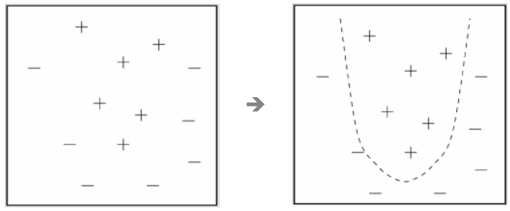
Kernels

CS6780 – Advanced Machine Learning
Spring 2019

Thorsten Joachims
Cornell University

Reading: Murphy 14.1, 14.2, 14.4
Schoelkopf/Smola Chapter 7.4, 7.6, 7.8

Non-Linear Problems

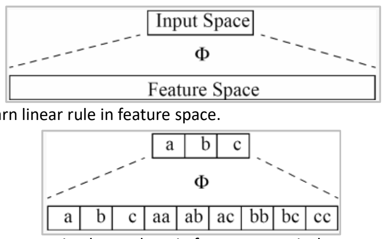


Problem:
- some tasks have non-linear structure
- no hyperplane is sufficiently accurate

How can SVMs learn non-linear classification rules?

Extending the Hypothesis Space

Idea: add more features



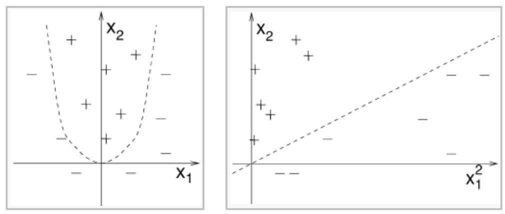
➔ Learn linear rule in feature space.

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: $\Phi(\vec{x}) = (x_1^2, x_2^2, x_1, x_2, x_1 x_2, 1)$ (6 attributes)



Dual SVM Optimization Problem

- Primal Optimization Problem

minimize: $P(\vec{w}, b, \vec{\xi}) = \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i=1}^{n}\xi_i$

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

$\Phi(\vec{x})$

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

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!
  - Example: Polynomials of degree p over N attributes in input space lead to $O(N^p)$ attributes in feature space!
- Solution:
  - The dual OP depends only on inner products
  - ➔ Kernel Functions $K(\vec{a}, \vec{b}) = \Phi(\vec{a}) \cdot \Phi(\vec{b})$
- Example:
  - For $\Phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, 1)$ 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:

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

$$\text{subject to: } \sum_{i=1}^{n} y_i \alpha_i = 0, \quad \forall_{i=1}^{n}: 0 \le \alpha_i \le C$$

- Classification:

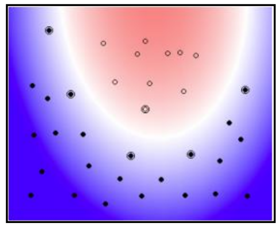
$$h(\vec{x}) = sign\left(\left[\sum_{i=1}^{n} \alpha_i y_i \Phi(\vec{x}_i)\right] \cdot \Phi(\vec{x}) + b\right) = sign\left(\sum_{i=1}^{n} \alpha_i y_i K(\vec{x}_i, \vec{x}) + b\right)$$

- 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: $K(\vec{a}, \vec{b}) = \exp\left(-\gamma[\vec{a} - \vec{b}]^2\right)$
  - Sigmoid: $K(\vec{a}, \vec{b}) = \tanh(\gamma[\vec{a} \cdot \vec{b}] + c)$

Examples of Kernels

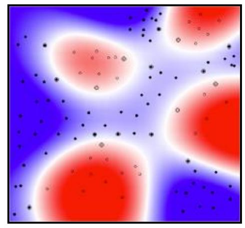
Polynomial

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



Radial Basis Function

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



What is a Valid Kernel?

Definition [simplified]: Let *X* be a nonempty set. A function is a valid kernel in *X* if for all *n* and all $x_1, \ldots, x_n \in X$ it produces a Gram matrix

$$G_{ij} = K(x_i, x_j)$$

that is symmetric

$$G = G^T$$

and positive semi-definite

$$\forall \vec{\alpha}: \vec{\alpha}^T G \vec{\alpha} \ge 0$$

How to Construct Valid Kernels

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

$$K(x,z) = \lambda K_1(x,z) + (1-\lambda) K_2(x,z)$$
$$K(x,z) = \alpha K_1(x,z)$$
$$K(x,z) = K_1(x,z) K_2(x,z)$$
$$K(x,z) = f(x) f(z)$$
$$K(x,z) = K_3(\phi(x),\phi(z))$$
$$K(x,z) = x^T K z$$

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!

Kernels for Discrete and Structured Data

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

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

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

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

Properties of SVMs with Kernels

- Expressiveness
  - SVMs with Kernel can represent any boolean function (for appropriate choice of kernel)
  - SVMs with 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 (but quadratic in number of examples without additional approximations)
- Design decisions
  - Kernel type and parameters
  - Value of C

What else can be “Kernelized”?

- Multi-class SVM
  - [Schoelkopf/Smola Book, Section 7.6]
- Regression SVM
  - [Schoelkopf/Smola Book, Section 1.6]
- Kernel PCA
  - [Schoelkopf/Smola Book, Section 13]
- Gaussian Processes
  - [Schoelkopf/Smola Book, Section 16]

… and any other method that can be written in terms of inner products.