional spaces tractable?
- How can kernels make non-vectorial data tractable?

Non-Linear Problems



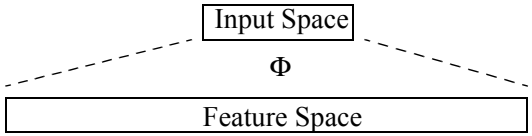
Problem:

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

How can SVMs learn non-linear classification rules?

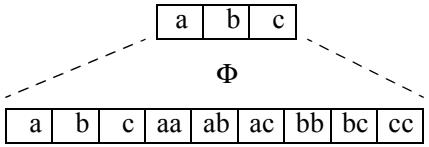
Extending the Hypothesis Space

Idea:



==> Find hyperplane in feature space!

Example:

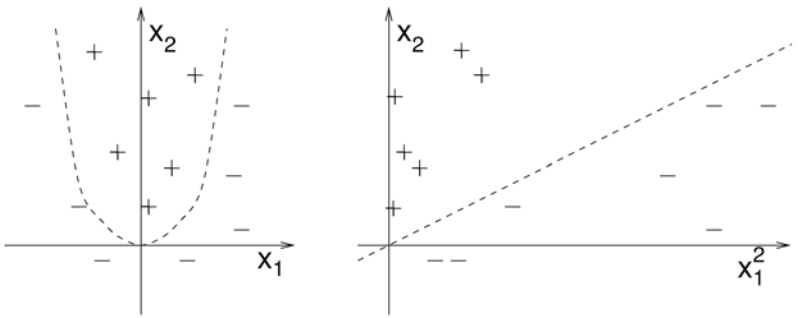


==> The separating hyperplane in features space is a degree two polynomial in input space.

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)



Kernels

Problem: Very many Parameters! Polynomials of degree p over N attributes in input space lead to $O(N^p)$ 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

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

gives inner product in feature space.

We do not need to represent the feature space explicitly!

SVM with Kernels

Training: maximize $L(\vec{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\vec{x_i}, \vec{x_j})$

$$\text{s. t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad und \qquad 0 \le \alpha_i \le C$$

Classification: For new example x $\quad h(\vec{x}) = sign\left(\sum_{x_i \in SV} \alpha_i y_i K(\vec{x_i},\vec{x}) + b \right)$

New hypotheses spaces through new Kernels:

Linear: $K(\vec{x_i},\vec{x_j}) = \vec{x_i} \cdot \vec{x_j}$

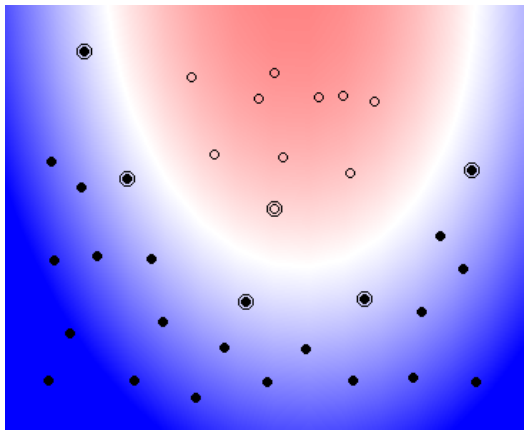
Polynomial: $K(\vec{x_i},\vec{x_j}) = [\vec{x_i} \cdot \vec{x_j} + 1]^d$

Radial Basis Functions: $K(\vec{x_i},\vec{x_j}) = \exp(-\gamma |\vec{x_i} - \vec{x_j}|^2)$

Sigmoid: $K(\vec{x_i},\vec{x_j}) = \tanh(\gamma(\vec{x_i} - \vec{x_j}) + c)$

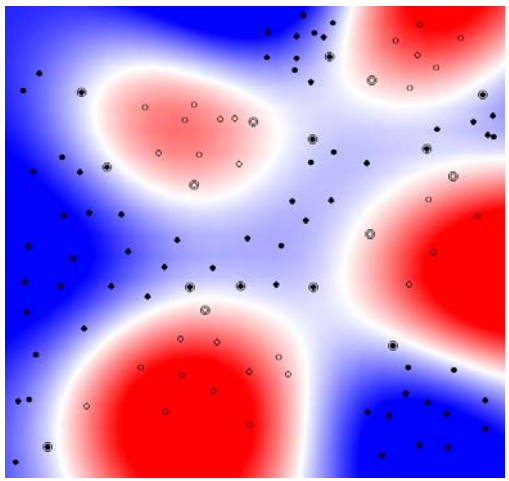
Example: SVM with Polynomial of Degree 2

Kernel: $K(\vec{x_i},\vec{x_j}) = [\vec{x_i} \cdot \vec{x_j} + 1]^2$



Example: SVM with RBF-Kernel

Kernel: $K(\vec{x_i},\vec{x_j}) = \exp(-\gamma |\vec{x_i} - \vec{x_j}|^2)$



What is a Valid Kernel?

Definition: Let X be a nonempty set. A function $K(\vec{x}_i, \vec{x}_j)$ is a valid kernel in X if for all n and all $\vec{x}_1, \ldots, \vec{x}_n \in X$ it produces a Gram matrix

$$G_{ij} = K(\vec{x}_i, \vec{x}_j)$$

that is symmetric

$$G = G^T$$

and positive semi-definite

$$\forall \vec{\alpha} \; \vec{\alpha}^T G \vec{\alpha} = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(\vec{x}_i, \vec{x}_j) \geq 0$$

How to Construct Valid Kernels?

Theorem: Let K_1 and K_2 be valid Kernels over $X \times X$, $X \subseteq \Re^N$, $a \geq 0$, $0 \leq \lambda \leq 1$, f a real-valued function on X, $\phi: X \to \Re^m$ with K_3 a kernel over $\Re^m \times \Re^m$, and K a summetric positive semi-definite matrix. Then the following functions are valid Kernels

$$K(\vec{x}, \vec{z}) = \lambda K_1(\vec{x}, \vec{z}) + (1 - \lambda) K_2(\vec{x}, \vec{z})$$

$$K(\vec{x}, \vec{z}) = a K_1(\vec{x}, \vec{z})$$

$$K(\vec{x}, \vec{z}) = K_1(\vec{x}, \vec{z}) K_2(\vec{x}, \vec{z})$$

$$K(\vec{x}, \vec{z}) = f(\vec{x}) f(\vec{z})$$

$$K(\vec{x}, \vec{z}) = K_3(\phi(\vec{x}), \phi(\vec{z}))$$

$$K(\vec{x}, \vec{z}) = \vec{x}^T K \vec{z}$$

=> Construct complex Kernels from simple Kernels.

Kernels for Discrete and Structured Representations

Kernels for Sequences: Two sequences are similar, if the have many common and consecutive subsequences.

Example [Lodhi et al., 2000]: For $0 \leq \lambda \leq 1$ consider the following features space

	c-a	c-t	a-t	b-a	b-t	c-r	a-r	b-r
$\phi(cat)$	λ^2	λ^3	λ^2	0	0	0	0	0
$\phi(car)$	λ^2	0	0	0	0	λ^3	λ^2	0
$\phi(bat)$	0	0	λ^2	λ^2	λ^3	0	0	0
$\phi(bar)$	0	0	0	λ^2	0	0	λ^2	λ^3

=> $K(car, cat) = \lambda^4$, efficient computation via dynamic programming.

=> Fisher Kernels [Jaakkola & Haussler, 1998]

Computing String Kernel (I)

Definitions:

- Σ^n: sequences of length n over alphabet Σ
- $\vec{i} = (i_1, \ldots, i_n)$: index sequence (sorted)
- $s(\vec{i})$: substring operator
- $r(\vec{i}) = i_n - i_1 + 1$: range of index sequence

Kernel: Average range of common subsequences of length n

$$K_n(s, t) = \sum_{u \in \Sigma^n} \sum_{\vec{i}; u = s(\vec{i})} \sum_{\vec{j}; u = s(\vec{j})} \lambda^{i_n + j_n - i_1 - j_1 + 2}$$

Auxiliary Function: Average range to end of sequence of common subsequences of length n

$$K_d'(s, t) = \sum_{u \in \Sigma^n} \sum_{\vec{i}; u = s(\vec{i})} \sum_{\vec{j}; u = s(\vec{j})} \lambda^{|s| + |t| - i_1 - j_1 + 2}$$

Computing String Kernel (II)

Kernel:

$$K_n(s,t) = 0 \qquad if(min(s,t) < n)$$

$$K_n(sx,t) = K_n(s,t) + \sum_{j;t_j = x} K'_{n-1}(s, t[1 \ldots j-1])\lambda^2$$

Auxiliary:

$$K'_0(s,t) = 1$$

$$K'_d(s,t) = 0 \qquad if(min(s,t) < d)$$

$$K'_d(sx,t) = \lambda K'_d(s,t) + \sum_{j;t_j = x} K'_{d-1}(s, t[1 \ldots j-1])\lambda^{|t|-j+2}$$