# L22: SVMs and kernel methods

**The non-separable case**

**Non-linear SVMs and kernel methods**

**A numerical example**

**Optimization techniques**

**SVM extensions**

**Discussion**

# The non-separable case

## The previous lecture focused on linearly separable problems

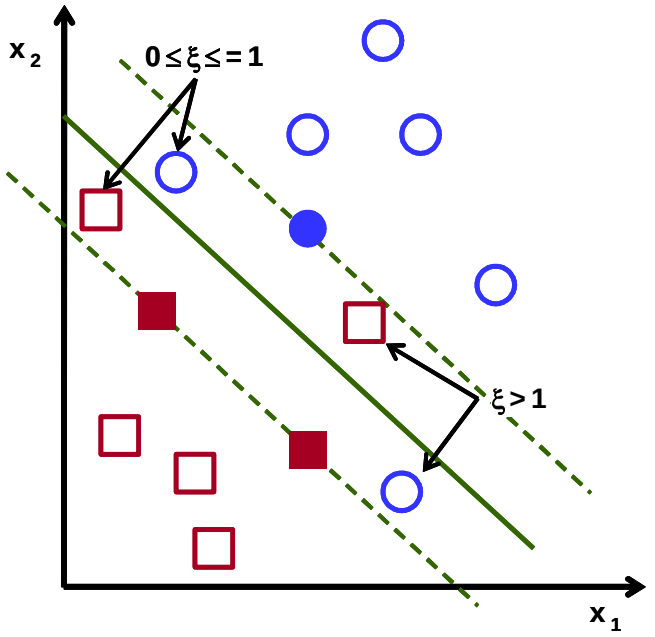
- In this lecture we will see how SVMs can be modified to handle datasets that are not linearly separable

## Solution

- The solution for the non-separable case is to introduce slack variables $\xi_i$ that relax the constraints of the canonical hyperplane equation

$$y_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i = 1..N$$

- The slack variables measure deviation from the ideal condition
  - For $0 \leq \xi \leq 1$, the data point falls on the right side of the separating hyperplane but within the region of maximum margin
  - For $\xi > 1$, the data point falls on the wrong side of the separating hyperplane




[Cherkassky and Mulier, 1998; Haykin, 1999]

# The non-separable case

**How does the optimization problem change with the introduction of slack variables?**

- Our goal is to find a hyperplane with minimum misclassification rate
- This may be achieved by minimizing the following objective function

$$\Theta(\xi) = \sum_{i=1}^{N} I(\xi_i - 1) \quad where \quad I(\xi_i) = \begin{cases} 0; & \xi \leq 0 \\ 1; & \xi > 0 \end{cases}$$

  - subject to the constraints on $\|w\|^2$ and the perceptron equation
- $\Theta(\xi)$ represents the total number of misclassified samples

- Unfortunately, minimization of $\Theta(\xi)$ is a difficult combinatorial problem (NP-complete) due to the non-linearity of the indicator function $I(\xi_i)$

[Cherkassky and Mulier, 1998]

- Instead, we approximate $\Theta(\xi)$ by

$$\Theta'(\xi) = \sum_{i=1}^{N} \xi_i$$

- which is an upper bound on the number of misclassifications, and minimize the joint objective function

$$J(w, \xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i$$

- subject to $\begin{cases} y_i(w^T x_i + b) \geq 1 - \xi_i & \forall i \\ \xi_i \geq 0 & \forall i \end{cases}$

## Interpretation of C

- Parameter C represents a trade-off between misclassification and capacity
  - Large values of C favor solutions with few misclassification errors
  - Small values of C denote a preference towards low-complexity solutions
- Therefore, this parameter can be viewed as a regularization parameter (recall ridge-regression in L17), the difference being that the minimization problem is now subject to constraints
- A suitable value for C is typically determined through cross-validation

## Solution

- Using a procedure similar to the one in L21, we can derive the dual problem as

$$L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

- subject to the constraints

$$\begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \quad i = 1 \dots N \end{cases}$$

## Comments

- Notice that neither the slack variables nor their associated Lagrange multipliers appear in the formulation of the dual problem
- Therefore, this represents the same optimization problem as the linearly separable case, with the exception that the constraints $\alpha_i \geq 0$ have been replaced by the more restrictive constraints $0 \leq \alpha_i \leq C$
  - The optimum solution for the weight vector remains the same:

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

  - and the bias can be found by choosing a training point for which $0 < \alpha_i < C$ ($\xi_i = 0$), and solving the KKT condition:

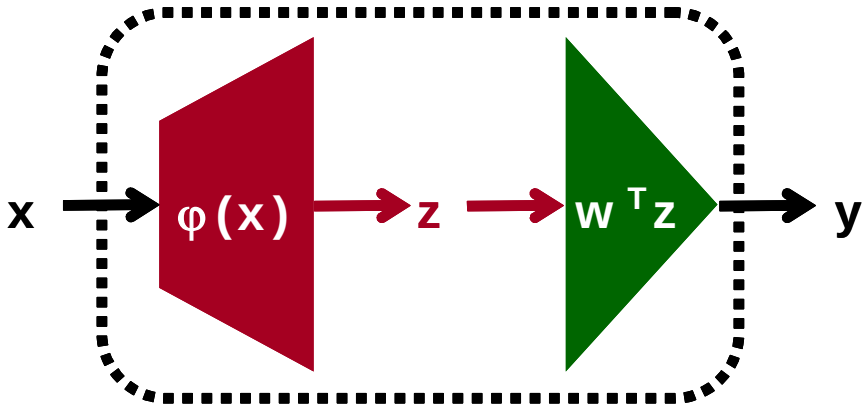
$$\alpha_i [y_i (w^T x_i + b) - 1 + \xi_i] = 0$$

[Cherkassky and Mulier, 1998; Haykin, 1999]

# Non-linear SVMs

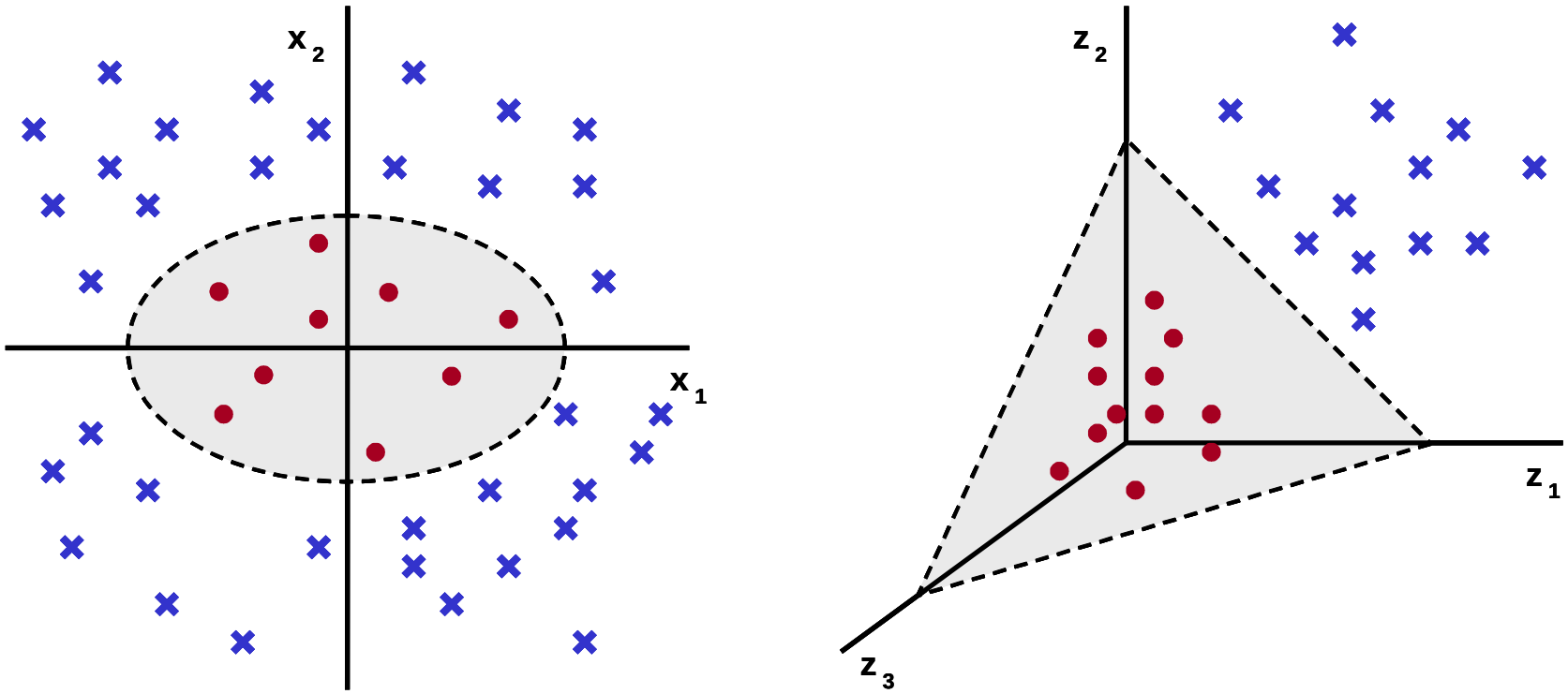
## Cover's theorem on the separability of patterns

- "*A complex pattern-classification problem cast in a high-dimensional space non-linearly is more likely to be linearly separable than in a low-dimensional space*"
- The power of SVMs resides in the fact that they represent a robust and efficient implementation of Cover's theorem
- SVMs operate in two stages
  - Perform a non-linear mapping of the feature vector x onto a high-dimensional space that is hidden from the inputs or the outputs
  - Construct an optimal separating hyperplane in the high-dim space



$$\varphi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$




[Schölkopf, 2002 @; http://kernel-machines.org/]

**Naïve application of this concept by simply projecting to a high-dimensional non-linear manifold has two major problems**

- **Statistical**: operation on high-dimensional spaces is ill-conditioned due to the "curse of dimensionality" and the subsequent risk of overfitting
- **Computational**: working in high-dim requires higher computational power, which poses limits on the size of the problems that can be tackled

**SVMs bypass these two problems in a robust and efficient manner**

- First, generalization capabilities in the high-dimensional manifold are ensured by enforcing a **largest margin** classifier
  - Recall that generalization in SVMs is strictly a function of the margin (or the VC dimension), regardless of the dimensionality of the feature space
- Second, projection onto a high-dimensional manifold is only **implicit**
  - Recall that the SVM solution depends only on the dot product $\langle x_i, x_j \rangle$ between training examples
  - Therefore, operations in high-dim space $\varphi(x)$ do not have to be performed explicitly if we find a function $K(x_i, x_j)$ such that $K(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle$
  - $K(x_i, x_j)$ is called a **kernel** function in SVM terminology

# Implicit mappings: an example

## Consider a pattern recognition problem in $R^2$

- Assume we choose a kernel function $K(x_i, x_j) = (x_i^T x_j)^2$
- Our goal is to find a non-linear projection $\varphi(x)$ such that $(x_i^T x_j)^2 = \varphi^T(x_i)\varphi(x_j)$
- Performing the expansion of $K(x_i, x_j)$

$$K(x_i, x_j) = (x_i^T x_j)^2 = \left((x_{1,1}, x_{1,2})^T (x_{2,1}, x_{2,2})\right)^2 = (x_{1,1}x_{2,1} + x_{1,2}x_{2,2})^2$$
$$= x_{1,1}^2 x_{2,1}^2 + 2x_{1,1}x_{2,1}x_{1,2}x_{2,2} + x_{1,2}^2 x_{2,2}^2$$
$$= (x_{1,1}^2, \sqrt{2}x_{1,1}x_{1,2}, x_{1,2}^2)^T (x_{2,1}^2, \sqrt{2}x_{2,1}x_{2,2}, x_{2,2}^2)$$

  - where $x_{i,k}$ denotes the $k^{th}$ coordinate of example $x_i$

- So in using the kernel $K(x_i, x_j) = (x_i^T x_j)^2$, we are implicitly operating on a higher-dimensional non-linear manifold defined by

$$\varphi(x_i) = [x_{i,1}^2, \sqrt{2}x_{i,1}x_{i,2}, x_{i,2}^2]^T$$

- Notice that the inner product $\varphi^T(x_i)\varphi(x_j)$ can be computed in $R^2$ by means of the kernel $(x_i^T x_j)^2$ without ever having to project onto $R^3$!

[Burges, 1998]

# Kernel methods

## Let's now see how to put together all these concepts

- Assume that our original feature vector $x$ lives in a space $R^D$
- We are interested in non-linearly projecting $x$ onto a higher dimensional implicit space $\varphi(x) \in R^{D1}$ $(D1 > D)$ where classes have a better chance of being linearly separable
  - Notice that we are not guaranteeing linear separability, we are only saying that we have a better chance because of Cover's theorem
- The separating hyperplane in $R^{D1}$ will be defined by

$$\sum_{j=1}^{D1} w_j \varphi_j(x) + b = 0$$

- To eliminate the bias term $b$, let's augment the feature vector in the implicit space with a constant dimension $\varphi_0(x) = 1$
- Using vector notation, the resulting hyperplane becomes

$$w^T \varphi(x) = 0$$

- From our previous results, the optimal (maximum margin) hyperplane in the implicit space is given by

$$w = \sum_{i=1}^{N} \alpha_i y_i \varphi(x_i)$$

– Merging this optimal weight vector with the hyperplane equation

$$w^T \varphi(x) = 0$$

$$\Rightarrow \left( \sum_{i=1}^{N} \alpha_i y_i \varphi(x_i) \right)^T \varphi(x) = 0$$

$$\Rightarrow \sum_{i=1}^{N} \alpha_i y_i \varphi(x_i)^T \varphi(x) = 0$$

– and, since $\varphi^T(x_i)\varphi(x_j) = K(x_i, x_j)$, the optimal hyperplane becomes

$$\sum_{i=1}^{N} \alpha_i y_i K(x_i, x) = 0$$

– Therefore, classification of an unknown example $x$ is performed by computing the weighted sum of the kernel function with respect to the support vectors $x_i$ (remember that only the support vectors have non-zero dual variables $\alpha_i$)

## How do we compute dual variables $\alpha_i$ in the implicit space?

- Very simple: we use the same optimization problem as before, and replace the dot product $\varphi^T(x_i)\varphi(x_j)$ with the kernel $K(x_i, x_j)$
- The Lagrangian dual problem for the non-linear SVM is simply

$$L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i^T, x_j)$$

- subject to the constraints

$$\begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \quad i = 1 \dots N \end{cases}$$

**How do we select the implicit mapping $\varphi(x)$?**

- As we saw in the example a few slides back, we will normally select a kernel function first, and then determine the implicit mapping $\varphi(x)$ that it corresponds to

**Then, how do we select the kernel function $K(x_i, x_j)$?**

- We must select a kernel for which an implicit mapping exists, this is, a kernel that can be expressed as the dot-product of two vectors

**For which kernels $K(x_i, x_j)$ does there exist an implicit mapping $\varphi(x)$?**

- The answer is given by Mercer's Condition

[Burges, 1998; Kaykin, 1999]

# Mercer's Condition

**Let $K(x, x')$ be a continuous symmetric kernel that is defined in the closed interval $a \leq x \leq b$**

- The kernel can be expanded in the series:

$$K(x, x') = \sum_{i=1}^{\infty} \lambda_i \varphi_i(x) \varphi_i(x')$$

  - Strictly speaking, the space where $\varphi(x)$ resides is a Hilbert space, a "generalization" of an Euclidean space where the inner product can be any inner product, not just the scalar dot product [Burges, 1998]
  - With positive coefficients $\lambda_i > 0 \ \ \forall i$
- For this expansion to be valid and for it to converge absolutely and uniformly, it is necessary and sufficient that the condition

$$\int_a^b \int_a^b K(x, x') \psi(x) \psi(x') dx dx' \geq 0$$

- holds for all $\psi(\cdot)$ for which $\int_a^b \psi^2(x) dx \leq \infty$
  - The functions $\varphi_i(x)$ are called eigenfunctions of the expansion, and the numbers $\lambda_i$ are the eigenvalues. The fact that all of the eigenvalues are positive means that the kernel is positive definite
- Notice that the dimensionality of the implicit space can be infinitely large
- Mercer's Condition only tells us whether a kernel is actually an inner-product kernel, but it does not tell us how to construct the functions $\varphi_i(x)$ for the expansion

[Kaykin, 1999]

## Which kernels meet Mercer's condition?

- Polynomial kernels

$$K(x, x') = (x^T x' + 1)^p$$

  - The degree of the polynomial is a user-specified parameter

- Radial basis function kernels

$$K(x, x') = \exp\left(-\frac{1}{2\sigma^2}\|x - x'\|^2\right)$$

  - The width $\sigma$ is a user-specified parameter, but the number of radial basis functions and their centers are determined automatically by the number of support vectors and their values

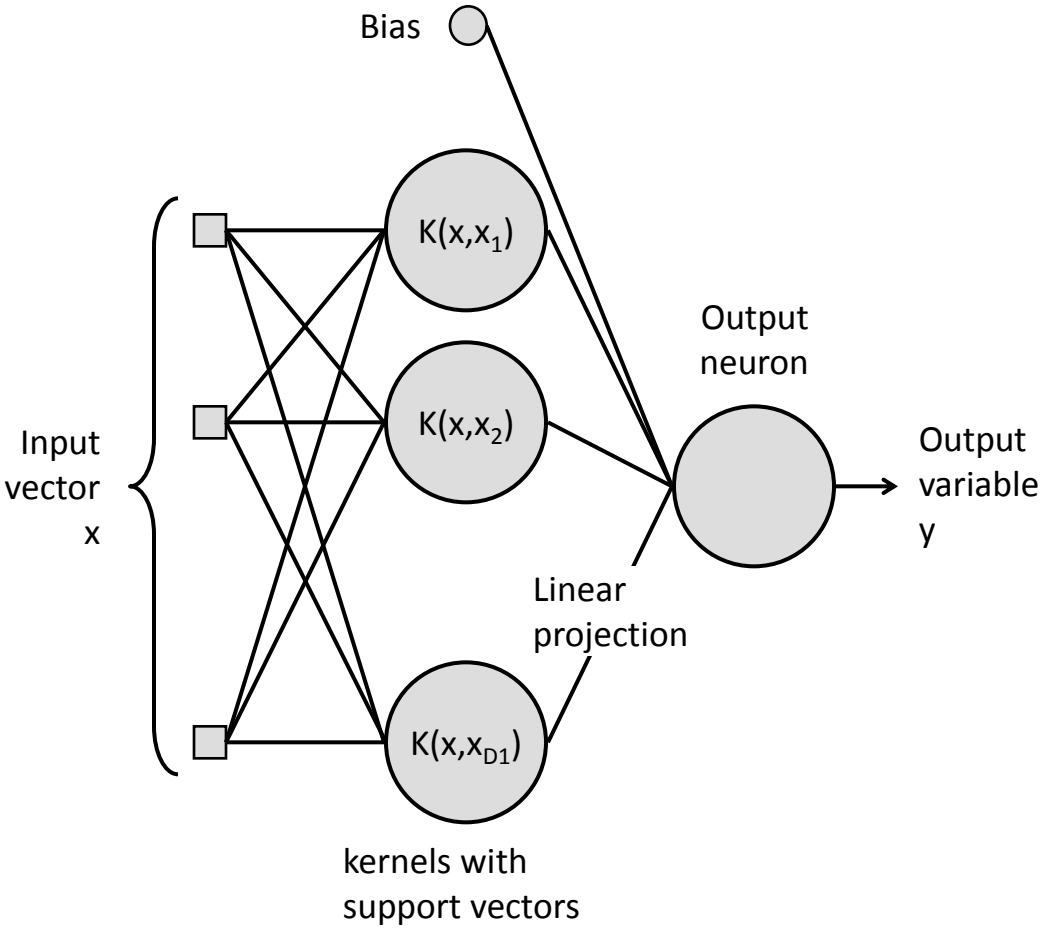
- Two-layer perceptron

$$K(x, x') = \tanh(\beta_0 x^T x' + \beta_1)$$

  - The number of hidden neurons and their weight vectors are determined automatically by the number of support vectors and their values, respectively. The H-O weights are the Lagrange multipliers $\alpha_i$
  - However, this kernel will only meet Mercer's condition for certain values of $\beta_0$ and $\beta_1$

[Burges, 1998; Kaykin, 1999]

# Architecture of an SVM




[Kaykin, 1999]

# Numerical example

**To illustrate the operation of a non-linear SVM we will solve the classical XOR problem**

- Dataset
  - Class 1: $x_1 = (-1,-1), \quad x_4 = (+1,+1)$
  - Class 2: $x_2 = (-1,+1), \quad x_3 = (+1,-1)$
- Kernel function
  - Polynomial of order 2: $K(x,x') = (x^T x' + 1)^2$

**Solution**

- The implicit mapping can be shown to be 5-dimensional

$$\varphi(x) = \begin{bmatrix} 1 & \sqrt{2}x_{i,1} & \sqrt{2}x_{i,2} & \sqrt{2}x_{i,1}x_{i,2} & x_{i,1}^2 & x_{i,2}^2 \end{bmatrix}^T$$

- To achieve linear separability, we will use $C = \infty$
- The objective function for the dual problem becomes

$$L_D(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}\sum_{i=1}^{4}\sum_{j=1}^{4}\alpha_i\alpha_j y_i y_j k_{ij}$$

  - subject to the constraints $\begin{cases} \sum_{i=1}^{N}\alpha_i y_i = 0 \\ 0 \le \alpha_i \le C \quad i = 1 \ldots N \end{cases}$

[Cherkassky and Mulier, 1998; Haykin, 1999]

– where the inner product is represented as a $4 \times 4$ K matrix

$$K = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$$

– Optimizing with respect to the Lagrange multipliers leads to the following system of equations

$$\begin{aligned} 9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 &= 1 \\ -\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 &= 1 \\ -\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 &= 1 \\ \alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 &= 1 \end{aligned}$$

– whose solution is $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.125$

– Thus, all data points are support vectors in this case

– For this simple problem, it is worthwhile to write the decision surface in terms of the polynomial expansion

$$w = \sum_{i=1}^{4} \alpha_i y_i \varphi(x_i) = \begin{bmatrix} 0 & 0 & 0 & 1/\sqrt{2} & 0 & 0 \end{bmatrix}^T$$

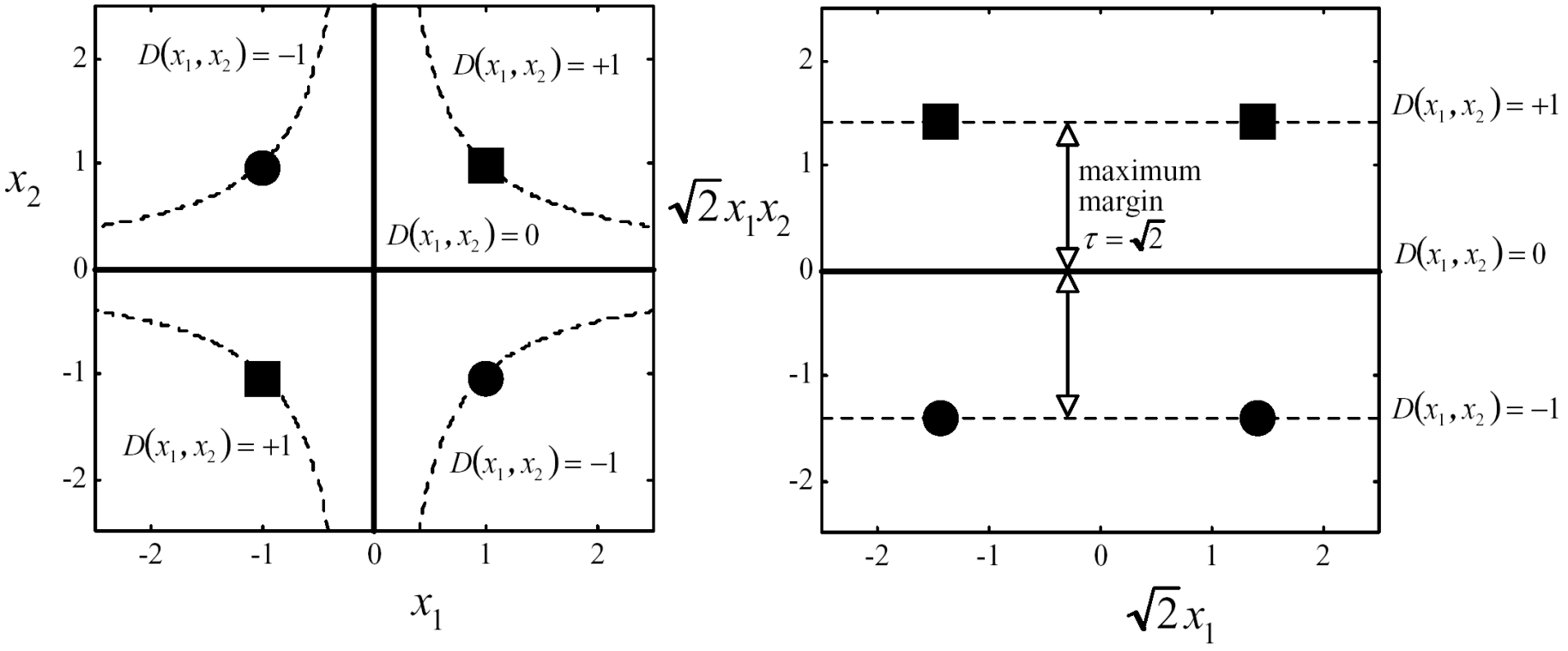
– resulting in the intuitive non-linear discriminant function

$$g(x) = \sum_{i=1}^{6} w_i \varphi_i(x) = x_1 x_2$$

– which has zero empirical error on the XOR training set

## Decision function defined by the SVM

– Notice that the decision boundaries are non-linear in the original space $R^2$, but linear in the implicit space $R^6$



# Optimization techniques for SVMs

**SVMs involve the solution of a quadratic programming (QP) problem**

$$J(\alpha) = 1^T \alpha - \frac{1}{2} \alpha^T H \, \alpha$$

- where $H_{ij} = y_i y_j K(x_i, x_j)$, s.t. $\begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \quad i = 1 \ldots N \end{cases}$

**Several commercial optimization libraries can be used to solve this dual QP problem**

- The use of these libraries is, however, limited to small- to medium-size problems (1,000s of examples) since the # elements in the quadratic matrix $H$ is equal to the square of the number of training examples

**A number of alternative optimization procedures have therefore been proposed by the SVM community**

- Chunking
- Decomposition methods
- Sequential Minimal Optimization

[Müller et al., 2001]

## Chunking

- This method is based on two facts
  - Many of the optimal $\alpha$s will be zero (or on the upper bound C). The QP solution is independent of these zero parameters, so their corresponding rows and columns in the quadratic matrix can be eliminated
  - In addition, the optimal $\alpha$s must meet the KKT condition
- At every step, chunking will solve a problem containing all the non-zero $\alpha$s plus some of the $\alpha$s that violate the KKT condition
- The size of the problem varies with every iteration, but is finally equal to the number of support vectors
- However, chunking is still limited by the maximum number of support vectors that fit in memory
- In addition, chunking requires an inner QP optimizer to solve each of the smaller problems

## Decomposition methods

- Decomposition methods are similar to chunking in concept, except for the size of the sub-problems is always fixed
- These methods are based on the fact that a sequence of QPs which contain at least one sample violating the KKT conditions will eventually converge to an optimal solution
- The original algorithm suggests adding and removing one example at every step, but this leads to very slow convergence
- Practical implementations use various heuristics to add or remove multiple examples at a time
- Decomposition methods still require an inner QP solver for the sub-problems

## Sequential Minimal Optimization (SMO)

- This algorithm represents the extreme case of a decomposition method: at every iteration SMO solves a QP problem of size TWO. This has two advantages
  - A QP problem of size two can be solved analytically; no QP solver is required
  - No extra matrix storage is required
- The main problem in SMO is how to choose a good pair of variables to optimize at every iteration. This is accomplished with a number of heuristics
- The implementation of SMO is straightforward, and the pseudo-code is even available [Platt, 1999]

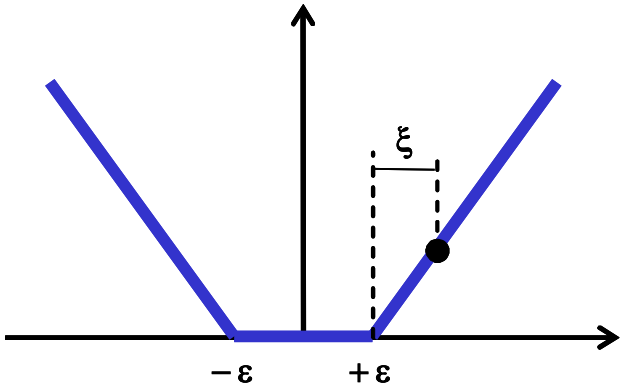
# SVM regression

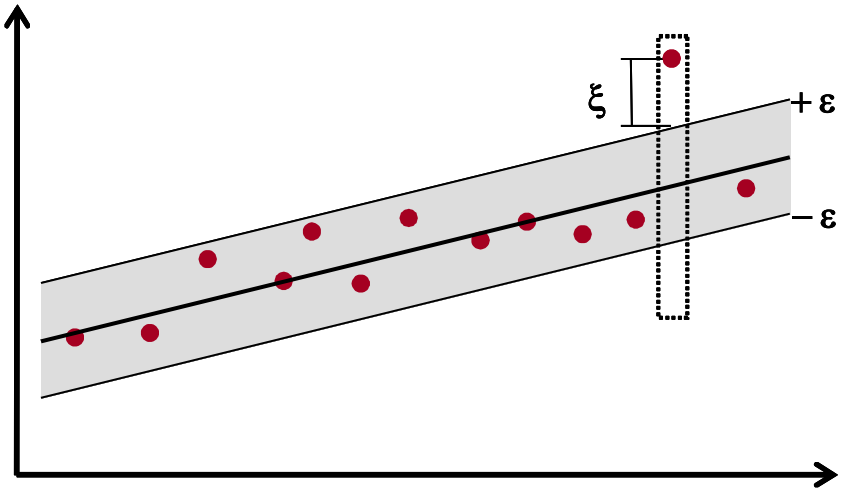
## SVM extensions for regression are also available

- SVM regression uses an $\epsilon$-insensitive loss function
- Geometrically, this can be thought of as fitting a tube of width $2\epsilon$ to the data
- The primal minimization problem is

$$minimize\ J(w,\xi,\xi^*) = \frac{1}{2}\|w\|^2 + C\ \sum_{i=1}^{N}(\xi_i + \xi_i^*)$$

$$subject\ to \begin{cases} (w^T x_i + b) - y_i \leq \epsilon + \xi_i \\ y_i - (w^T x_i + b) \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases}$$



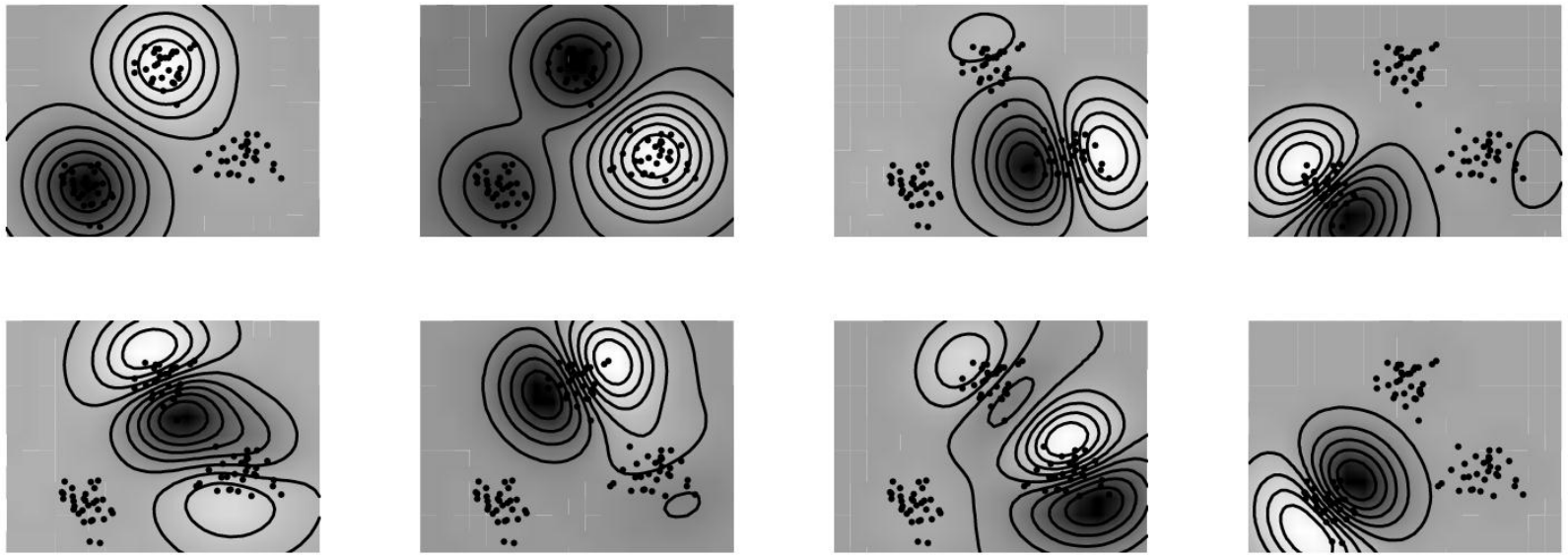




[Schölkopf et al., 1999]

# Kernel PCA

## SMVs can also be used to perform non-linear PCA

- In this case, the problem involves the computation of eigenvectors and eigenvalues of the SVM Kernel matrix $K(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle$
- Because Kernel PCA is implicitly performed in a high-dimensional feature space, it can extract more features that those available in the original feature space (see example below)
- Similarly, SMV extensions to Fisher's LDA are also available



*First 8 non-linear principal components from a 2-dimensional dataset (from [Schölkopf et al., 1996])*

# Discussion

## Advantages of SVMs

- There are no local minima, because the solution is a QP problem
- The optimal solution can be found in polynomial time
- Few model parameters to select: the penalty term C, the kernel function and parameters (e.g., spread $\sigma$ in the case of RBF kernels)
- Final results are stable and repeatable (e.g., no random initial weights)
- SVM solution is sparse; it only involves the support vectors
- SVMs represent a general methodology for many PR problems: classification, regression, feature extraction, clustering, novelty detection, etc.
- SVMs rely on elegant and principled learning methods
- SVMs provide a method to control complexity independently of dimensionality
- SVMs have been shown (theoretically and empirically) to have excellent generalization capabilities

[Bennett and Campbell, 2000]

## Challenges

- Do SVMs always perform best? Can they beat a hand-crafted solution for a particular problem?
- Do SVMs eliminate the model selection problem? Can the kernel functions be selected in a principled manner? SVMs still require selection of a few parameters, typically through cross-validation
- How does one incorporate domain knowledge? Currently this is performed through the selection of the kernel and the introduction of "artificial" examples
- How interpretable are the results provided by an SVM?
- What is the optimal data representation for SVM? What is the effect of feature weighting? How does an SVM handle categorical or missing features?