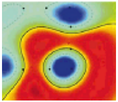
Lecture 11: Support Vector Machines

Machine Learning 1



TU Berlin – WiSe 2016/2017

Hyperplane $y = \operatorname{sgn}(\mathbf{w} \cdot \Phi(x) + b)$ in $\mathcal{F}$

$$\begin{array}{ll} \min & \|\mathbf{w}\|^2 \\ \text{subject to} & y_i \cdot [(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b] \geq 1 \quad \text{for } i = 1 \ldots N \end{array}$$

(i.e. training data separated correctly, otherwise introduce slack variables).

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i \cdot ((\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b) - 1\right).$$

obtain unique α_i by QP (no local minima!): dual problem

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \quad \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0,$$

$$\text{i.e.} \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \quad \text{and} \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i).$$

Substitute both into L to get the *dual problem*

Hyperplane in $\mathcal{F}$ with slack variables: SVM

$$\min \quad \|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i{}^p$$

$$\text{subject to} \quad y_i \cdot [(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b] \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \quad \text{for } i = 1 \ldots N$$

(introduce slack variables if training data not separated correctly)

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i \cdot ((\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b) - 1\right).$$

obtain unique α_i by QP (no local minima!): dual problem

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \quad \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0,$$

$$\text{i.e.} \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \quad \text{and} \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i).$$

Substitute both into L to get the *dual problem*

Dual Problem

$$\text{maximize} \qquad W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

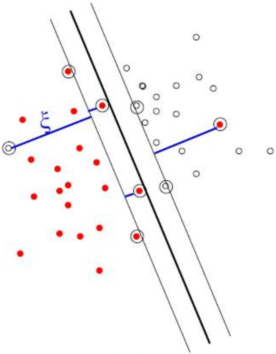
$$\text{subject to} \qquad C \geq \alpha_i \geq 0, \quad i = 1, \ldots, N, \quad \text{and} \quad \sum_{i=1}^{N} \alpha_i y_i = 0.$$

Note: solution determined by training examples (SVs) on /in the margin. Remark: duality gap.

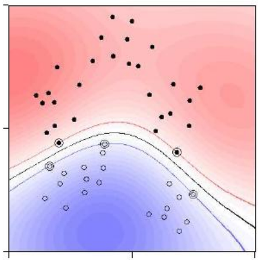
$$y_i \cdot [(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b] > 1 \qquad \Longrightarrow \alpha_i = 0 \longrightarrow \quad \mathbf{x}_i \text{ irrelevant or}$$

$$y_i \cdot [(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b] = 1 \quad (\mathit{on} \text{ /in margin}) \longrightarrow \quad \mathbf{x}_i \text{ Support Vector}$$

A Toy Example: $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2)$



linear SV with slack variables



nonlinear SVM, Domain: $[-1, 1]^2$

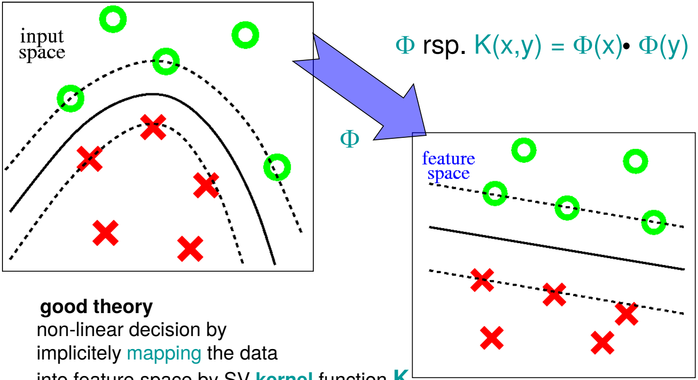
Kernel Trick

- Saddle Point: $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)$.
- Hyperplane in $\mathcal{F}$: $y = \mathrm{sgn}\left(\mathbf{w} \cdot \Phi(x) + b\right)$
- putting things together “kernel trick”

$$\begin{aligned} f(\mathbf{x}) &= \mathrm{sgn}\left(\mathbf{w} \cdot \Phi(\mathbf{x}) + b\right) \\ &= \mathrm{sgn}\left(\sum_{i=1}^{N} \alpha_i y_i \, \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) + b\right) \\ &= \mathrm{sgn}\left(\sum_{i \in \#\mathrm{SVs}} \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) + b\right) \qquad \textbf{sparse!} \end{aligned}$$

- trick: $k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$, i.e. **never use Φ: only k!!!**

Support Vector Machines in a nutshell



Φ rsp. $K(x,y) = \Phi(x) \bullet \Phi(y)$

good theory
non-linear decision by
implicitely mapping the data
into feature space by SV **kernel** function **K**

Kernels ...

- kernels hold key to learning problem.
- chosing kernels ...
  - Mercer condition (ℓ_2 integrability & positivity)
  - kernel reflects prior (Smola, Schölkopf & Müller 98, Girosi 98)
  - approximating LOO bounds give good model selection results (Tsuda et al. 2001, Vapnik & Chapelle 2000)
- So: **engineer** an appropriate kernel from prior knowledge! (Jaakola and Haussler 1998, Watkins 2000, Zien et al 2000, recently a large body of interesting work)
- And: use **careful** model selection to find appropriate kernel parameters, i.e. chose appropriate degree of polynomial or bandwidth of Gaussian kernel

Digestion: Use of kernels

- Question: What makes kernel methods (e.g. SVM) perform well?
- Answer:
  - In the first place: a good idea/theory.
  - But also: The kernel
- Using kernels, we work explicitly in extremely high dimensional spaces (RKHS) with interesting features for themselves (depending on the kernel) [SSM et al. 98]
- Common choices: Gaussian kernel $\exp(-\|\boldsymbol{x}-\boldsymbol{y}\|^2/c)$ or polynomial kernel $(\boldsymbol{x}\cdot\boldsymbol{y})^d$.
- Almost any linear algorithm can be transformed to feature space. [SSM et al. 98]
- With suitable regularization it outperforms its linear counterpart. [Mika et al. 02]

[Zien et al. 00, Tsuda et al. 02, Sonnenburg et al. 05]

- The kernel can be adopted to specific tasks, e.g. using prior knowledge

Kernels for graphs, trees, strings etc.

Digestion

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{d\left(\log\frac{2N}{d}+1\right)-\log(\eta/4)}{N}}$$

$$K(\mathbf{x},\mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$$

SRM (VC theory)

feature spaces & RKHS

kernel trick

mathematical programs

SV classification & regression

feature extraction $\longrightarrow$ kernelize it!

kernel choice

Boosting
Large margins
Bounds, sparsity

APPLICATIONS

OCR, Bioinformatics **(DNA, drug discovery)**
categorization, biomedical data (BCI, EEG),
benchmarks, time-series analysis, power,
quarks, object recognition, textmining