ECE7252
Statistical Learning for Signal Processing

Lectures 15-16: Support Vector Machines

Chin-Hui Lee
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
chl@ece.gatech.edu

Learning and Empirical Risk Minimization

- The aim of any learning machine is to estimate *g(x)* from a finite set of observations by minimizing some kind of an error function, e.g., the *empirical risk*:

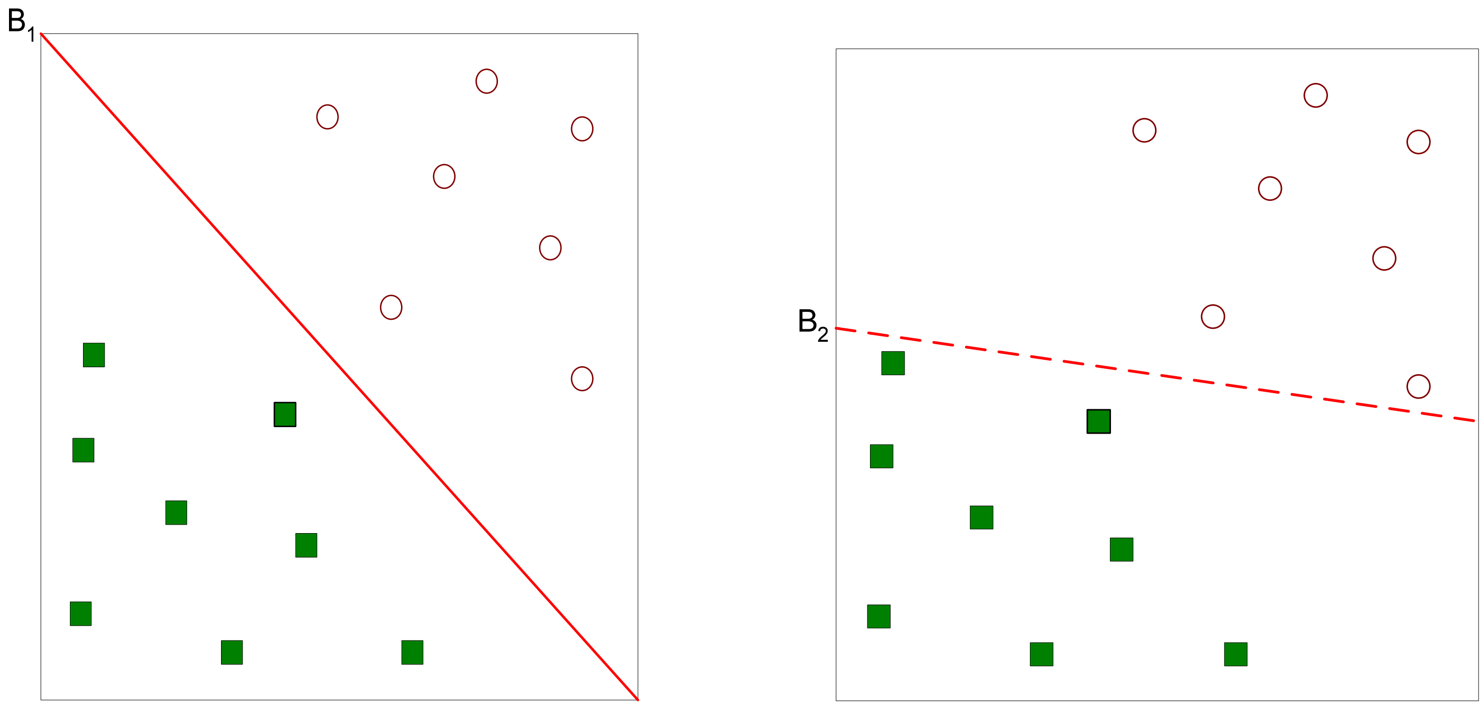
$$R_{emp}(w, w_0) = \frac{1}{n}\sum_{k=1}^{n}[z_k - g(x_k, w, w_0)]^2$$

class labels: $$z_k = \begin{cases} +1 \; if \;\; \mathbf{x}_k \in \omega_1 \\ -1 \; if \;\; \mathbf{x}_k \in \omega_2 \end{cases}$$

Learning and Generalization

- The conventional *empirical risk* minimization over training data **does not** imply good generalization to novel test data
  – There could be a number of different functions which all give a good approximation to the training data set
  – It is difficult to determine a function which best captures the true underlying structure of the data distribution

Learning, Generalization and Selection



Which solution is better?

Strengths of SVMs

- Good generalization in theory
- Good generalization in practice
- Work well with few training instances
- Find globally best model
- Efficient algorithms
- Amenable to the kernel trick

Linear Separators

- Training instances

  $x \in \Re^n$

  $y \in \{-1, 1\}$

$w \in \Re^n$

$b \in \Re$

- Hyperplane

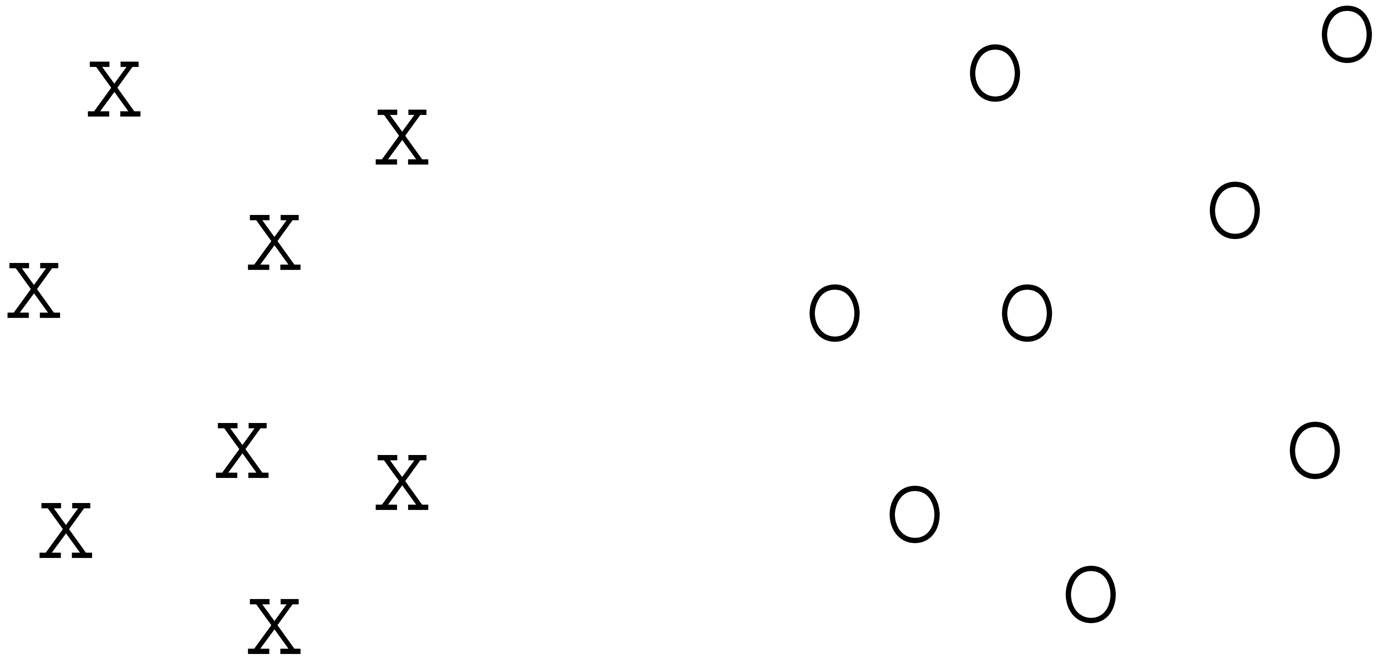
  $<w, x> + b = 0$

  $w_1x_1 + w_2x_2 \ldots + w_nx_n + b = 0$

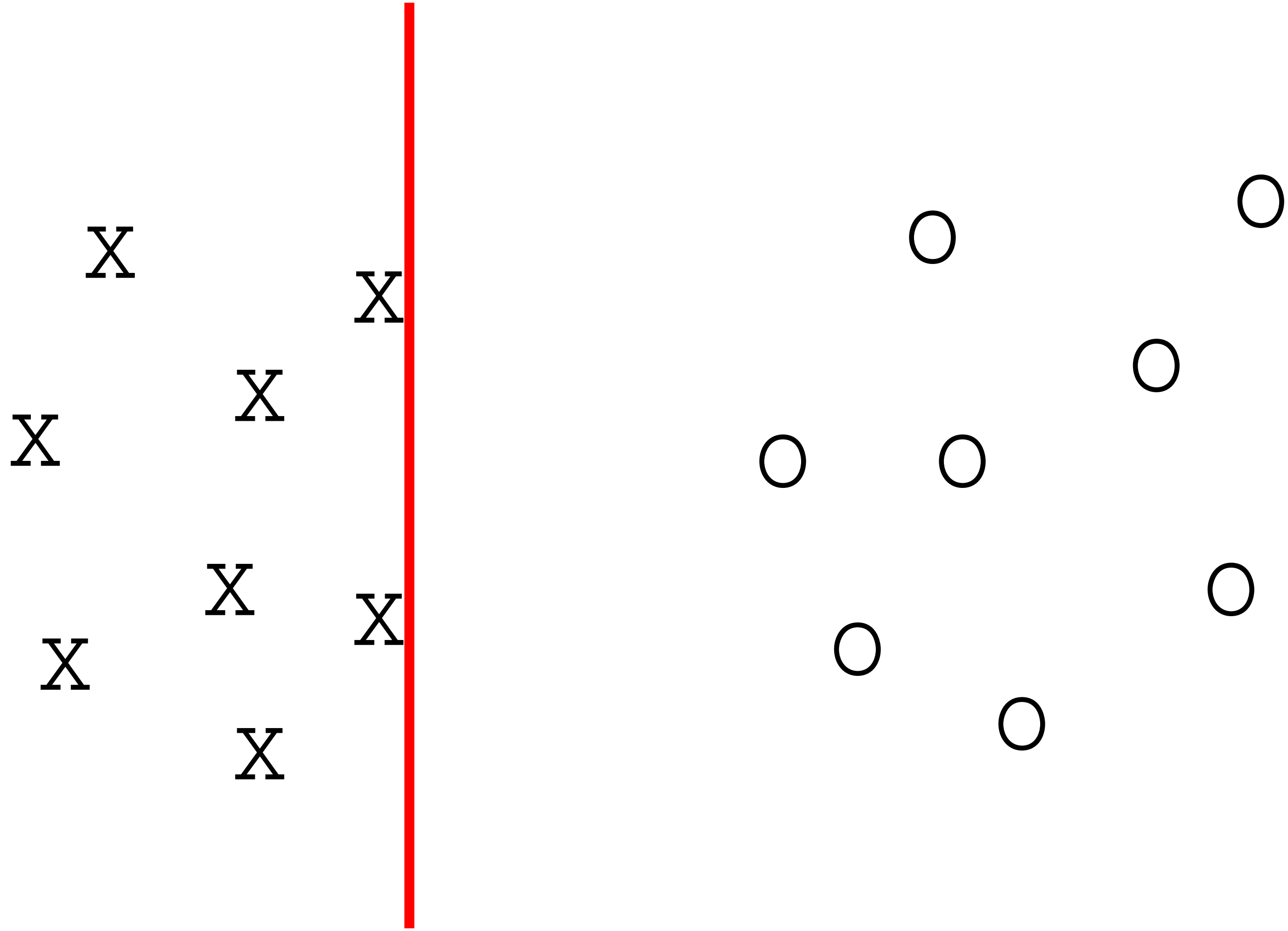
- Decision function

  $f(x) = sign(<w, x> + b)$

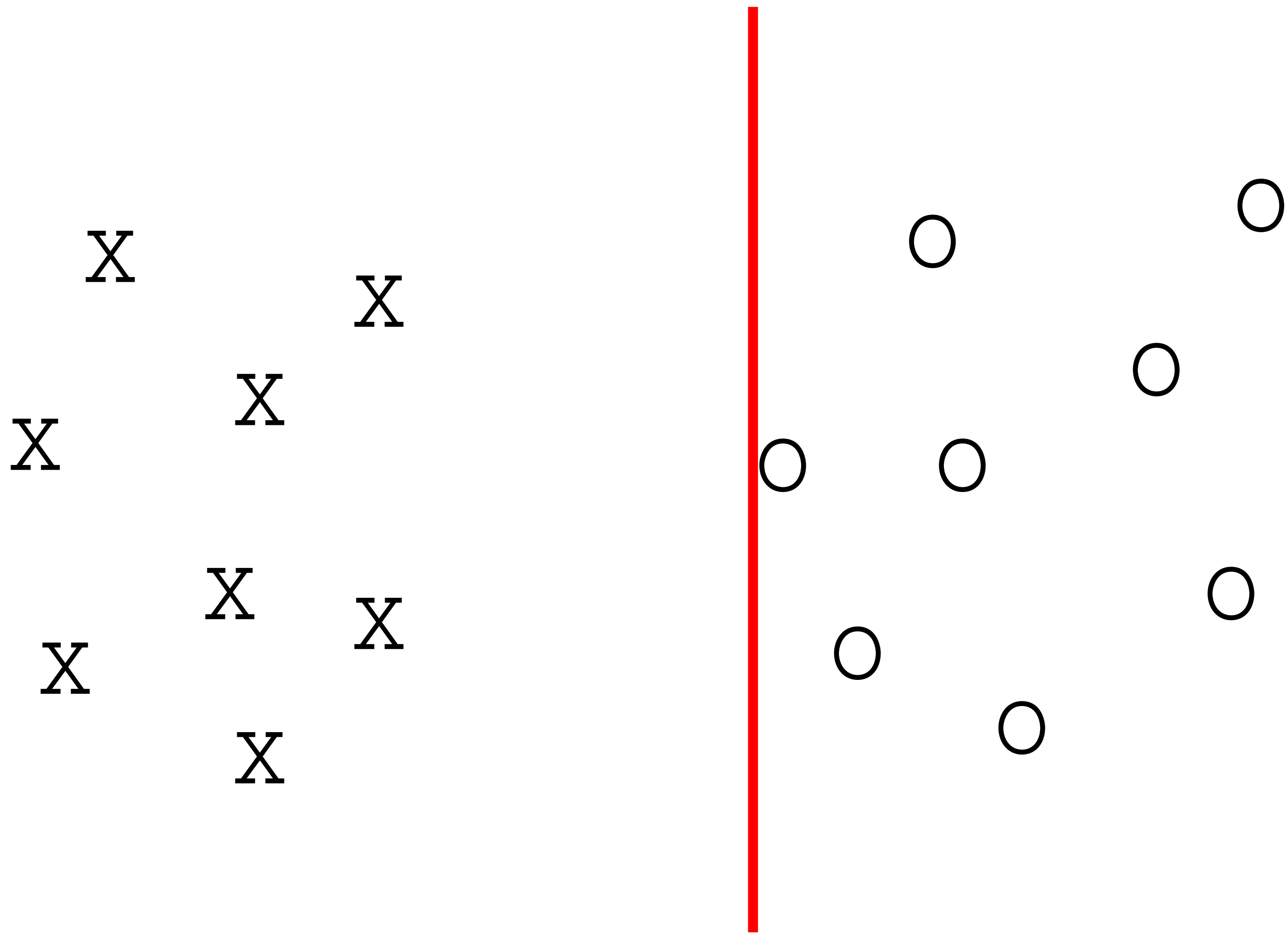
Intuitions



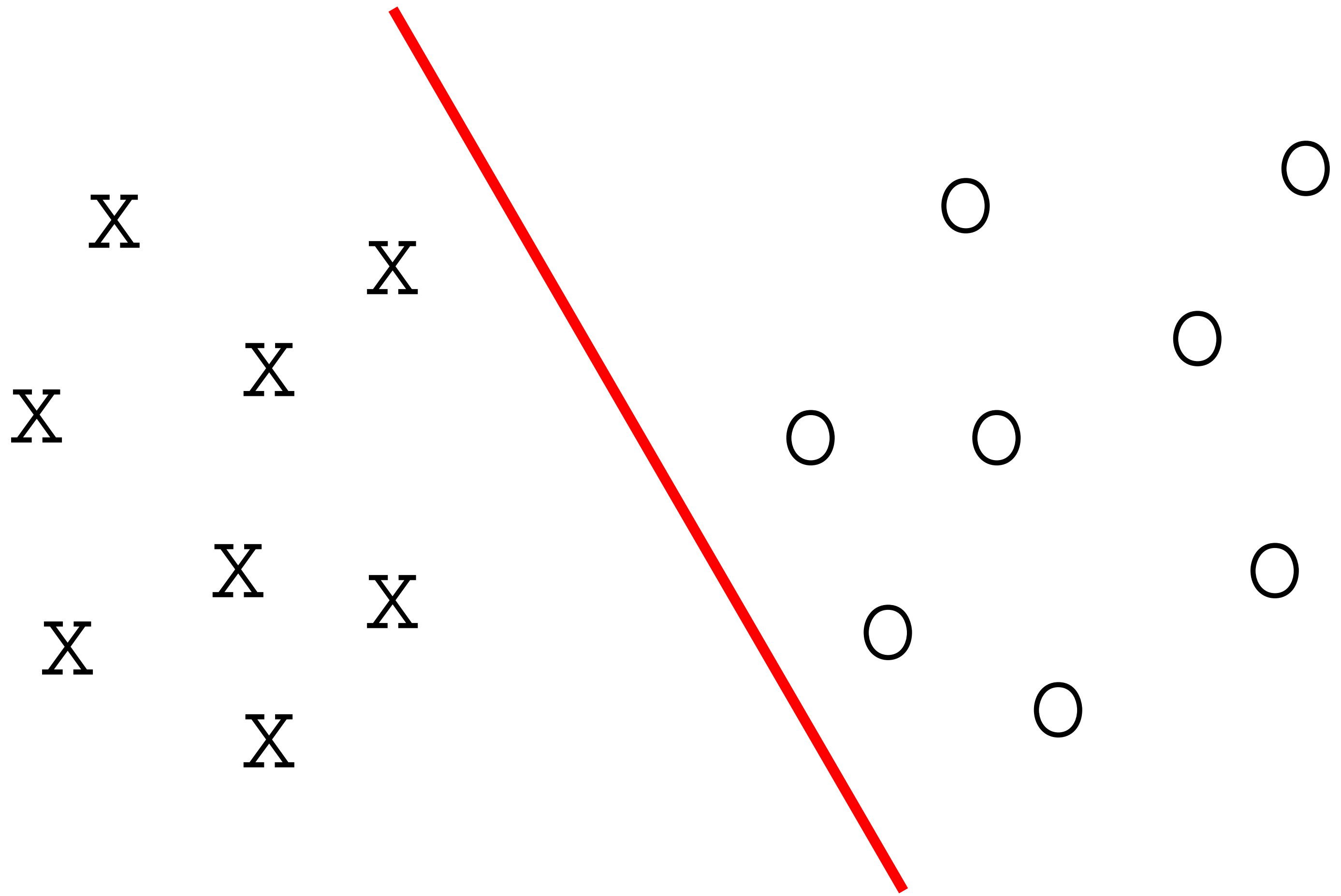
Intuitions



Intuitions

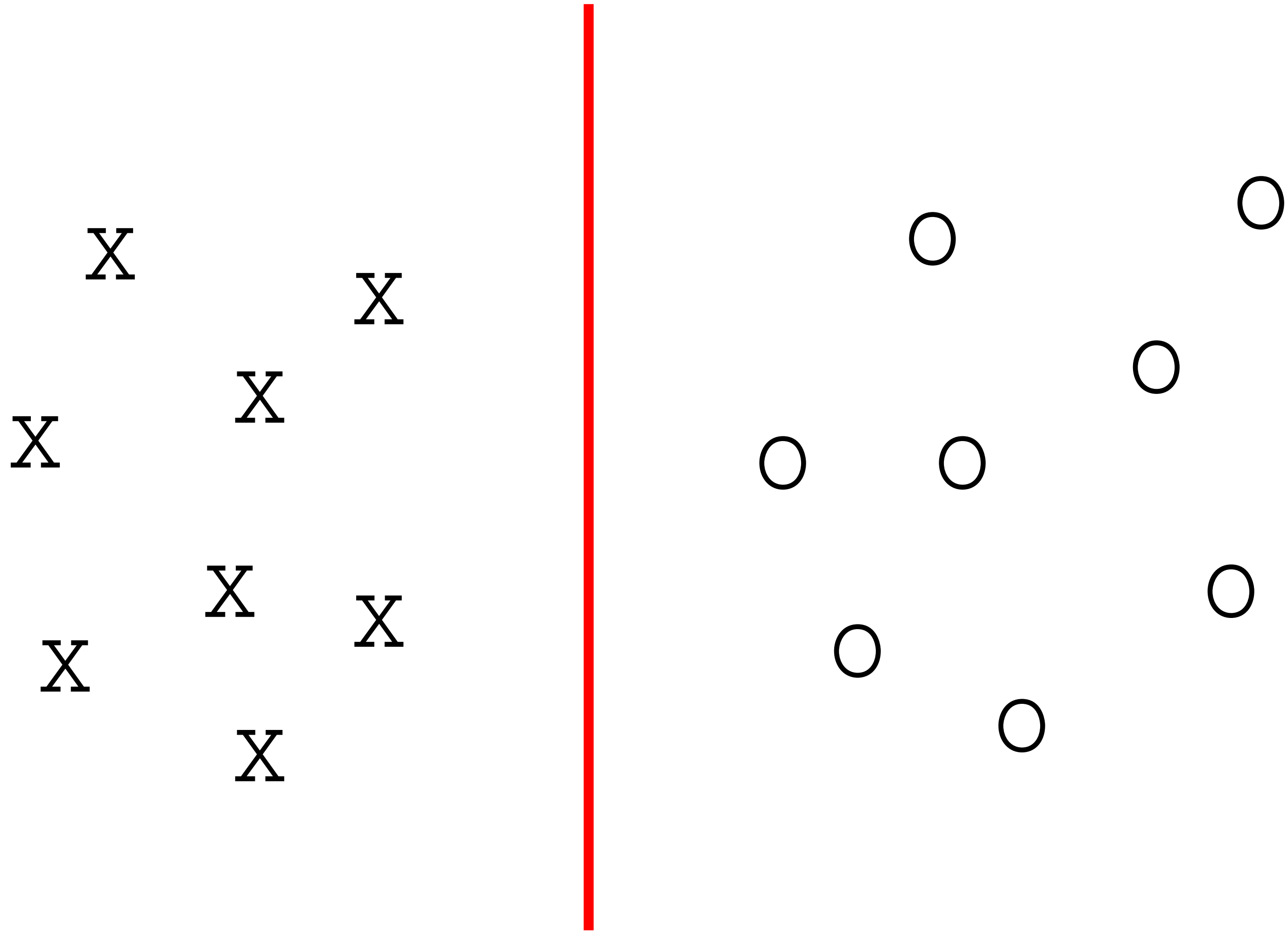


Intuitions

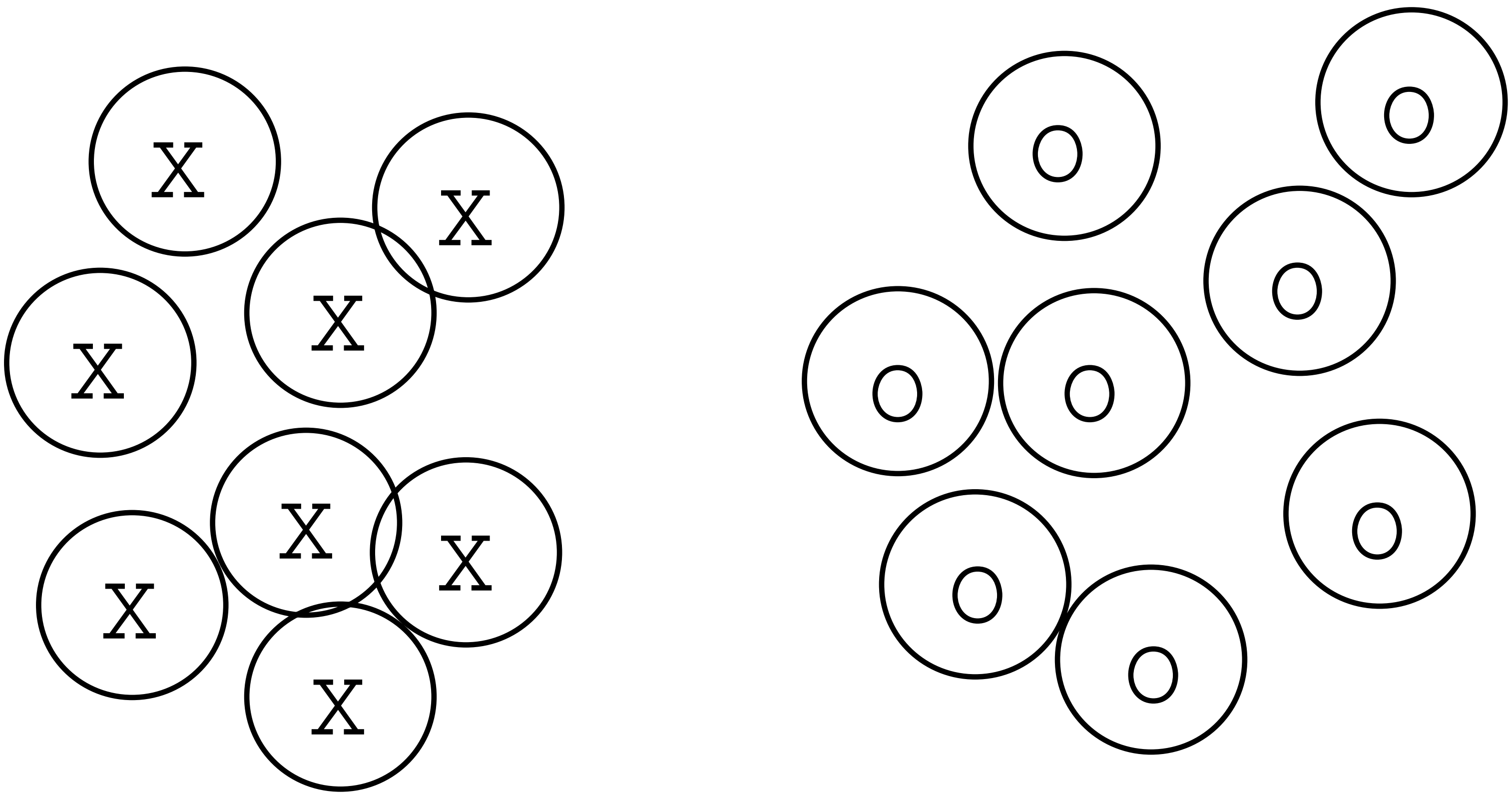


Center of Signal and Image Processing
Georgia Institute of Technology

A "Good" Separator

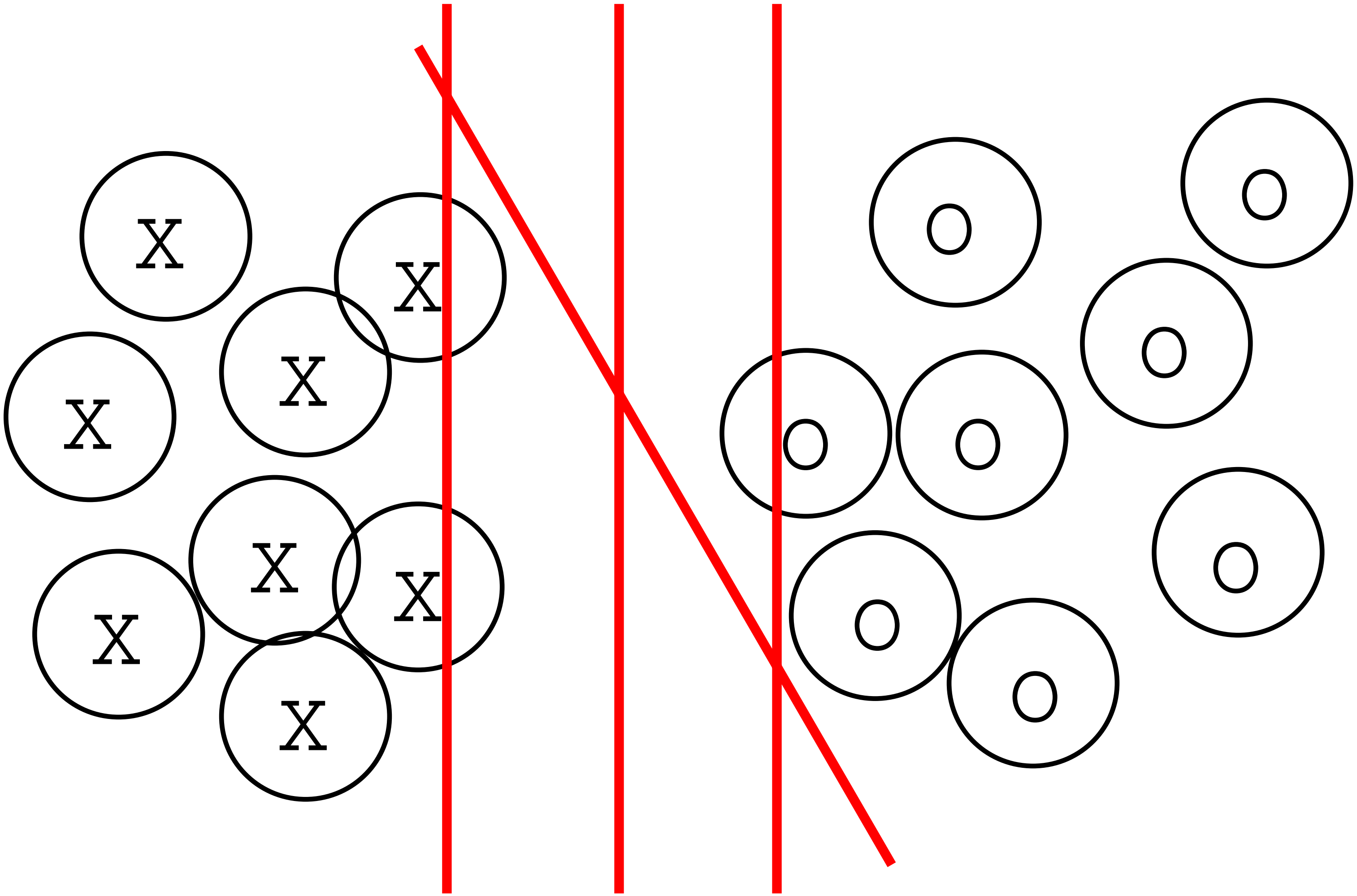


Noise in the Observations

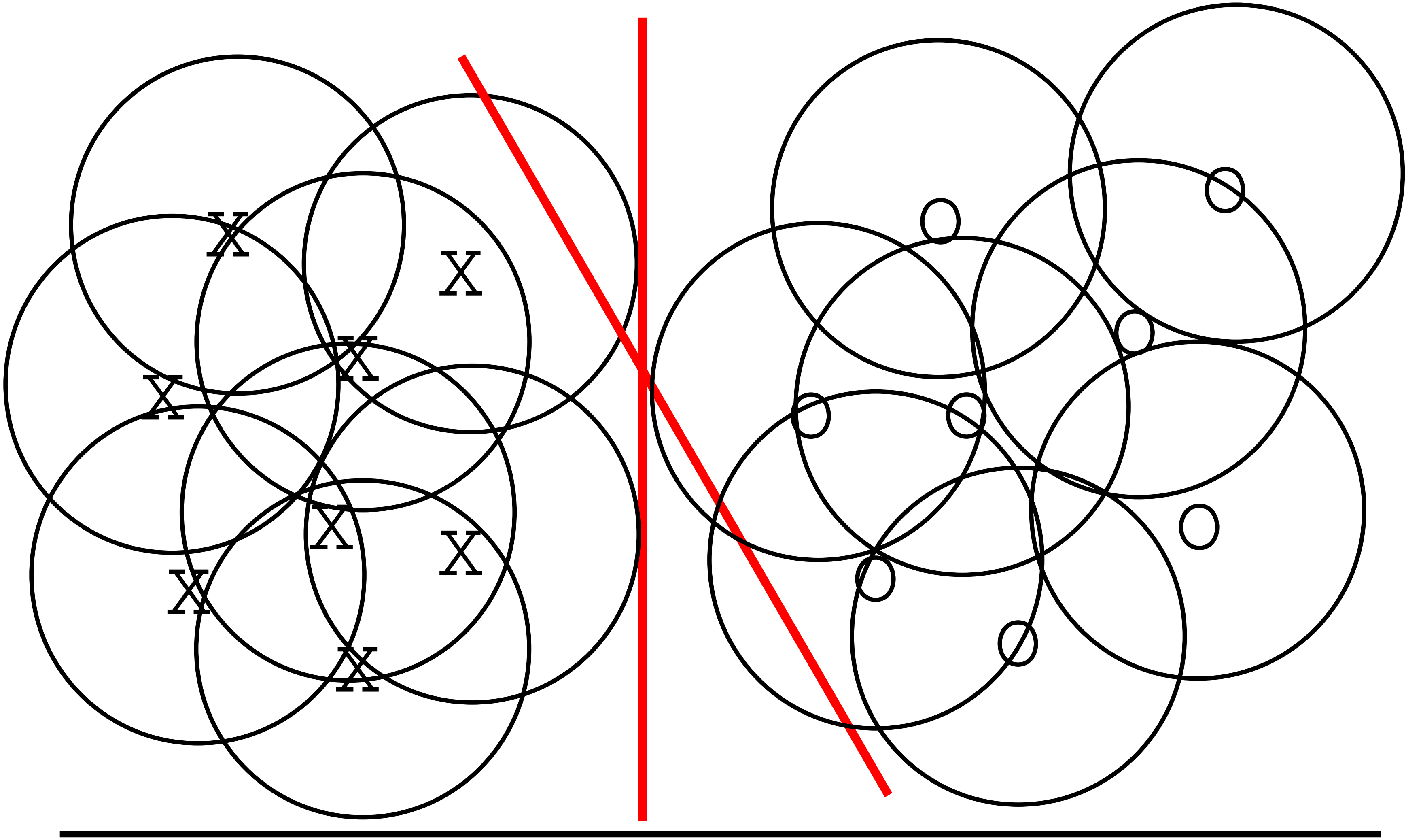


Center of Signal and Image Processing
Georgia Institute of Technology

Ruling Out Some Separators

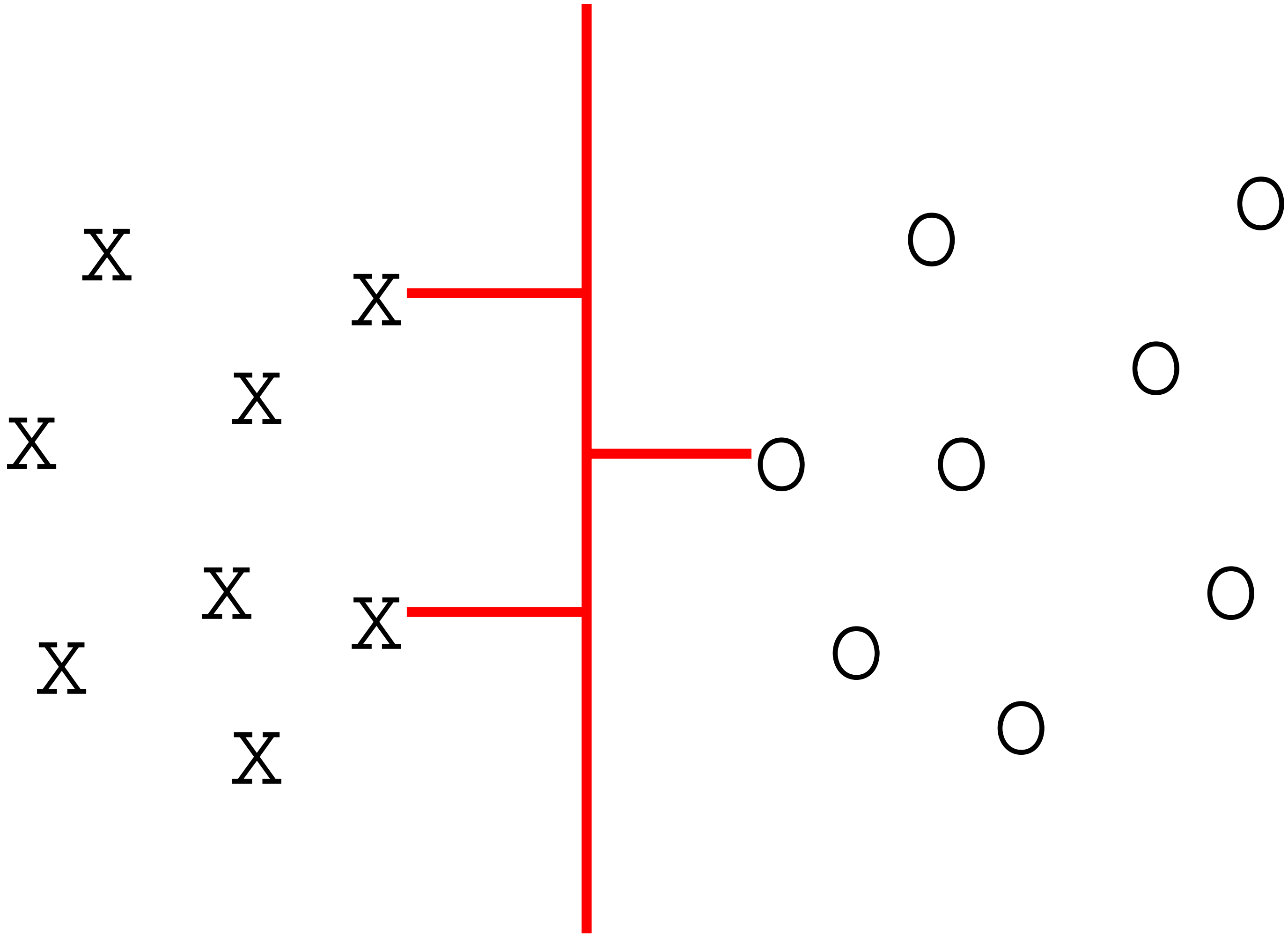


Lots of Noise

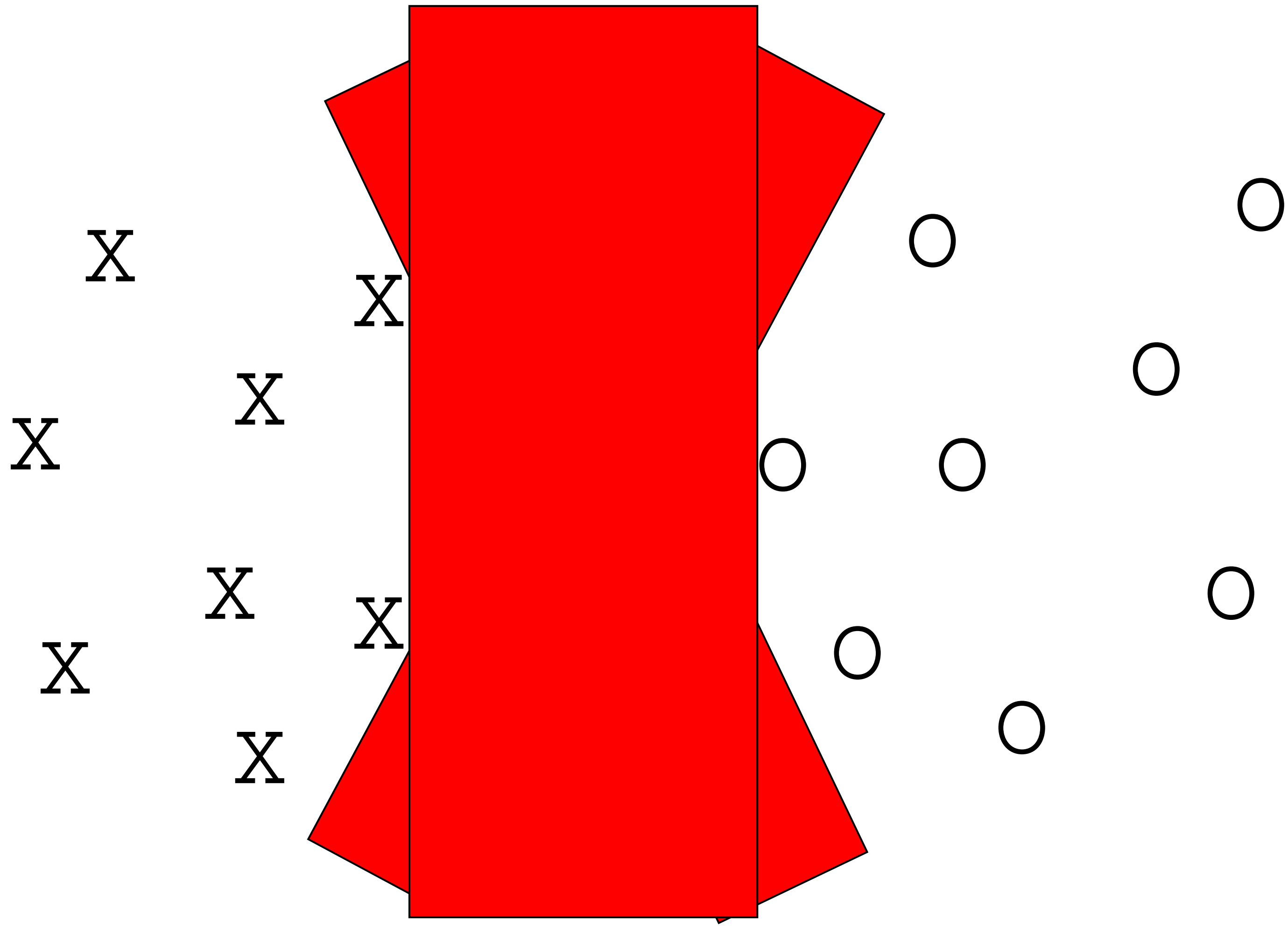


Center of Signal and Image Processing
Georgia Institute of Technology

Maximizing the Margin



"Fat" Separators



Overview of SVM

- SVMs perform *structural risk minimization* to achieve good generalization
- The optimization criterion is the width of the margin between the classes
- Training is equivalent to solving a quadratic programming problem with linear constraints.
- Primarily two-class classifiers but can be extended to multiple classes

Why Maximize Margin?

- Increasing margin reduces *capacity*
- Must restrict capacity to generalize
  - m training instances
  - 2^m ways to label them
  - What if function class that can separate them all?
  - *Shatters* the training instances
- VC Dimension is largest *m* such that function class can shatter some set of *m* points

The Math

- Training instances

  $x \in \Re^n$

  $y \in \{-1, 1\}$
- Decision function

  $f(x) = sign(\langle w,x \rangle + b)$

  $w \in \Re^n$

  $b \in \Re$
- Find *w* and *b* that
  - Perfectly classify training instances
    - Assuming linear separability
  - Maximize margin

The Math

- For perfect classification, we want
  - $y_i\ (\langle w, x_i \rangle + b) \geq 0$ for all i
  - Why?
- To maximize the margin, we want
  - w that minimizes $|w|^2$

Statistical Learning: Capacity & VC Dimension

- To guarantee an "upper bound on generalization error", the capacity of the learned functions must be controlled
  - Intuitively, functions with high *capacity* can represent many dichotomies for a given data set.
- In statistical learning, the Vapnik-Chervonenkis (VC) dimension is one of the most popular measures of capacity
  - Remember Shannon channel capacity

Structural Risk Minimization

- A function that:

  (1) minimizes the empirical risk

  (2) has low VC dimension

  will generalize well <u>regardless of the dimensionality of the input space</u> (structural risk minimization).

$$err_{true} \leq err_{train} + \sqrt{\frac{VC(\log(2n/VC)+1)-\log(\delta/4)}{n}}$$

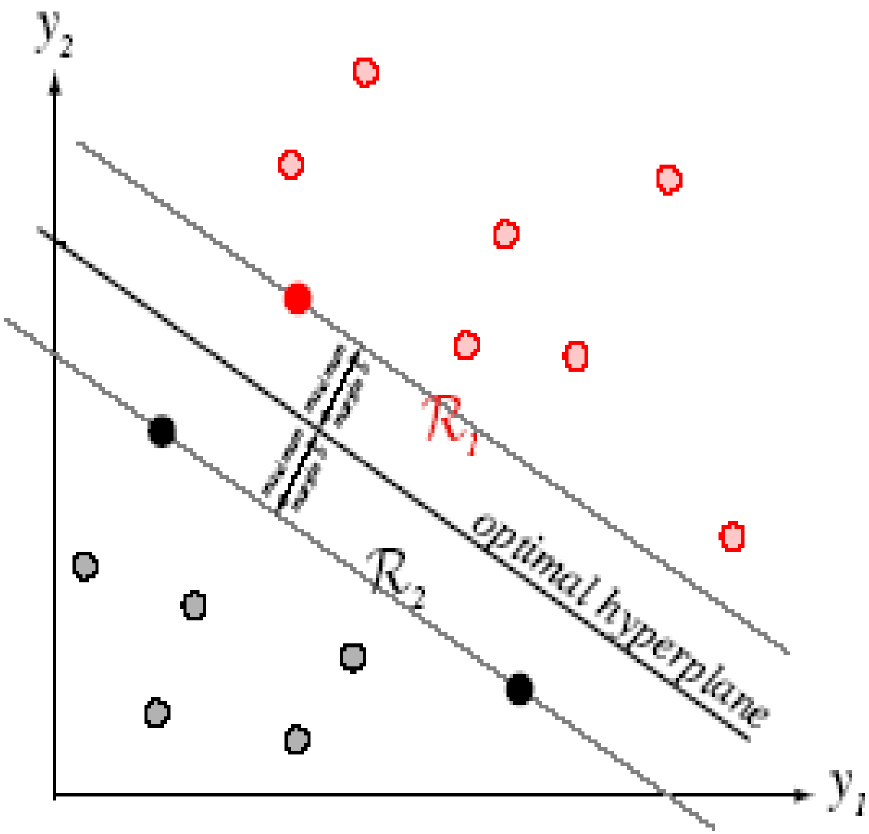
(not very tight bound ..)

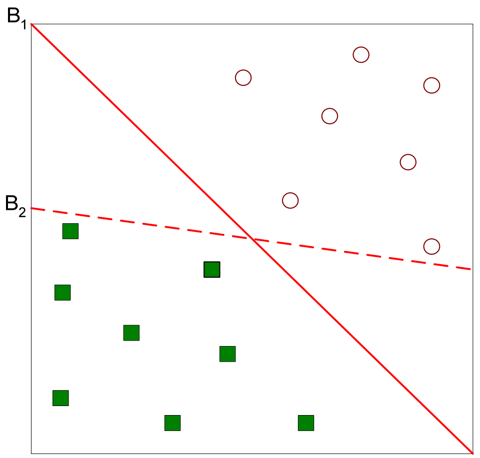
with probability *(1-δ)* (Vapnik, 1995, "*Structural Minimization Principle*")

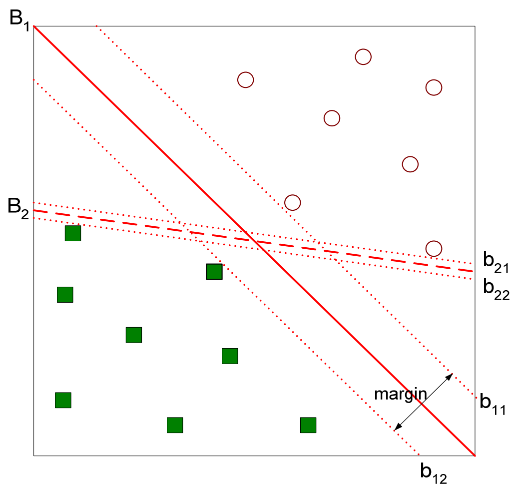
Margin of separation & Optimal Hyperplane

- Vapnik has shown that maximizing the margin of separation between the classes is equivalent to minimizing the VC dimension
- The optimal hyperplane is the one giving the largest margin of separation between the classes
- The empty area around the decision boundary defined by the distance to the nearest training patterns (i.e., support vectors)
- These are the most difficult patterns to classify

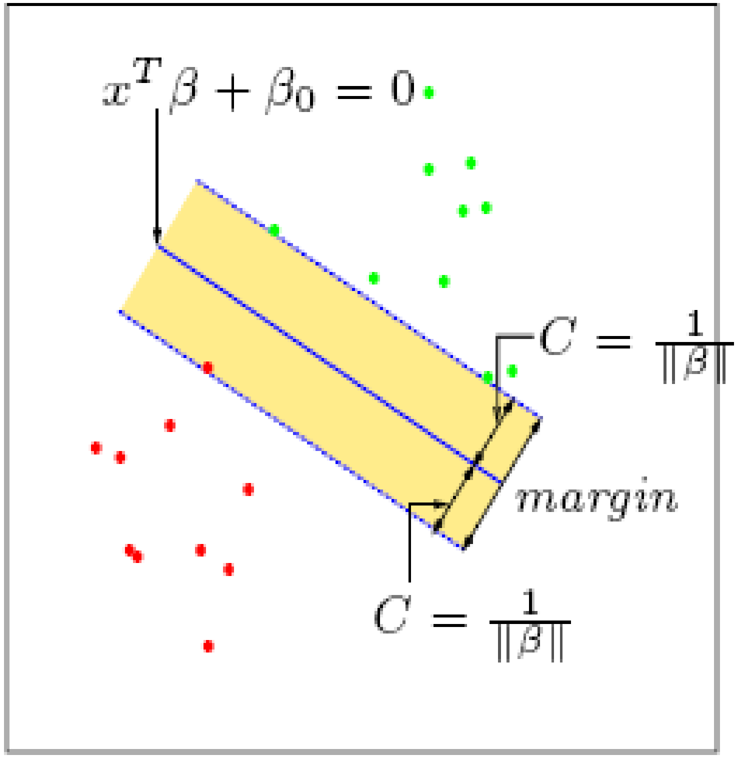
Margin of Separation and Support Vectors

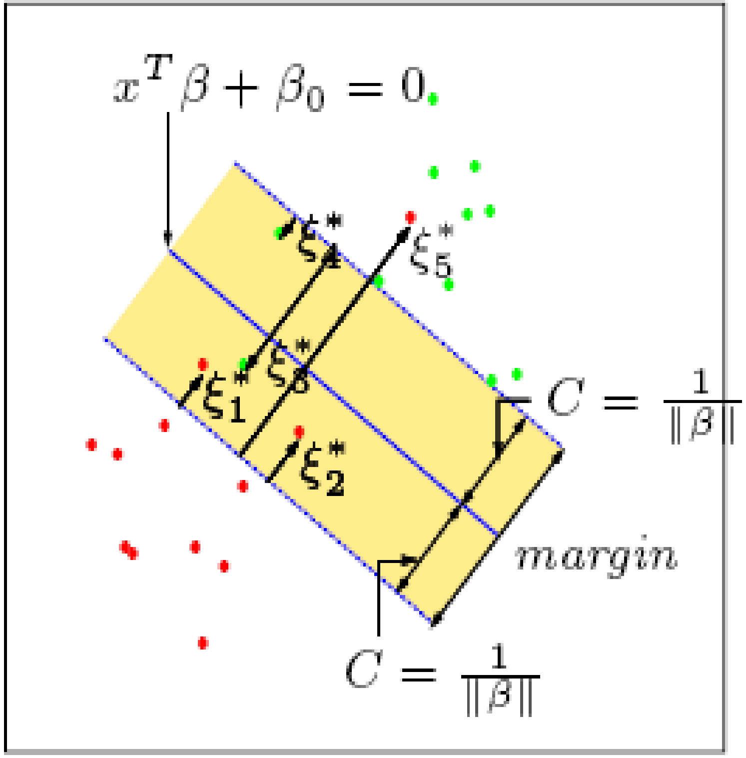






Margin: Separable & Non-Separable Cases





Linear SVM: the Separable Case

- Linear discriminant: $g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0$

  Decide ω_1 if $g(\mathbf{x}) > 0$ and ω_2 if $g(\mathbf{x}) < 0$

- Class labels:

$$z_k = \begin{cases} +1 \ \mathit{if} \ \ \mathbf{x}_k \in \omega_1 \\ -1 \ \mathit{if} \ \ \mathbf{x}_k \in \omega_2 \end{cases}$$

- Normalized version:

$$z_k g(\mathbf{x}_k) > 0 \quad or \quad z_k(\mathbf{w}^t \mathbf{x}_k + w_0) > 0, \quad for \ k = 1, 2, \ldots, n$$

Linear SVM: the Separable Case (Cont'd)

- The distance of a point $\mathbf{x}_k$ from the separating hyperplane should satisfy the constraint:

$$\frac{z_k g(\mathbf{x}_k)}{\|w\|} \geq b, \quad b > 0$$

- To ensure uniqueness, impose: $b\|w\|=1$

- The above constraint becomes:

$$z_k g(\mathbf{x}_k) \geq 1 \quad where \ \ b = \frac{1}{\|w\|}$$

Linear SVM: the Separable Case (Cont'd)

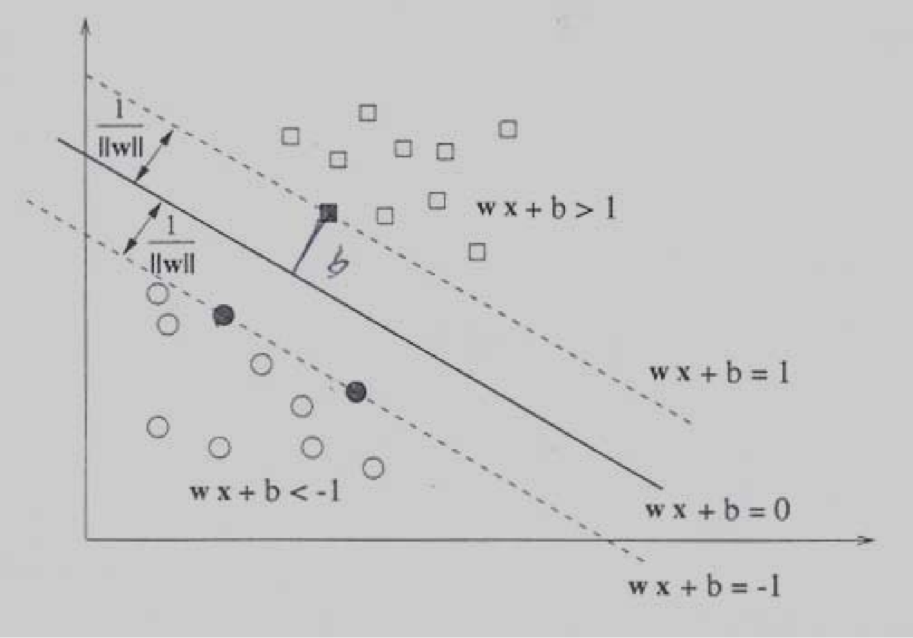
$$\textbf{Problem 1}: \text{Minimize } \frac{1}{2}\|w\|^2$$

$$\text{subject to } z_k(w^t x_k + w_0) \geq 1, \quad k = 1, 2, \ldots, n$$

$$z_k(\mathbf{w}^t \mathbf{x}_k + w_0) > 1, \; for \; k = 1, 2, \ldots, n$$

maximize margin:

$$\frac{2}{\|w\|}$$



quadratic programming problem !

Linear SVM: Dual Formulation

- Use Lagrange optimization:

$$L(\mathbf{w}, w_0, \lambda) = \frac{1}{2} \| \mathbf{w} \|^2 - \sum_{k=1}^{n} \lambda_k [z_k (\mathbf{w}^t \mathbf{x}_k + w_0) - 1], \quad \lambda_k \geq 0$$

- Easier to solve the "dual" problem:

$$\textbf{Problem 2}: \text{Maximize } \sum_{k=1}^{n} \lambda_k - \frac{1}{2} \sum_{k,j}^{n} \lambda_k \lambda_j z_k z_j \mathbf{x}_j^t \mathbf{x}_k$$

$$\text{subject to } \sum_{k=1}^{n} z_k \lambda_k = 0, \quad \lambda_k \geq 0, \; k = 1, 2, \ldots, n$$

- Decision function

$$f(x) = \text{sign}(\Sigma_i \, \alpha_i \, y_i \langle x, x_i \rangle + b)$$

Center of Signal and Image Processing
Georgia Institute of Technology

Linear SVM: Solution

- The solution is given by:

$$w = \sum_{k=1}^{n} z_k \lambda_k \mathbf{x}_k$$

$$w_0 = z_k - w^t \mathbf{x}_k$$

only support vectors contribute to the solution!!

$$\boxed{g(x) = w^T \bullet x + w_0 = \sum_k z_k \lambda_k (x^T \bullet x_k) + w_0}$$

- It can be shown that if x_k is <u>not</u> a support vector, then λ_k=0

What if Not Perfectly Linearly Separable?

- Cannot find *w* and *b* that satisfy

  $y_i\ (<w,x_i> + b) \geq 1$ for all i

- Introduce slack variables ξ_i

  $y_i\ (<w,x_i> + b) \geq 1 - \xi_i$ for all i

- Minimize

  $|w|^2 + C \Sigma\ \xi_i$

Linear SVM: the Non-Separable Case

- Allow misclassifications (i.e., soft margin classifier) by introducing error variables ψ_k :

$$z_k(w^t\mathbf{x}_k + w_0) \geq 1 - \psi_k, \quad k = 1, 2, \ldots, n$$

Problem 3: Minimize $\frac{1}{2}\|w\|^2 + c\sum_{k=1}^{n}\psi_k$

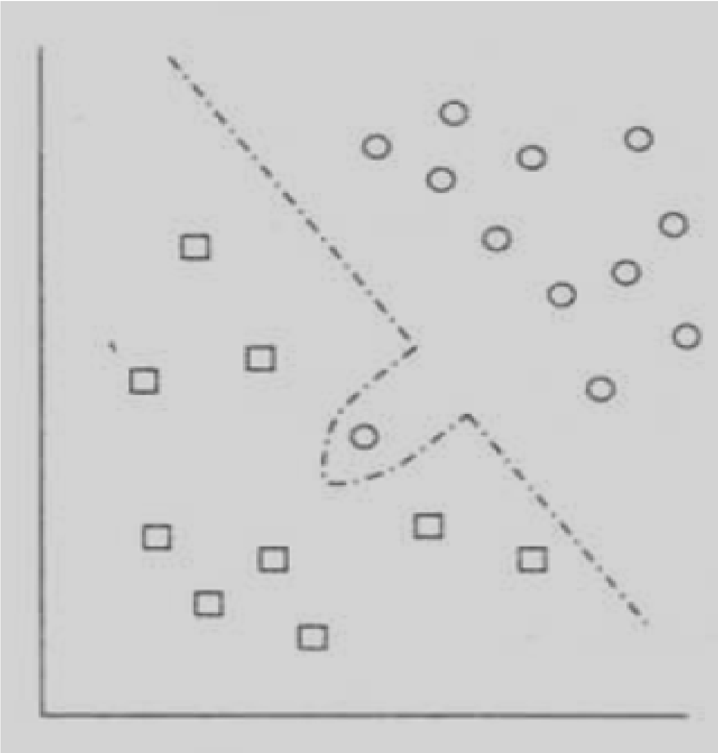
subject to $z_k(w^t x_k + w_0) \geq 1 - \psi_k, \quad k = 1, 2, \ldots, n$

- The result is a hyperplane that minimizes the sum of errors ψ_k while maximizing the margin for the correctly classified data.

Center of Signal and Image Processing
Georgia Institute of Technology
CSIP

Tradeoff between Margin & Misclassification

- The constant c controls the tradeoff between margin and misclassification errors (aims to prevent outliers from affecting the optimal hyperplane).



Linear SVM: Regularization

- We can reformulate "Problem 3" as maximizing the following problem (*dual problem*):

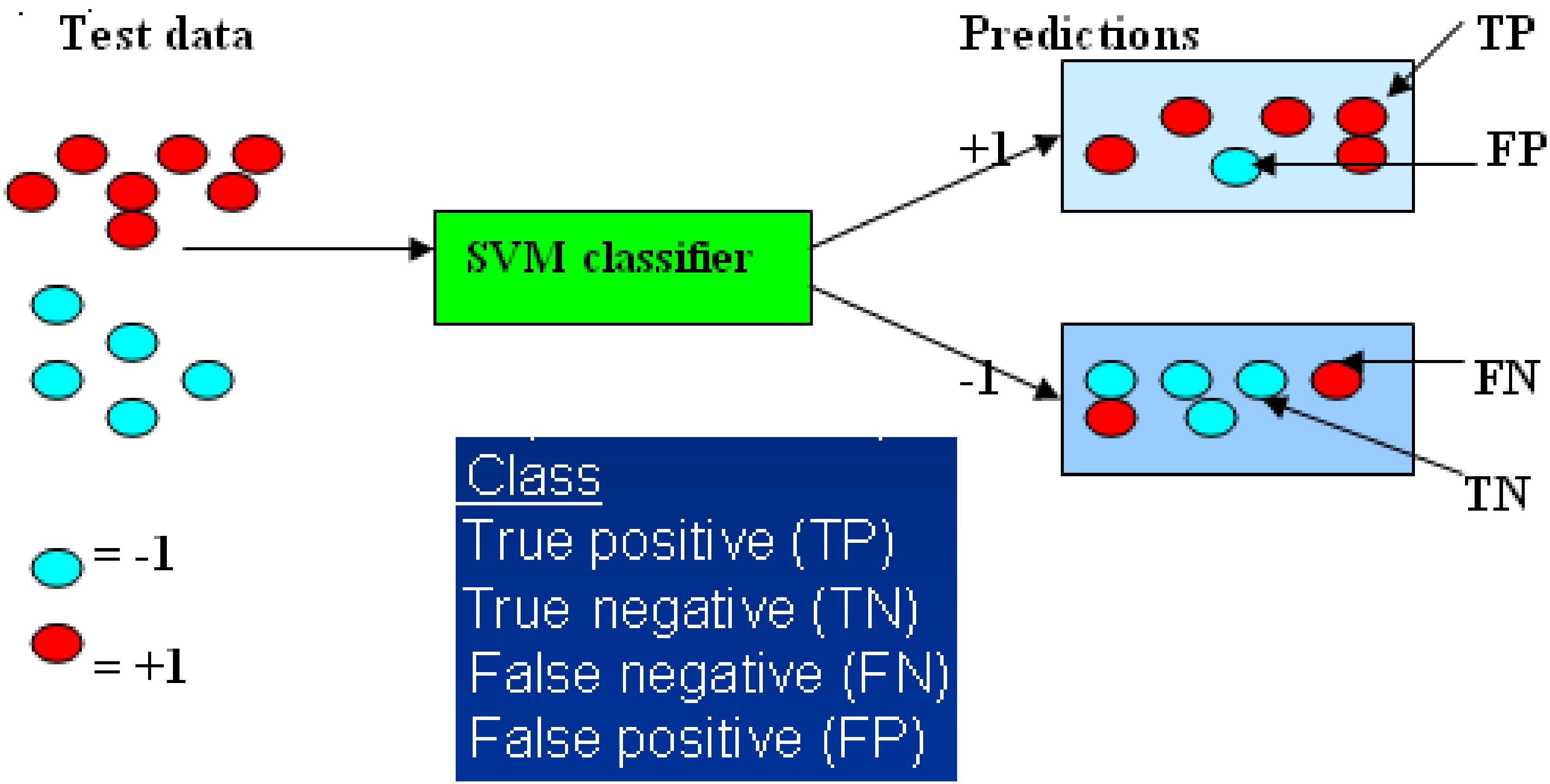
$$\textbf{Problem 4}: \text{Maximize} \sum_{k=1}^{n} \lambda_k - \frac{1}{2} \sum_{k,j}^{n} \lambda_k \lambda_j z_k z_j x_j^t x_k$$

$$\text{subject to} \sum_{k=1}^{n} z_k \lambda_k = 0 \text{ and } 0 \leq \lambda_k \leq c,\ k = 1, 2, \ldots, n$$

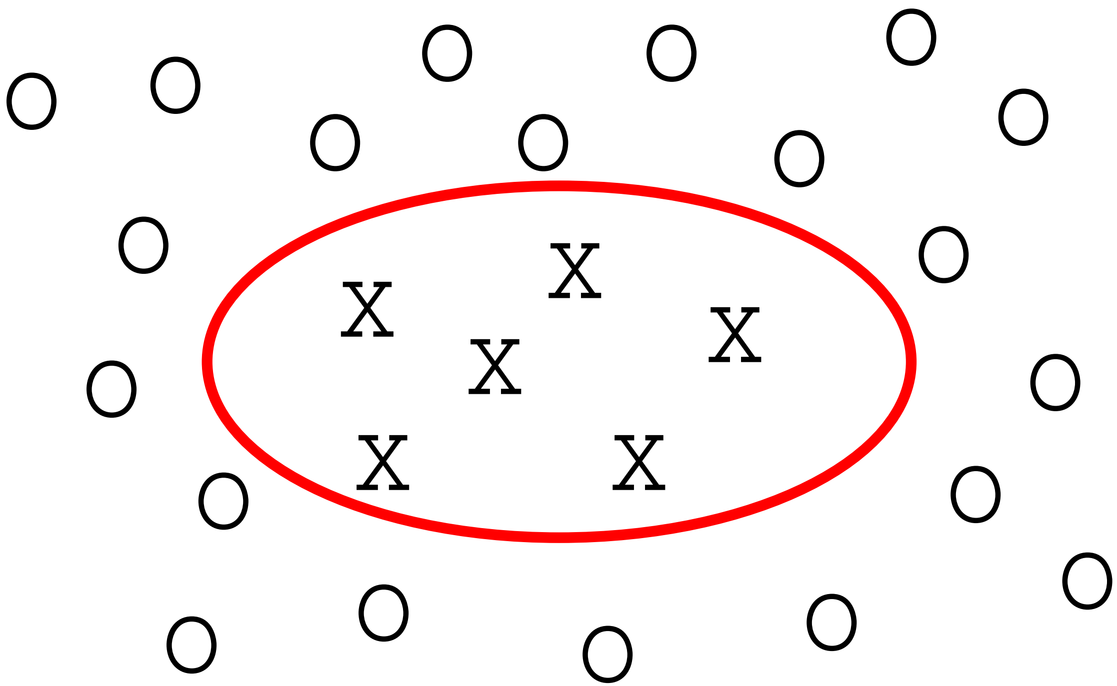
where the use of error variables ψ_k constraint the range of the Lagrange coefficients from 0 to c.

SVM-Based Classification

SVM Performance Measurement:



What if Surface is Non-Linear?



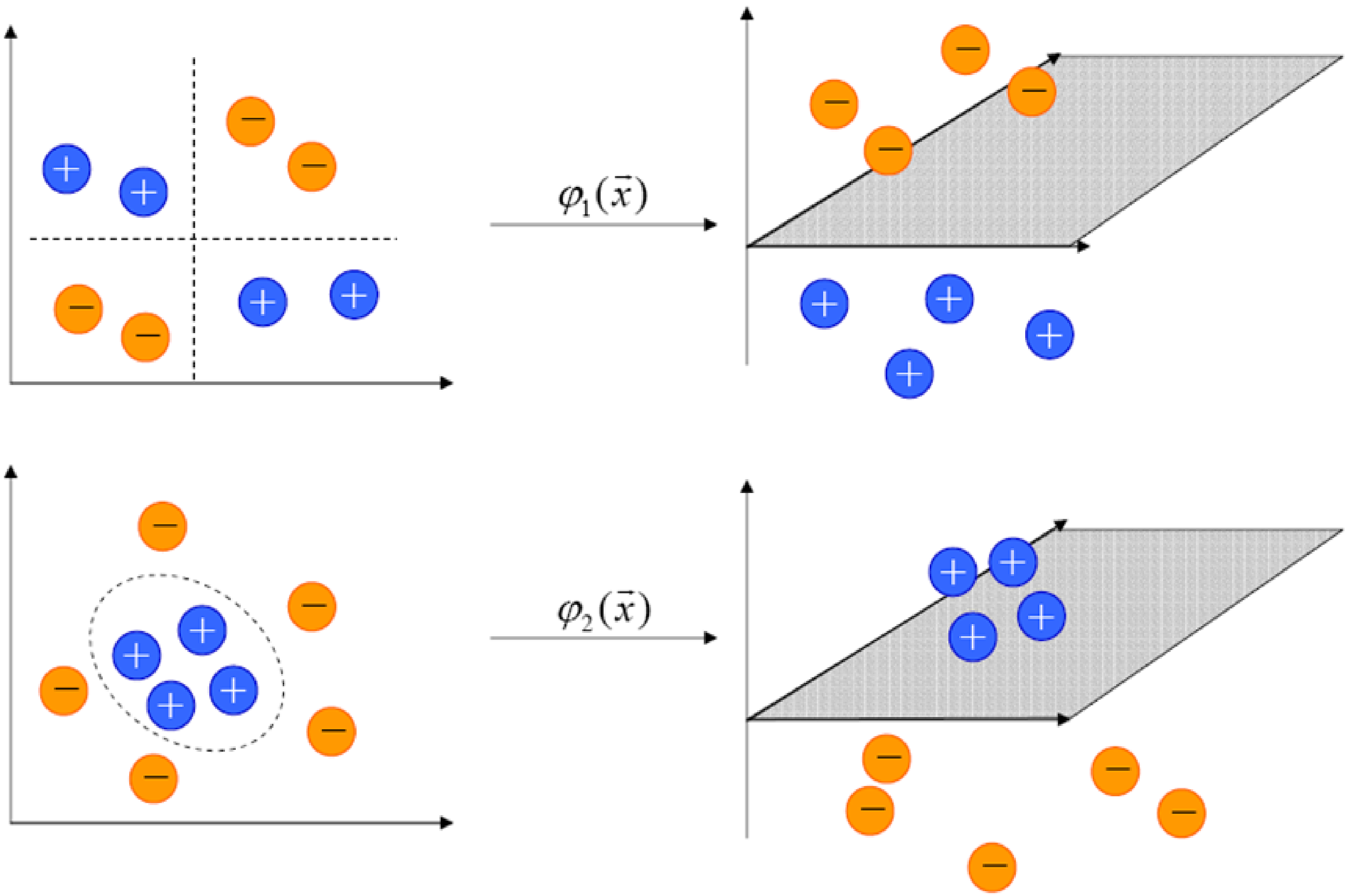
Nonlinear SVM

- Extending the above concepts to the non-linear vase relies on preprocessing the data to represent them in a much higher dimensionality space.

$$x_k \rightarrow \Phi(x_k)$$

- Using an appropriate nonlinear mapping $\Phi()$ to a sufficiently high dimensional space, data from two classes can always be separated by a hyperplane.

Nonlinear SVM (Cont'd)



Linearly Separable in Higher Dimension

Nonlinear SVM (Cont'd)

- The decision function for the optimal hyperplane is given by

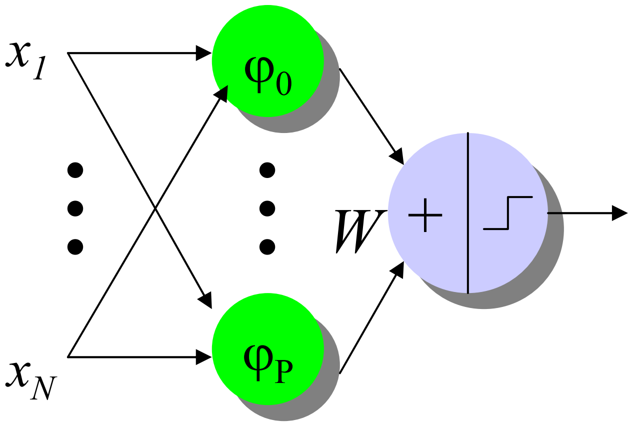
$$g(x) = \sum_{k=1}^{n} z_k \lambda_k (\Phi(x).\, \Phi(x_k)) + w_0$$

- The decision rule is the same as before:

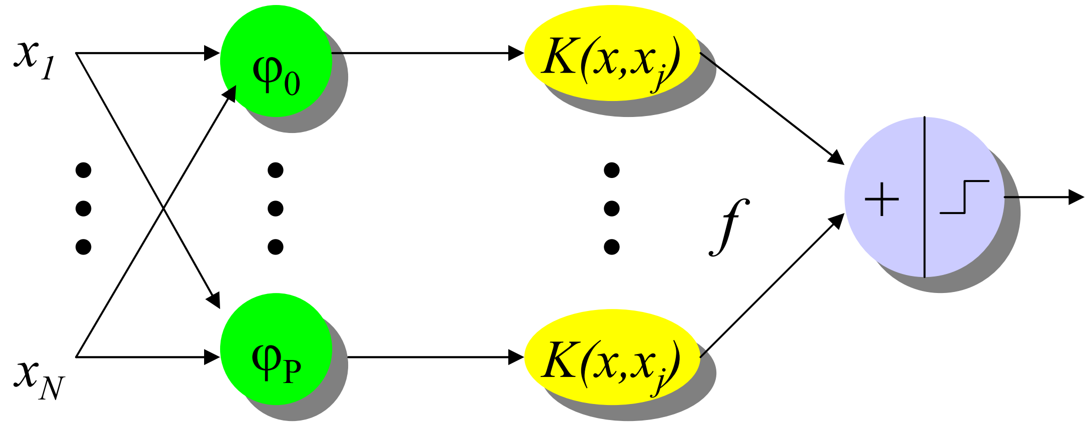
$$\textit{decide } \omega_1 \textit{ if } g(x)>0 \textit{ and } \omega_2 \textit{ if } g(x)<0$$

- The disadvantage of this approach is that the mapping $x_k \rightarrow \Phi(x_k)$ might be very computationally intensive to compute.

SVM Using Nonlinear Kernels



Nonlinear transform



Kernel evaluation

Using kernel, low dimensional feature vectors will be mapped to high dimensional (may be infinite dim) kernel feature space where the data are likely to be linearly separable

The Kernel Trick

"Given an algorithm which is formulated in terms of a positive definite kernel K_1, one can construct an alternative algorithm by replacing K_1 with another positive definite kernel K_2"

- SVMs can use the kernel trick

The Kernel Trick

- Compute dot products using a kernel function

$$K(\mathbf{x}, \mathbf{x}_k) = \Phi(\mathbf{x}).\Phi(\mathbf{x}_k)$$

- Advantages of using a kernel
  - No need to know $\Phi()$!!
  - The discriminant is given by:

$$\boxed{g(\mathbf{x}) = \sum_{k=1}^{n} z_k \lambda_k K(\mathbf{x}, \mathbf{x}_k) + w_0}$$

Kernels

- What does it *mean* to be a kernel?

  $K(x_1,x_2) = \langle \Phi(x_1), \Phi(x_2) \rangle$ for some Φ

- What does it *take* to be a kernel?
  - The Gram matrix $G_{ij} = K(x_i, x_j)$
  - Positive definite matrix
    - $\Sigma_{ij} c_i c_j G_{ij} \geq 0$ for $c_i, c_j \in \Re$
  - Positive definite kernel
    - For all samples of size *m*, induces a positive definite Gram matrix

The Kernel Trick (Cont'd)

Polynomial kernel: $K(x,y)=(x \cdot y)^d$

- The kernel trick implies that the computation remains feasible even if the feature space has very high dimensionality.

* It can be shown for the case of polynomial kernels that the data is mapped to a space of dimension $h = \binom{p+d-1}{d}$ where p is the original dimensionality.

* Suppose p=256 and $d = 4$, then h=183,181,376 !!

* A dot product in the high dimensional space would require $O(h)$ computations while the kernel requires only $O(p)$ computations.

Polynomial Kernel

- Consider a polynomial kernel

$$K(x,y) = (1 + x^T y)^2 = 1 + 2\sum_{i=1}^{m} x_i y_i + 2\sum_{i=1}^{m}\sum_{j=i+1}^{m} x_i y_i x_j y_j + \sum_{i=1}^{m} x_i^2 y_i^2$$

- Let $K(x,y) = \varphi^T(\mathbf{x})\,\varphi(\mathbf{y})$, then

$$\begin{aligned}\varphi(\mathbf{x}) = [1\ \ & x_1^2, \ldots, x_m^2, \sqrt{2}\,x_1, \ldots, \sqrt{2}x_m, \sqrt{2}\,x_1\,x_2, \ldots, \sqrt{2}\,x_1x_m, \\ & \sqrt{2}\,x_2\,x_3, \ldots, \sqrt{2}\,x_2x_m, \ldots, \sqrt{2}\,x_{m-1}x_m] \\ & = [1\ \varphi_1(x), \ldots, \varphi_p(x)]\end{aligned}$$

where $p = 1 + m + m + (m-1) + (m-2) + \ldots + 1 = (m+2)(m+1)/2$
Hence, using a kernel, a low dimensional pattern classification problem (with dimension m) is solved in a higher dimensional space (dimension $p+1$). But only $\phi_j(x)$ corresponding to support vectors are used for pattern classification!

Choice of Kernel Is Not Unique!

Example: consider $x \in R^2$, $\Phi(x) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix} \in R^3$, and $K(x, y) = (x.y)^2$

$$(x.y)^2 = (x_1y_1 + x_2y_2)^2$$

$$\Phi(x).\Phi(y) = x_1^2y_1^2 + 2x_1y_1x_2y_2 + x_2^2y_2^2 = (x_1y_1 + x_2y_2)^2$$

- Note that neither the mapping $\Phi()$ nor the high dimensional space are unique.

$$\Phi(x) = \frac{1}{\sqrt{2}}\begin{pmatrix} (x_1^2 - x_2^2) \\ 2x_1x_2 \\ (x_1^2 + x_2^2) \end{pmatrix} \in R^3 \quad \text{or} \quad \Phi(x) = \begin{pmatrix} x_1^2 \\ x_1x_2 \\ x_1x_2 \\ x_2^2 \end{pmatrix} \in R^4$$

Suitable Kernel Functions

- Kernel functions which can be expressed as a dot product in some space satisfy the *Mercer's* condition (see Burges' paper).

- The *Mercer's* condition does not tell us how to construct $\Phi()$ or even what the high dimensional space is.

- By using different kernel functions, SVM implement a variety of learning machines, some of which coincide with classical architectures (see below).

$$\textit{polynomial:}\ K(x, x_k) = (x \,.\, x_k)^d$$

$$\textit{sigmoidal:}\ K(x, x_k) = \tanh(v_k(x \,.\, x_k) + c_k)$$

(corresponds to a two-layer sigmoidal neural network)

$$\textit{Gaussian:}\ K(x, x_k) = \exp\left(\frac{-\|x - x_k\|^2}{2\sigma_k^2}\right)$$

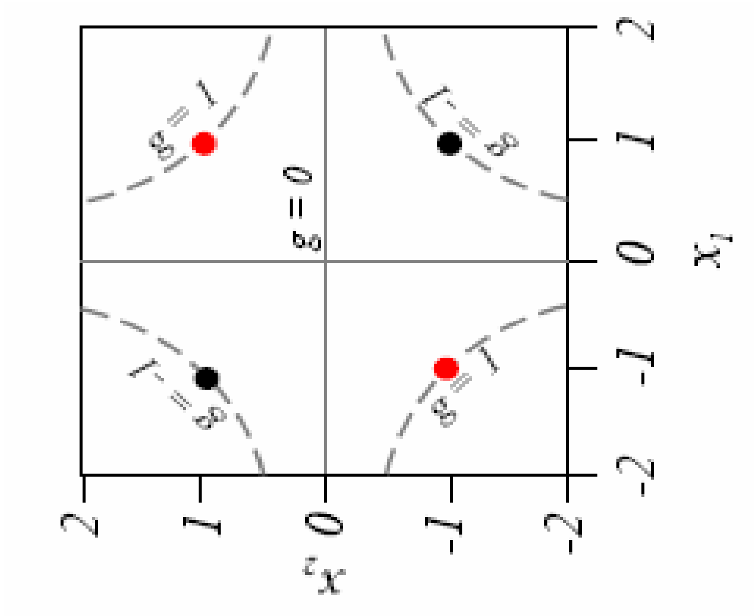
(corresponds to a radial basis function (RBF) neural network)

An Example

- Consider the XOR problem which is non-linearly separable:

(1,1) and (-1, -1) belong to ω_1

(1,-1) and (-1, 1) belong to ω_2

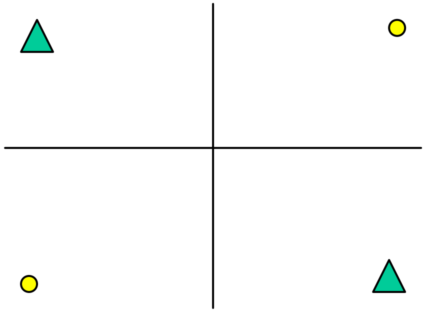


Numerical Example: XOR Problem

- Training samples:

  (–1 –1; –1), (–1 1 +1),

  (1 –1 +1), (1 1 –1)



$\boldsymbol{x} = [x_1, x_2]^T$. Use $K(\mathbf{x},\mathbf{y}) = (1 + \mathbf{x}^T\mathbf{y})^2$ one has

$\varphi(\mathbf{x}) = [1 \quad x_1^2 \quad x_2^2 \quad \sqrt{2}\, x_1, \ \sqrt{2}\, x_2, \ \sqrt{2}\, x_1 x_2]^T$

$$\Phi = \begin{bmatrix} 1 & 1 & 1 & -\sqrt{2} & -\sqrt{2} & \sqrt{2} \\ 1 & 1 & 1 & -\sqrt{2} & \sqrt{2} & -\sqrt{2} \\ 1 & 1 & 1 & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 1 & 1 & 1 & \sqrt{2} & \sqrt{2} & \sqrt{2} \end{bmatrix} \qquad K(\mathbf{x}_i, \mathbf{x}_j) = \Phi\Phi^T = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$$

Note dim[$\varphi(\mathbf{x})$] = 6 > dim[$\boldsymbol{x}$] = 2! dim(K) = N_s = # of support vectors.

XOR Problem (Cont'd)

- Note that $K(x_i, x_j)$ can be calculated directly without using Φ!

$$\text{e.g.}\quad K_{1,1} = \left(1 + \begin{bmatrix} -1 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix}\right)^2 = 9; \quad K_{1,2} = \left(1 + \begin{bmatrix} -1 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix}\right)^2 = 1$$

The corresponding Lagrange multiplier $\alpha = (1/8)[1 \;\; 1 \;\; 1 \;\; 1]^T$. $y = wT\varphi(\mathbf{x}) = -x_1 x_2$

$$W = \sum_{i=1}^{N} \alpha_i d_i \varphi(\mathbf{x}_i) = \Phi^T \begin{bmatrix} \alpha_1 d_1 & \alpha_2 d_2 & \cdots & \alpha_N d_N \end{bmatrix}^T$$

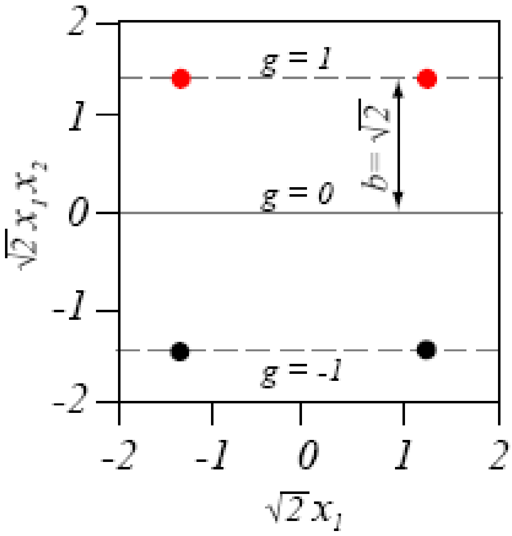
$$= \frac{1}{8}(-1)\varphi(\mathbf{x}_1) + \frac{1}{8}(1)\varphi(\mathbf{x}_2) + \frac{1}{8}(1)\varphi(\mathbf{x}_3) + \frac{1}{8}(-1)\varphi(\mathbf{x}_4) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}^T$$

(x_1, x_2)	$(-1, -1)$	$(-1, +1)$	$(+1, -1)$	$(+1, +1)$
$y = -x_1 x_2$	-1	$+1$	$+1$	-1

Example (Cont'd)

- Consider the following mapping (many other mappings could be used too):

$$y = \Phi(x) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_2 \\ x_2^2 \\ 1 \end{pmatrix}$$



Example (Cont'd)

- The above transformation maps x_k to a 6-dimensional space:

$$y_1 = \Phi(x_1) = \begin{pmatrix} 1 \\ \sqrt{2} \\ \sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \end{pmatrix} \qquad y_3 = \Phi(x_3) = \begin{pmatrix} 1 \\ -\sqrt{2} \\ \sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \end{pmatrix}$$

$$y_2 = \Phi(x_2) = \begin{pmatrix} 1 \\ \sqrt{2} \\ -\sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \end{pmatrix} \qquad y_4 = \Phi(x_4) = \begin{pmatrix} 1 \\ -\sqrt{2} \\ -\sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \end{pmatrix}$$

Example (Cont'd)

- We seek to maximize:

$$\sum_{k=1}^{4} \lambda_k - \frac{1}{2} \sum_{k,j}^{4} \lambda_k \lambda_j z_k z_j \Phi(x_j^t) \Phi(x_k)$$

$$\text{subject to } \sum_{k=1}^{4} z_k \lambda_k = 0, \;\; \lambda_k \geq 0, k = 1, 2, \ldots, 4$$

- The solution turns out to be:

$$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{8}$$

- Since all $\lambda_k \neq 0$, all x_k are support vectors !

Example (Cont'd)

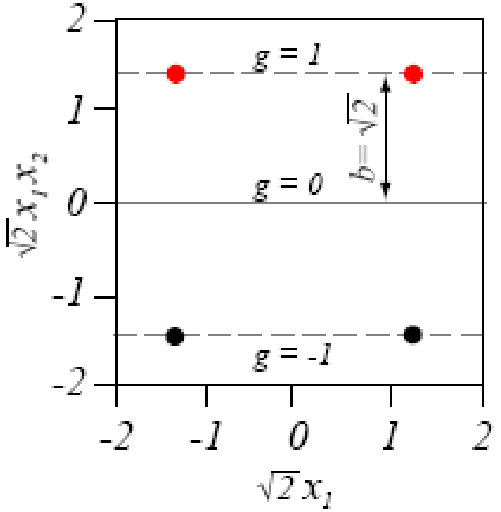
- We can now compute w:

$$w = \sum_{k=1}^{4} z_k \lambda_k \Phi(x_k) = \frac{1}{8}\begin{pmatrix} 1 \\ \sqrt{2} \\ \sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \end{pmatrix} - \frac{1}{8}\begin{pmatrix} 1 \\ \sqrt{2} \\ -\sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \end{pmatrix} + \frac{1}{8}\begin{pmatrix} 1 \\ -\sqrt{2} \\ \sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \end{pmatrix} - \frac{1}{8}\begin{pmatrix} 1 \\ -\sqrt{2} \\ -\sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 0 \\ 0 \\ \sqrt{2} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

- The solution for w_0 can be determined using any support vector, e.g., x_1:

$$w^t \Phi(x_1) + w_0 = z_1 \quad \text{or} \quad w_0 = z_1 - w^t x_1 = 0$$

Example (Cont'd)



- The margin b is computed as follows:

$$b = \frac{1}{\|w\|} = \sqrt{2}$$

- The decision function is the following:

$$\boxed{g(x) = w^t \Phi(x) + w_0 = x_1 x_2}$$

where we decide ω_1 if $g(x) > 0$ and ω_2 if $g(x) < 0$

Comments on SVMs

- Global optimization method, no local optima (i.e., based on exact optimization, not approximate methods)
- The performance of SVMs depends on the choice of the kernel and its parameters
  - The best choice of kernel for a given problem is still a research problem

Other Types of Kernels

type of SVM	$K(x,y)$	Comments
Polynomial learning machine	$(\boldsymbol{x}^T\boldsymbol{y} + 1)^p$	p: selected a priori
Radial basis function	$\exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{y}\|^2\right)$	σ^2: selected a priori
Two-layer perceptron	$\tanh(\beta_o \boldsymbol{x}^T\boldsymbol{y} + \beta_1)$	only some β_o and β_1 values are feasible.

What kernel is feasible? It must satisfy the "Mercer's theorem"!

Mercer's Theorem

- Let $K(\mathbf{x},\mathbf{y})$ be a continuous, symmetric kernel, defined on $a \leq \mathbf{x},\mathbf{y} \leq b$. $K(\mathbf{x},\mathbf{y})$ admits an eigen-function expansion

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

with $\lambda_i > 0$ for each i. This expansion converges absolutely and uniformly if and only if

$$\int_b^a \int_b^a K(\mathbf{x},\mathbf{y}) \psi(\mathbf{x}) \psi(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0$$

for all $\psi(\mathbf{x})$ such that: $\int_b^a \psi^2(\mathbf{x}) d\mathbf{x} < \infty$

Comments on SVMs (cont'd)

- Its complexity depends on the number of support vectors, not on the dimensionality of the transformed space
- Appear to avoid overfitting in high dimensional spaces and generalize well using a small training set
- The optimal design of multi-class SVM classifiers is a research topic

Summary

- Today's Class
  - SVM (Chapter 13)
- Next Classes
  - Kernel Methods (Chapter 6)
- Exercises: make sure you know the topics discussed and how to do all the exercises suggested in Chapter 12
- Reading Assignments
  - HTF, Chapters 6 & 12