SIGIR 2003 Tutorial

Support Vector and Kernel Methods

Thorsten Joachims

Cornell University
Computer Science Department

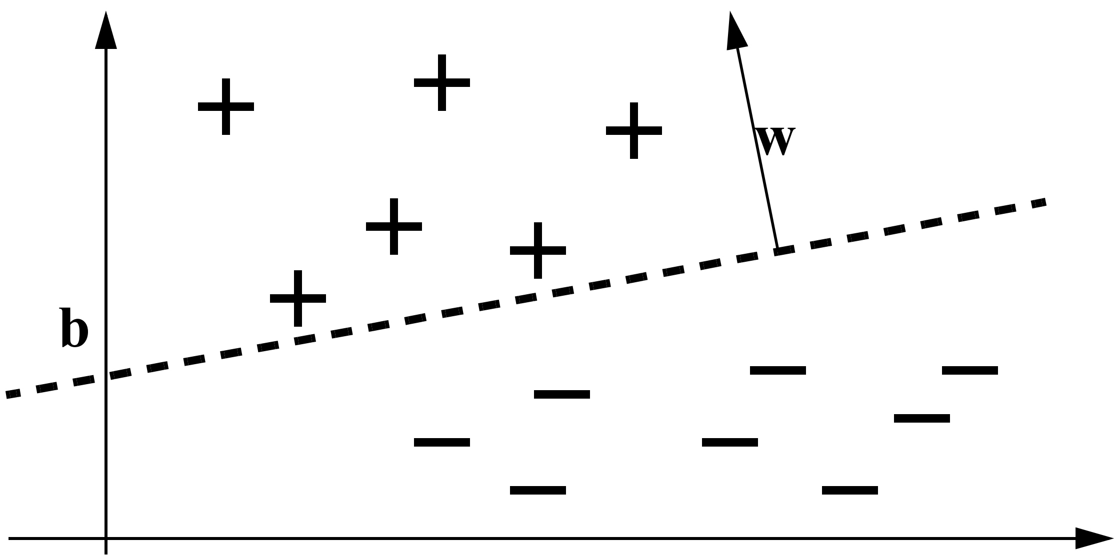
tj@cs.cornell.edu
http://www.joachims.org

Linear Classifiers

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

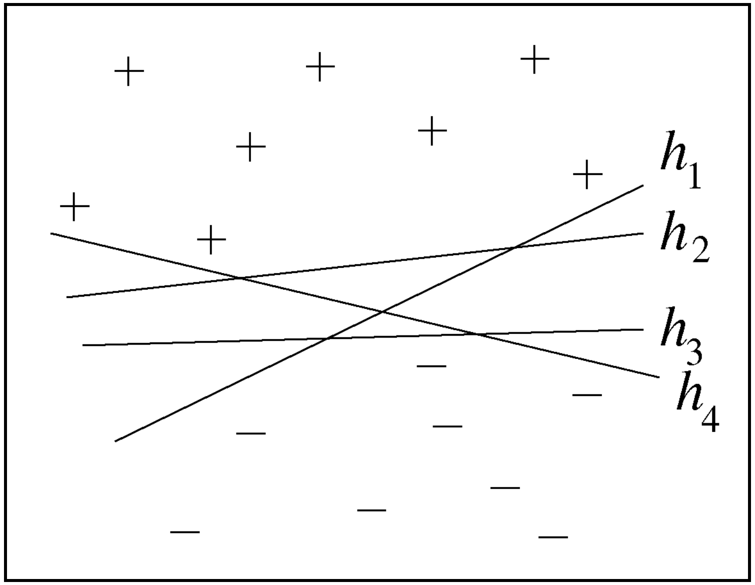
$$h(\vec{x}) = sign\left[\sum_{i=1}^{N} \vec{w}_i \vec{x}_i + b\right] = \begin{cases} 1 & if \sum_{i=1}^{N} \vec{w}_i \vec{x}_i + b > 0 \\ -1 & else \end{cases}$$

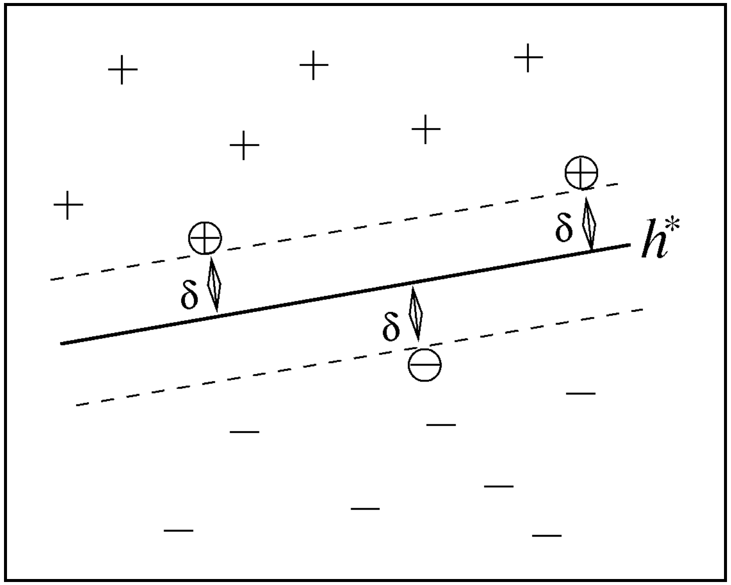
Geometric Interpretation (Hyperplane):



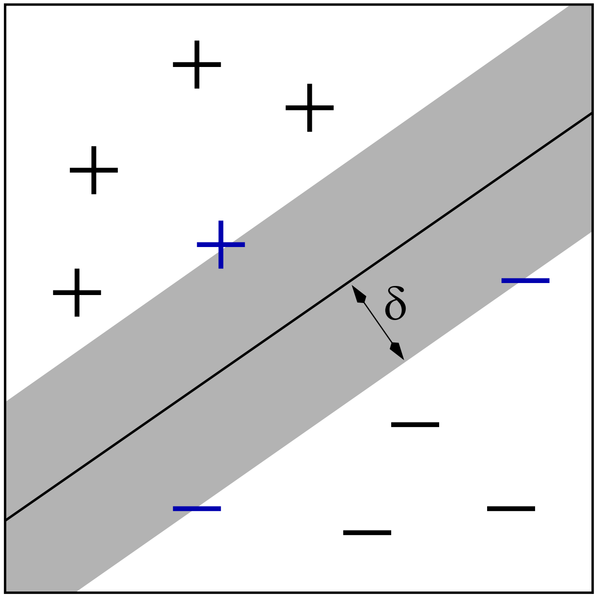
Optimal Hyperplane (SVM Type 1)

Assumption: The training examples are linearly separable.





Maximizing the Margin



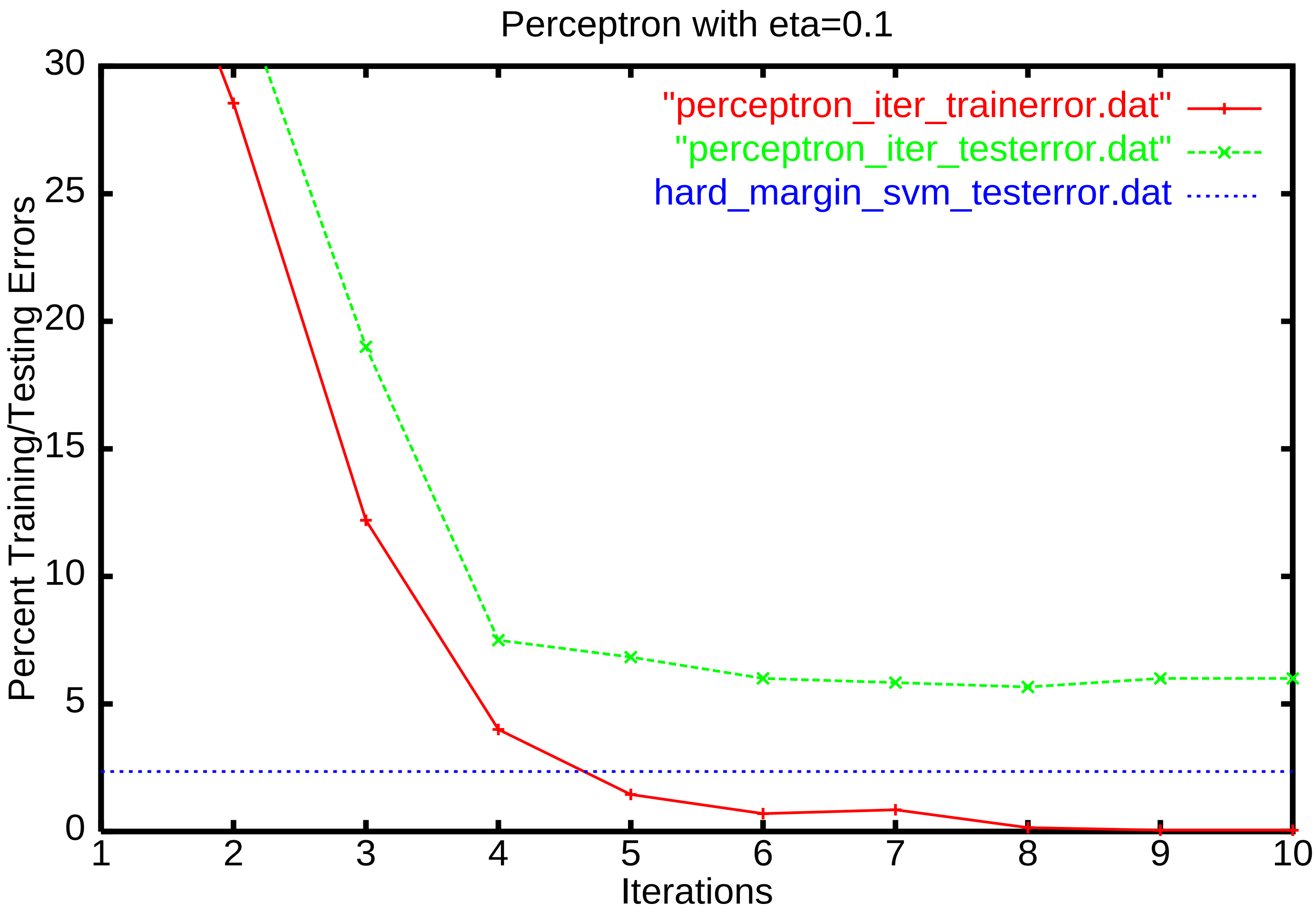
The hyperplane with maximum margin

<~ (roughly, see later) ~>

The hypothesis space with minimal VC-dimension according to SRM

Support Vectors: Examples with minimal distance.

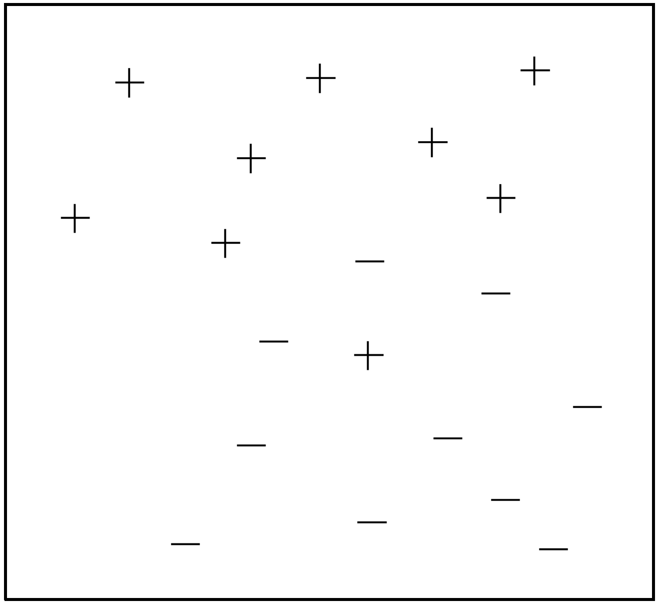
Example: Optimal Hyperplane vs. Perceptron

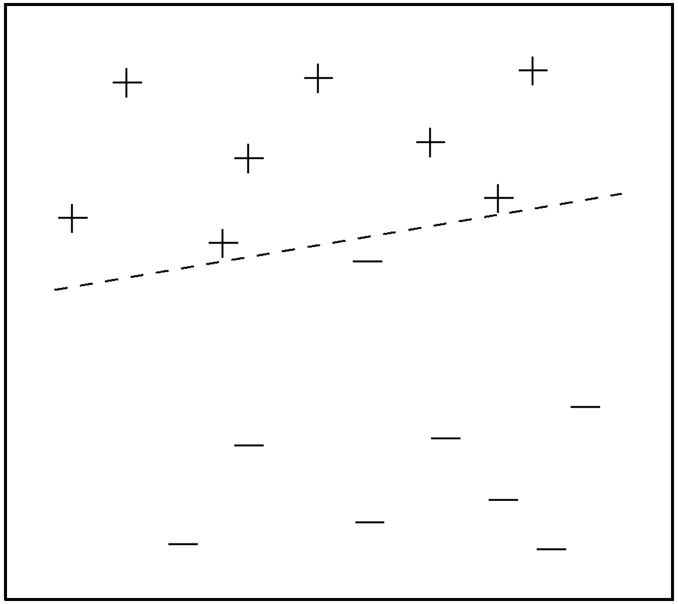


Train on 1000 pos / 1000 neg examples for “acq” (Reuters-21578).

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 simultanously.

Hard Margin:

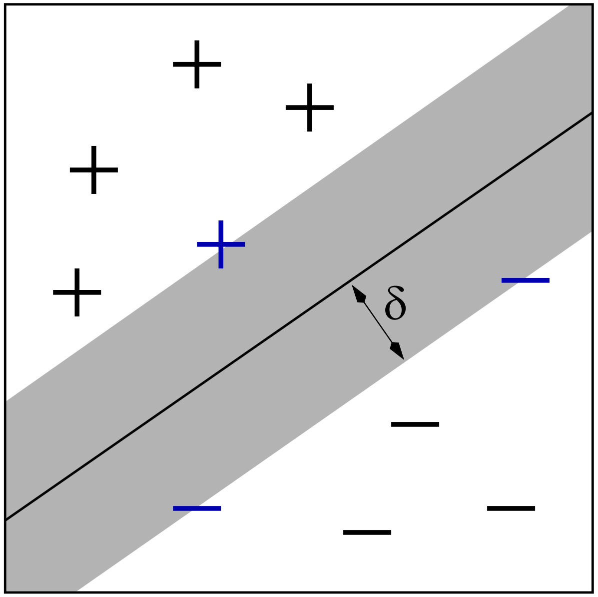
minimize $P(\vec{w}, b) = \frac{1}{2}\vec{w} \cdot \vec{w}$

s. t. $y_i[\vec{w} \cdot \vec{x}_i + b] \ge 1$

Soft Margin:

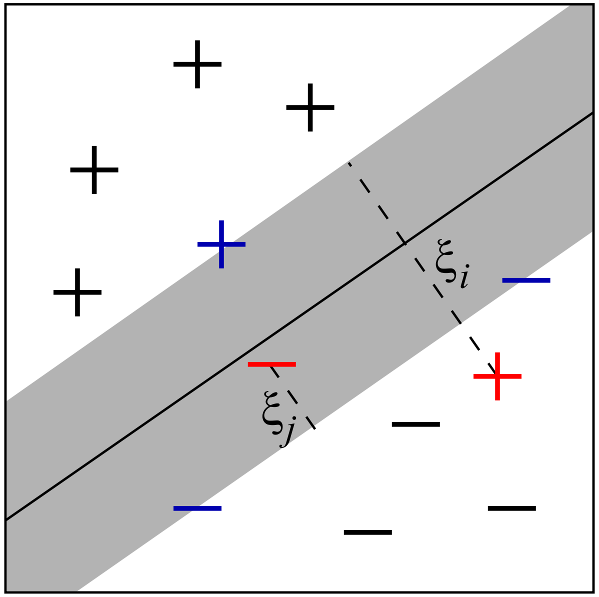
minimize $P(\vec{w}, b, \vec{\xi}) = \frac{1}{2}\vec{w} \cdot \vec{w} + C\sum_{i=1}^{n} \xi_i$

s. t. $y_i[\vec{w} \cdot \vec{x}_i + b] \ge 1 - \xi_i$ *and* $\xi_i \ge 0$



Hard Margin
(separable)

Soft Margin
(training error)

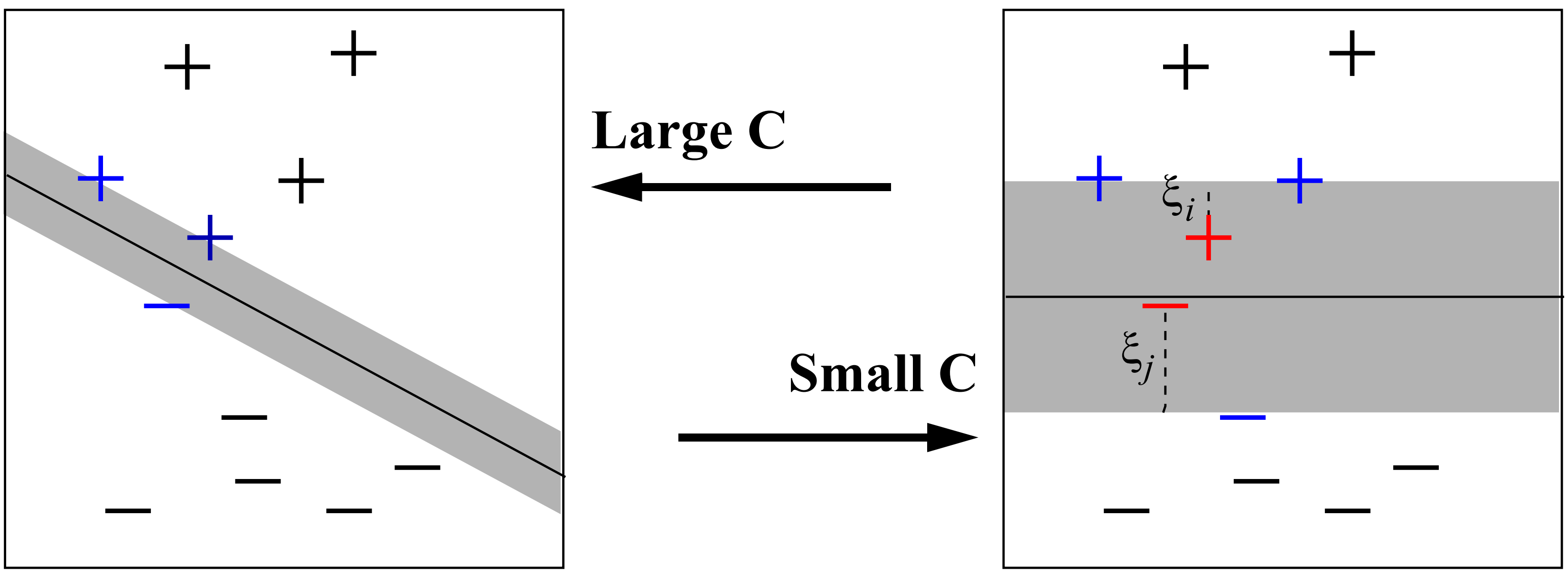


Controlling Soft-Margin Separation

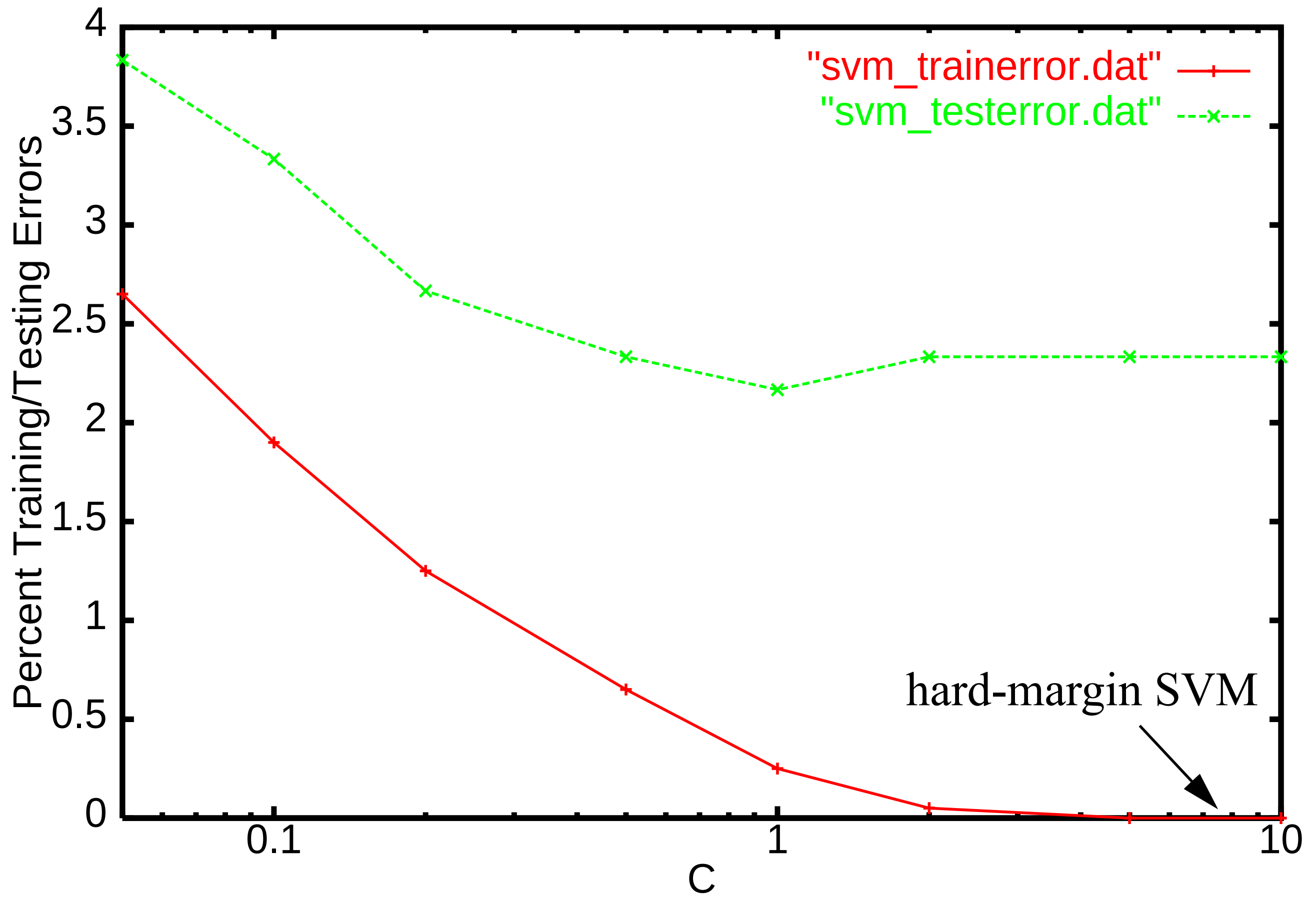
Soft Margin: minimize $P(\vec{w}, b, \vec{\xi}) = \frac{1}{2}\vec{w} \cdot \vec{w} + C \sum_{i=1}^{n} \xi_i$

s. t. $y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1 - \xi_i$ *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



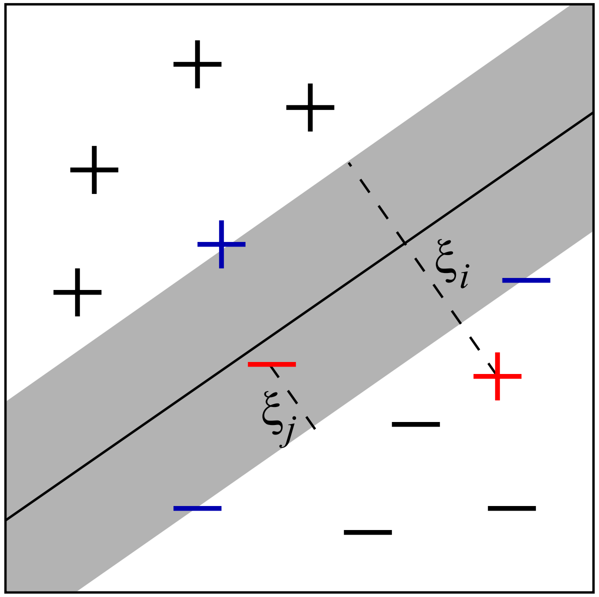
Observation: Typically no local optima, but not necessarily...

Properties of the Soft-Margin Dual OP

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

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

- typically single solution (i. e. $\langle \vec{w}, b \rangle$ is unique)
- one factor α_i for each training example
  - "influence" of single training example limited by C
  - $0 < \alpha_i < C$ <=> SV with $\xi_i = 0$
  - $\alpha_i = C$ <=> 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 $\alpha_i°$ of the dual OP,

$$\vec{w}° = \sum_{i=1}^{n} \alpha_i° y_i \vec{x}_i \qquad b° = \frac{1}{2}(\vec{w}_0 \cdot \vec{x}^{pos} + \vec{w}_0 \cdot \vec{x}^{neg})$$

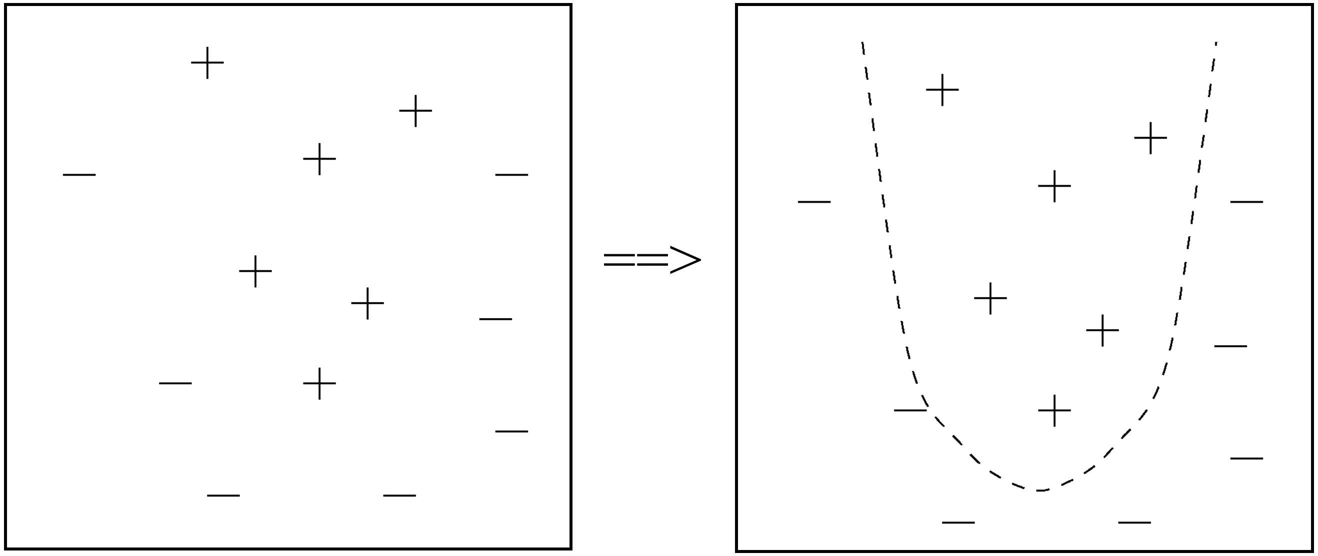
is the solution of the primal OP.

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

=> 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:

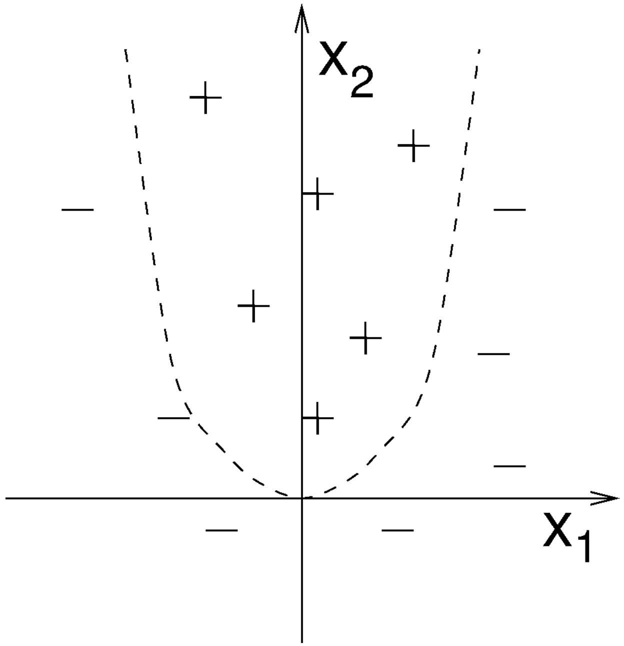
- some tasks have non-linear structure
- no hyperplane is sufficiently accurate

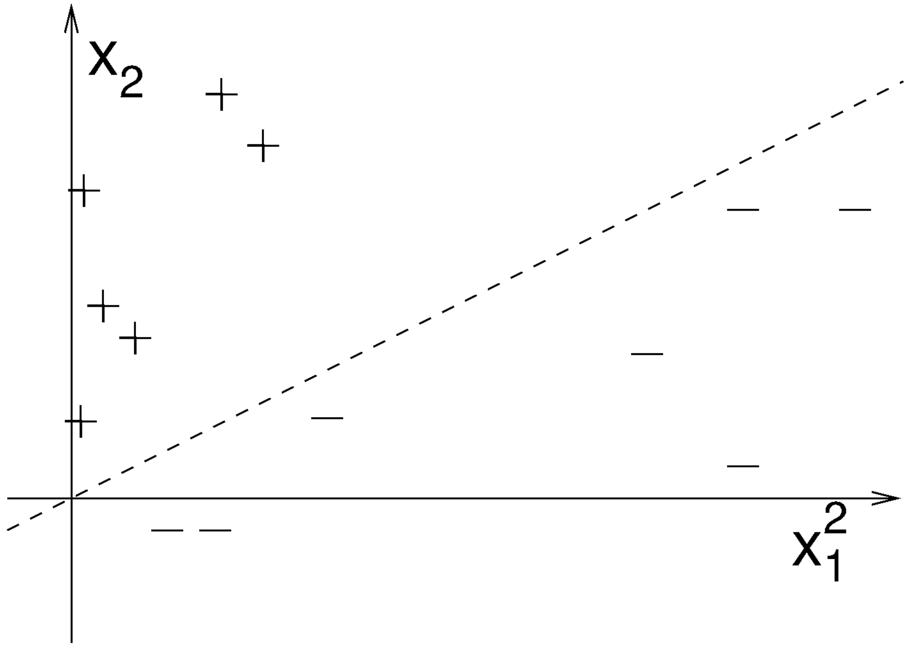
How can SVMs learn non-linear classification rules?

Example

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

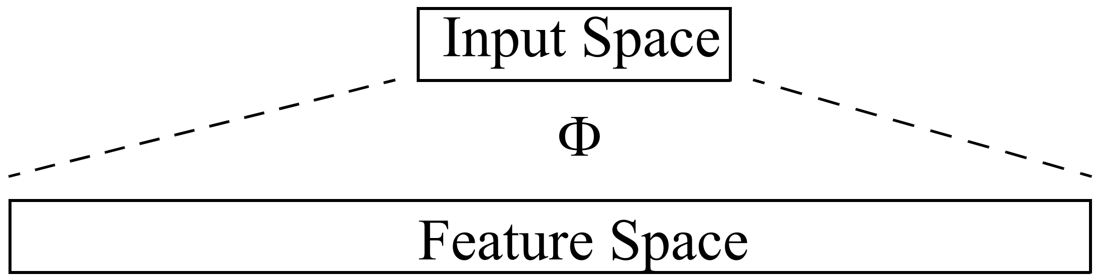
Feature Space: $\Phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1)$ (6 Attributes)





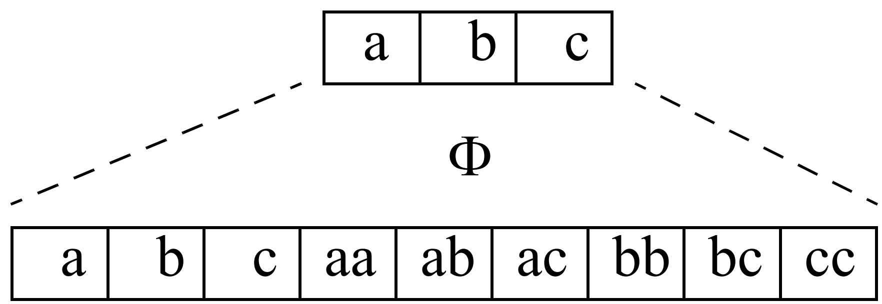
Extending the Hypothesis Space

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 => Kernel Functions

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

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

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

gives inner product in feature space.

We do not need to represent the feature space explicitly!

SVM with Kernels

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

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

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: $K(\vec{x}_i, \vec{x}_j) = [\vec{x}_i \cdot \vec{x}_j + 1]^d$

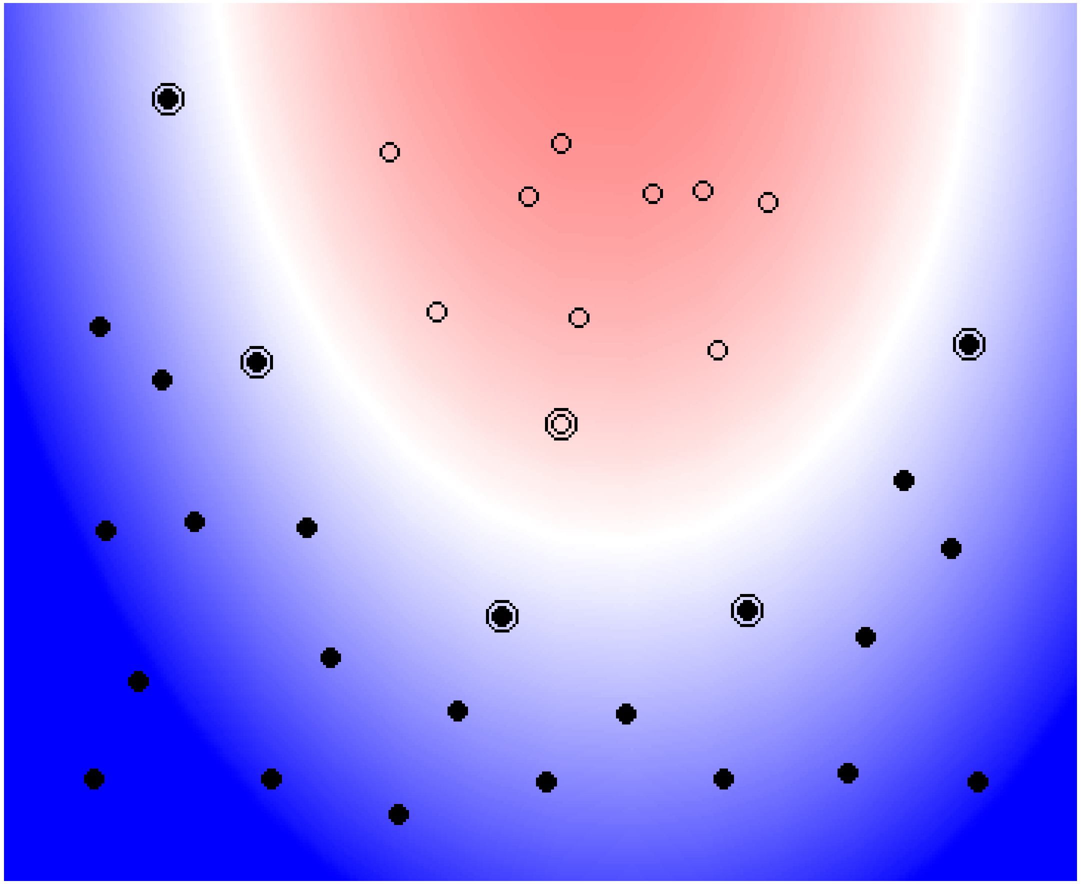
Radial Basis Functions: $K(\vec{x}_i, \vec{x}_j) = \exp(-|\vec{x}_i - \vec{x}_j|^2 / \sigma^2)$

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

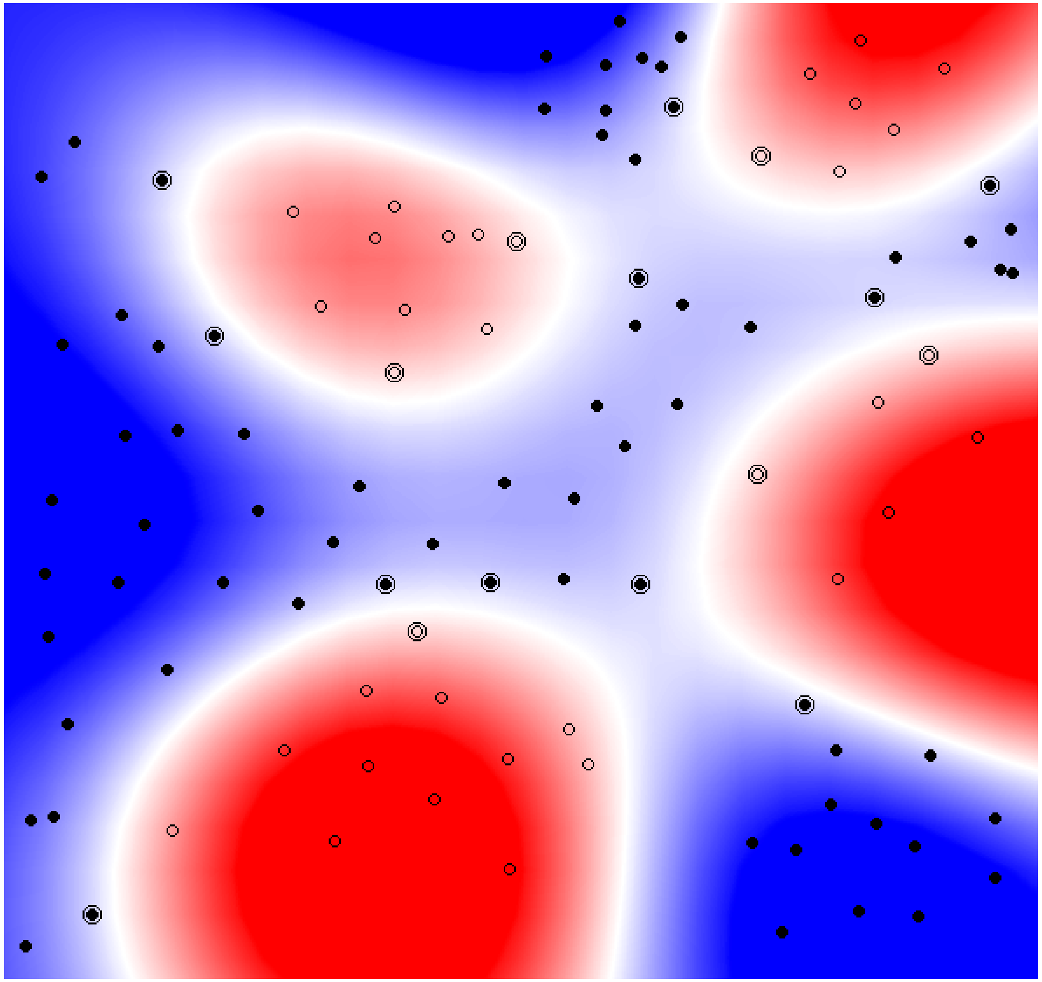
plot by Bell SVM applet



Example: SVM with RBF-Kernel

Kernel: $K(\vec{x}_i, \vec{x}_j) = \exp(-|\vec{x}_i - \vec{x}_j|^2 / \sigma^2)$

plot by Bell SVM applet



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 $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$ $\quad \vec{x}_i \in \Re^N \quad y_i \in \{1, -1\}$
- Hypothesis space according to kernel $K(\vec{x}_i, \vec{x}_j)$
- 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 $\langle \vec{w}, b \rangle$.

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

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: $x_{norm} = \frac{x - mean(X)}{\sqrt{var(X)}}$
  - by other distribution
- normalization to bring feature vectors onto the same scale
  - directional data: text classification
  - by normalizing the length of the vector $\vec{x}_{norm} = \frac{\vec{x}}{\|\vec{x}\|}$ 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 $K(\vec{x}_i,\vec{x}_j) = sign(y_i y_j)$
- normalization also applies to kernel
  - relative weight for implicit features
  - normalize per example for directional data

$$K(\vec{x}_i,\vec{x}_j) = \frac{K(\vec{x}_i,\vec{x}_j)}{\sqrt{K(\vec{x}_i,\vec{x}_i)}\sqrt{K(\vec{x}_j,\vec{x}_j)}}$$

  - potential problems with large numbers, for example polynomial kernel $K(\vec{x}_i,\vec{x}_j) = [\vec{x}_i \cdot \vec{x}_j + 1]^d$ for large d

Selecting Regularization Parameter C

Common Method

- a reasonable starting point and/or default value is $C_{def} = \frac{1}{\sum K(\vec{x}_i, \vec{x}_i)}$
- search for C on a log-scale, for example

$$C \in [10^{-4} C_{def}, \ldots, 10^{4} C_{def}]$$

- selection via cross-validation or via approximation of leave-one-out [Jaakkola&Haussler,1999][Vapnik&Chapelle,2000][Joachims,2000]

Note

- optimal value of C scales with the feature values

Selecting Kernel Parameters

Problem

- results often very sensitive to kernel parameters (e.g. variance γ 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 $y^{(i)} = \begin{cases} 1 & if(y = i) \\ -1 & else \end{cases}$
  - assign new example to $y = \arg max[h^{(i)}(\vec{x})]$
- pairwise decomposition into $k(k-1)$ binary problems
  - learn one binary SVM $h^{(i)}$ per class pair $y^{(i,j)} = \begin{cases} 1 & if(y = i) \\ -1 & if(y = j) \end{cases}$
  - 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]