SIGIR 2003 Tutorial

Support Vector and Kernel Methods

Thorsten Joachims

Cornell University
Computer Science Department

tj@cs.cornell.edu
http://www.joachims.org
Linear Classifiers

Rules of the Form: weight vector $\vec{w}$, threshold $b$

$$h(\hat{x}) = \text{sign} \left[ \sum_{i=1}^{N} \hat{w}_i \hat{x}_i + b \right] = \begin{cases} 1 & \text{if } \sum_{i=1}^{N} \hat{w}_i \hat{x}_i + b > 0 \\ -1 & \text{else} \end{cases}$$

Geometric Interpretation (Hyperplane):
Optimal Hyperplane (SVM Type 1)

**Assumption:** The training examples are linearly separable.
Maximizing the Margin

The hyperplane with maximum margin

(<~ (roughly, see later) ~>)

The hypothesis space with minimal VC-dimension according to SRM

**Support Vectors:** Examples with minimal distance.
Example: Optimal Hyperplane vs. Perceptron

Train on 1000 pos / 1000 neg examples for “acq” (Reuters-21578).
Non-Separable Training Samples

- For some training samples there is no separating hyperplane!
- Complete separation is suboptimal for many training samples!

=> minimize trade-off between margin and training error.
**Soft-Margin Separation**

**Idea:** Maximize margin and minimize training error simultaneously.

**Hard Margin:**
minimize \( P(\hat{w}, b) = \frac{1}{2} \hat{w} \cdot \hat{w} \)

s. t. \( y_i[\hat{w} \cdot \hat{x}_i + b] \geq 1 \)

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

s. t. \( y_i[\hat{w} \cdot \hat{x}_i + b] \geq 1 - \xi_i \) and \( \xi_i \geq 0 \)
Controlling Soft-Margin Separation

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

s. t. \( y_i [\hat{w} \cdot \hat{x}_i + b] \geq 1 - \xi_i \) and \( \xi_i \geq 0 \)

- \( \sum \xi_i \) is an upper bound on the number of training errors.
- \( C \) is a parameter that controls trade-off between margin and error.
Example Reuters “acq”: Varying C

Observation: Typically no local optima, but not necessarily...
Properties of the Soft-Margin Dual OP

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

s. t. $\sum_{i=1}^{n} \alpha_i y_i = 0$  
$und$  
$0 \leq \alpha_i \leq C$

- typically single solution (i.e. $\langle \hat{w}, b \rangle$ is unique)
- one factor $\alpha_i$ for each training example
  - “influence” of single training example limited by $C$
  - $0 < \alpha_i < C \iff$ SV with $\xi_i = 0$
  - $\alpha_i = C \iff$ SV with $\xi_i > 0$
  - $\alpha_i = 0$ else
- based exclusively on inner product between training examples
Primal <=> Dual

**Theorem:** The primal OP and the dual OP have the same solution. Given the solution \( \alpha_i^\circ \) of the dual OP,

\[
\vec{w}^\circ = \sum_{i=1}^{n} \alpha_i^\circ y_i \vec{x}_i \quad b^\circ = \frac{1}{2}(\vec{w}_0 \cdot \vec{x}^{pos} + \vec{w}_0 \cdot \vec{x}^{neg})
\]

is the solution of the primal OP.

**Theorem:** For any set of feasible points \( P(\vec{w}, b) \geq D(\vec{\alpha}) \).

\( \Rightarrow \) two alternative ways to represent the learning result
- weight vector and threshold \( \langle \vec{w}, b \rangle \)
- vector of “influences” \( \alpha_1, \ldots, \alpha_n \)
Non-Linear Problems

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

How can SVMs learn non-linear classification rules?
Example

Input Space: $\mathbf{x} = (x_1, x_2)$ (2 Attributes)

Feature Space: $\Phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1)$ (6 Attributes)
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.
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., 1992] The dual OP need only inner products $\Rightarrow$ Kernel Functions

$$K(\hat{x}_i, \hat{x}_j) = \Phi(\hat{x}_i) \cdot \Phi(\hat{x}_j)$$

**Example:** For $\Phi(\hat{x}) = (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1)$ calculating

$$K(\hat{x}_i, \hat{x}_j) = [\hat{x}_i \cdot \hat{x}_j + 1]^2 = \Phi(\hat{x}_i) \cdot \Phi(\hat{x}_j)$$

gives inner product in feature space.

**We do not need to represent the feature space explicitly!**
SVM with Kernels

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

\[
s. t. \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \quad und \quad 0 \leq \alpha_i \leq C
\]

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

New hypotheses spaces through new Kernels:
Linear: \( K(\hat{x}_i, \hat{x}_j) = \hat{x}_i \cdot \hat{x}_j \)
Polynomial: \( K(\hat{x}_i, \hat{x}_j) = [\hat{x}_i \cdot \hat{x}_j + 1]^d \)
Radial Basis Functions: \( K(\hat{x}_i, \hat{x}_j) = \exp(-|\hat{x}_i - \hat{x}_j|^2 / \sigma^2) \)
Sigmoid: \( K(\hat{x}_i, \hat{x}_j) = \tanh(\gamma(\hat{x}_i - \hat{x}_j) + c) \)
Example: SVM with Polynomial of Degree 2

Kernel: $K(\hat{x}_i, \hat{x}_j) = [\hat{x}_i \cdot \hat{x}_j + 1]^2$

plot by Bell SVM applet
Example: SVM with RBF-Kernel

Kernel: \( K(\tilde{x}_i, \tilde{x}_j) = \exp(-|\tilde{x}_i - \tilde{x}_j|^2 / \sigma^2) \)
Two Reasons for Using a Kernel

(1) Turn a linear learner into a non-linear learner
   (e.g. RBF, polynomial, sigmoid)

(2) Make non-vectorial data accessible to learner
   (e.g. string kernels for sequences)
Summary
What is an SVM?

Given:
- Training examples \((\hat{x}_1, y_1), \ldots, (\hat{x}_n, y_n)\) \(\hat{x}_i \in \mathbb{R}^N\) \(y_i \in \{1, -1\}\)
- Hypothesis space according to kernel \(K(\hat{x}_i, \hat{x}_j)\)
- Parameter \(C\) for trading-off training error and margin size

Training:
- Finds hyperplane in feature space generated by kernel.
- The hyperplane has maximum margin in feature space with minimal training error (upper bound \(\sum \xi_i\)) given \(C\).
- The result of training are \(\alpha_1, \ldots, \alpha_n\). They determine \(\langle \hat{w}, b \rangle\).

Classification: For new example \(h(\hat{x}) = \text{sign}\left(\sum_{x_i \in SV} \alpha_i y_i K(\hat{x}_i, \hat{x}) + b\right)\)
Part 2: How to use an SVM effectively and efficiently?

- normalization of the input vectors
  - selecting C
- handling unbalanced datasets
  - selecting a kernel
- multi-class classification
- selecting a training algorithm
How to Assign Feature Values?

Things to take into consideration:

• importance of feature is monotonic in its absolute value
  • the larger the absolute value, the more influence the feature gets
  • typical problem: number of doors [0-5], price [0-100000]
  • want relevant features large / irrelevant features low (e.g. IDF)

• normalization to make features equally important
  • by mean and variance: $x_{norm} = \frac{x - \text{mean}(X)}{\sqrt{\text{var}(X)}}$
  • by other distribution

• normalization to bring feature vectors onto the same scale
  • directional data: text classification
  • by normalizing the length of the vector $\hat{x}_{norm} = \frac{\hat{x}}{||\hat{x}||}$ according to some norm
  • changes whether a problem is (linearly) separable or not

• scale all vectors to a length that allows numerically stable training
Selecting a Kernel

Things to take into consideration:
• kernel can be thought of as a similarity measure
  • examples in the same class should have high kernel value
  • examples in different classes should have low kernel value
• ideal kernel: equivalence relation \( K(\hat{x}_i, \hat{x}_j) = \text{sign}(y_i y_j) \)
• normalization also applies to kernel
  • relative weight for implicit features
  • normalize per example for directional data

\[
K(\hat{x}_i, \hat{x}_j) = \frac{K(\hat{x}_i, \hat{x}_j)}{\sqrt{K(\hat{x}_i, \hat{x}_i) \cdot K(\hat{x}_j, \hat{x}_j)}}
\]

• potential problems with large numbers, for example polynomial kernel \( K(\hat{x}_i, \hat{x}_j) = [\hat{x}_i \cdot \hat{x}_j + 1]^d \) for large \( d \)
Selecting Regularization Parameter $C$

Common Method
- a reasonable starting point and/or default value is $C_{\text{def}} = \frac{1}{\sum K(\hat{x}_i, x_i)}$
- search for $C$ on a log-scale, for example
  \[ C \in [10^{-4}C_{\text{def}}, \ldots, 10^4C_{\text{def}}] \]

- selection via cross-validation or via approximation of leave-one-out

Note
- optimal value of $C$ scales with the feature values
Selecting Kernel Parameters

Problem
- results often very sensitive to kernel parameters (e.g. variance $\gamma$ in RBF kernel)
- need to simultaneously optimize C, since optimal C typically depends on kernel parameters

Common Method
- search for combination of parameters via exhaustive search
- selection of kernel parameters typically via cross-validation

Advanced Approach
- avoiding exhaustive search for improved search efficiency [Chapelle et al, 2002]
Handling Multi-Class / Multi-Label Problems

Standard classification SVM addresses binary problems \( y \in \{1, -1\} \)

**Multi-class classification:** \( y \in \{1, \ldots, k\} \)
- one-against-rest decomposition into \( k \) binary problems
  - learn one binary SVM \( h^{(i)} \) per class with \( y^{(i)} = \begin{cases} 1 & \text{if}(y = i) \\ -1 & \text{else} \end{cases} \)
  - assign new example to \( y = \arg\max[h^{(i)}(\hat{x})] \)
- pairwise decomposition into \( k(k-1) \) binary problems
  - learn one binary SVM \( h^{(i)} \) per class pair \( y^{(i,j)} = \begin{cases} 1 & \text{if}(y = i) \\ -1 & \text{if}(y = j) \end{cases} \)
  - assign new example by majority vote
  - reducing number of classifications [Platt et al., 2000]
- multi-class SVM [Weston & Watkins, 1998]
- multi-class SVM via ranking [Crammer & Singer, 2001]