



# CSE 4621

# Machine Learning

Lecture 15

**Md. Hasanul Kabir, PhD.**  
Professor, CSE Department  
Islamic University of Technology (IUT)



---

# Support Vector Machine & Its Applications

---

A portion (1/3) of  
the slides are taken from  
Prof. Andrew Moore's  
SVM tutorial at

<http://www.cs.cmu.edu/~awm/tutorials>

**Mingyue Tan**

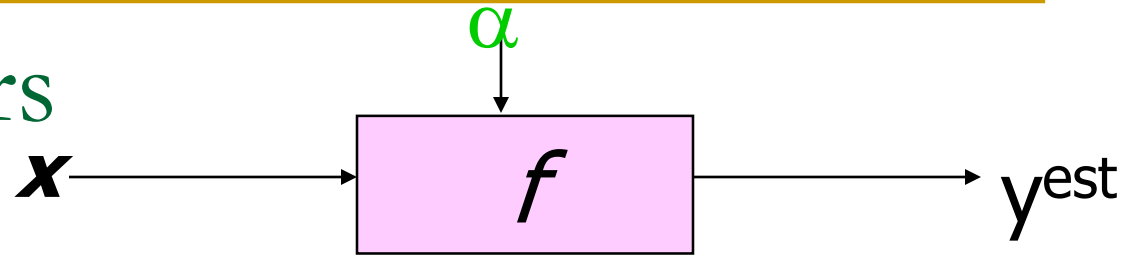
The University of British Columbia  
Nov 26, 2004

---

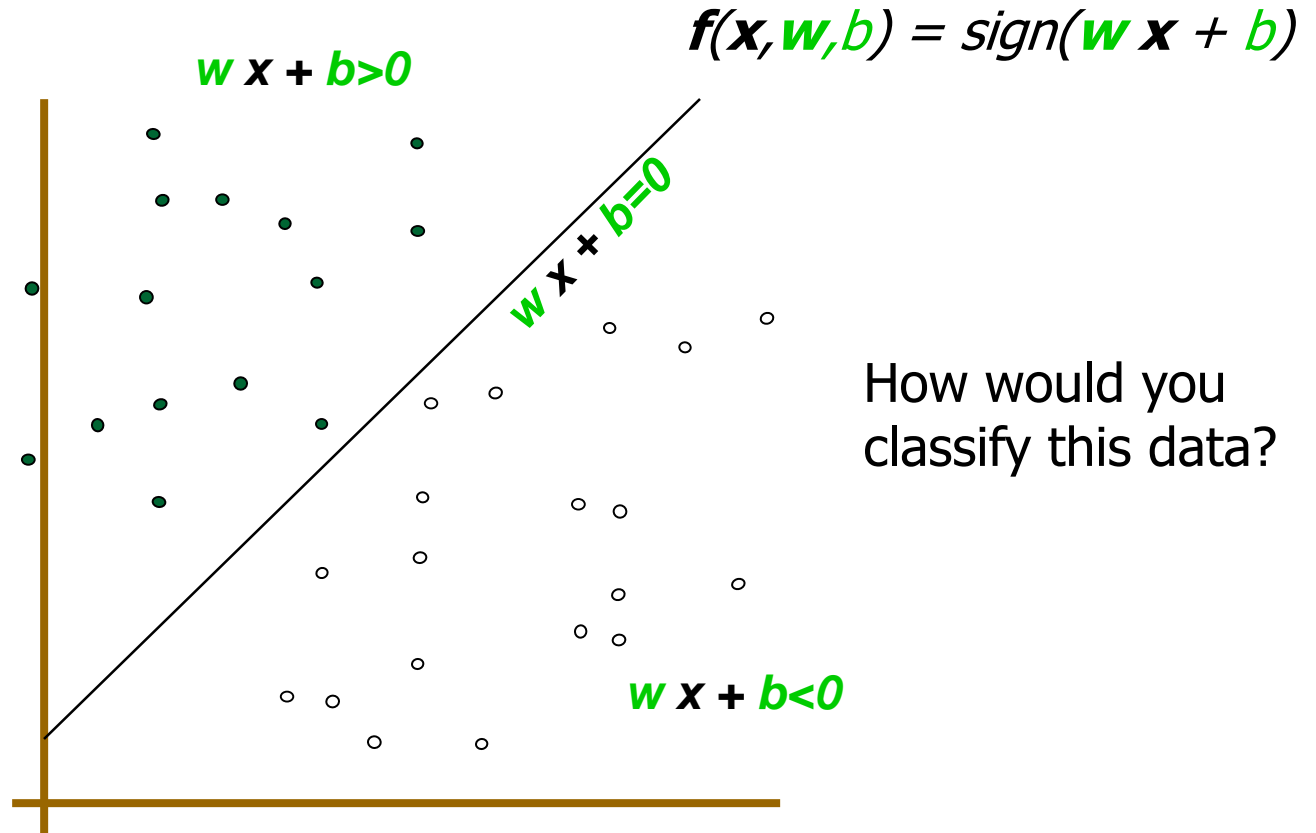
# Overview

- Intro. to Support Vector Machines (SVM)
  - Properties of SVM
  - Applications
    - Gene Expression Data Classification
    - Text Categorization *if time permits*
  - Discussion
-

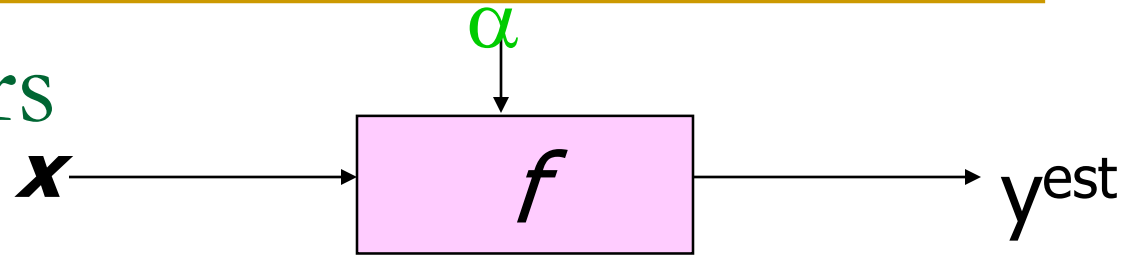
# Linear Classifiers



- denotes +1
- denotes -1

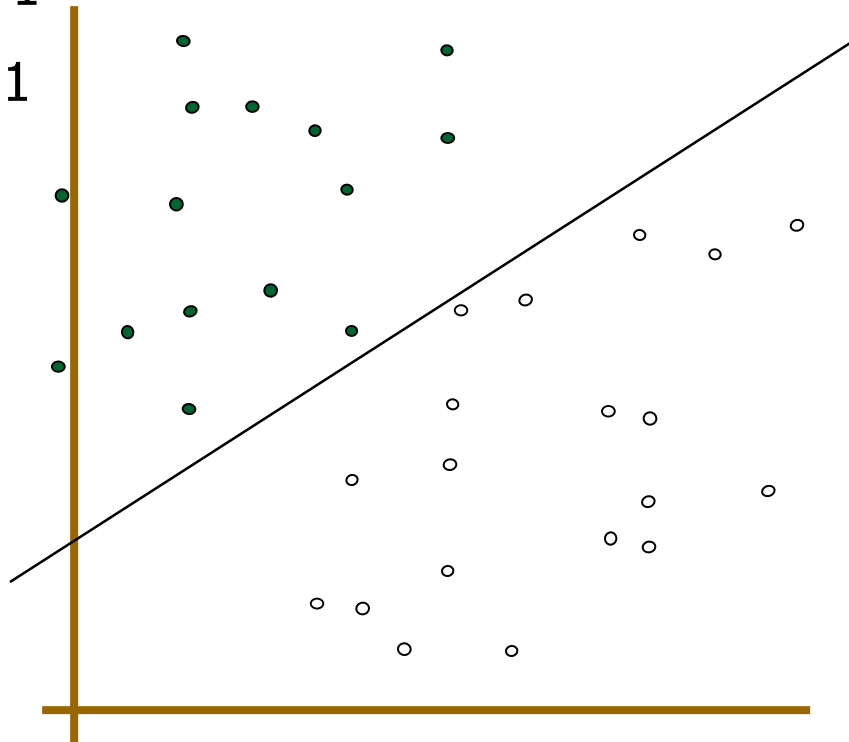


# Linear Classifiers



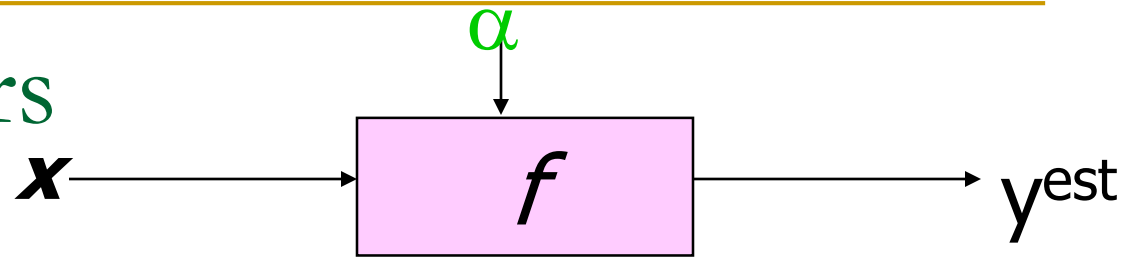
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \mathbf{x} + b)$$

- denotes +1
- denotes -1

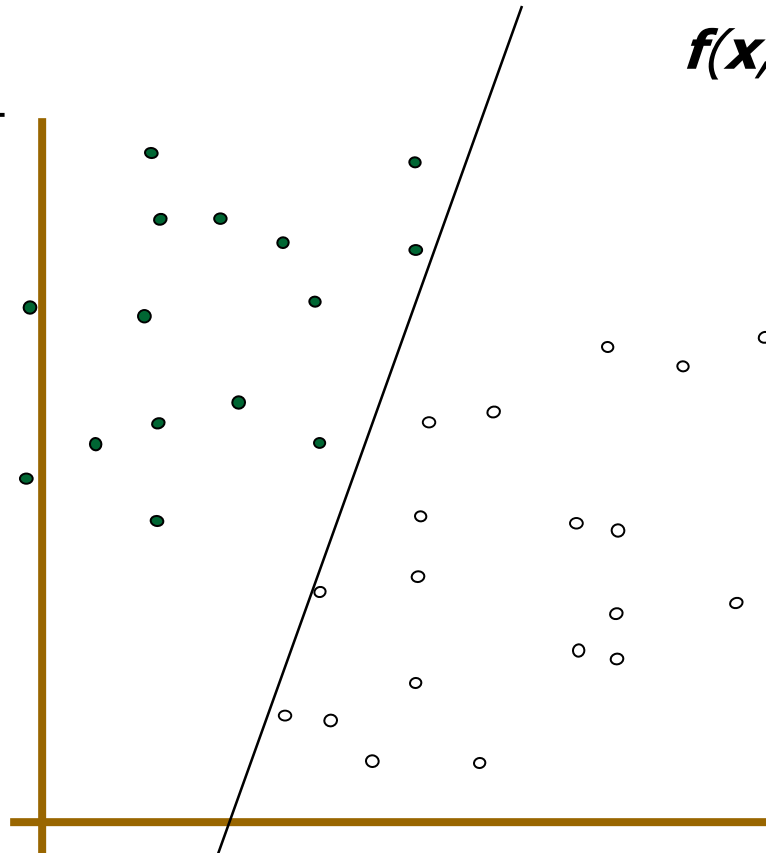


How would you classify this data?

# Linear Classifiers



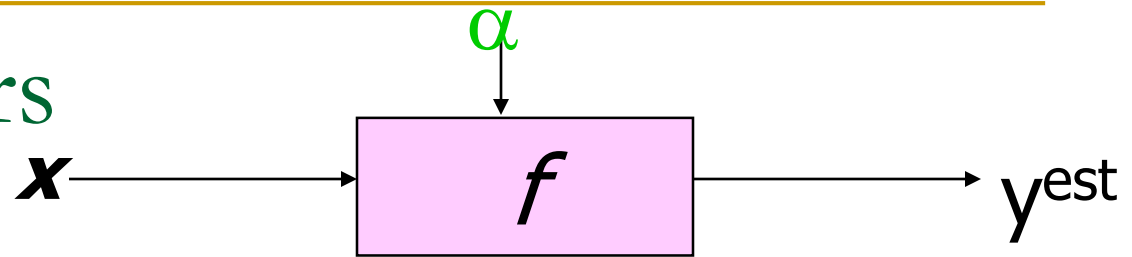
- denotes +1
- denotes -1



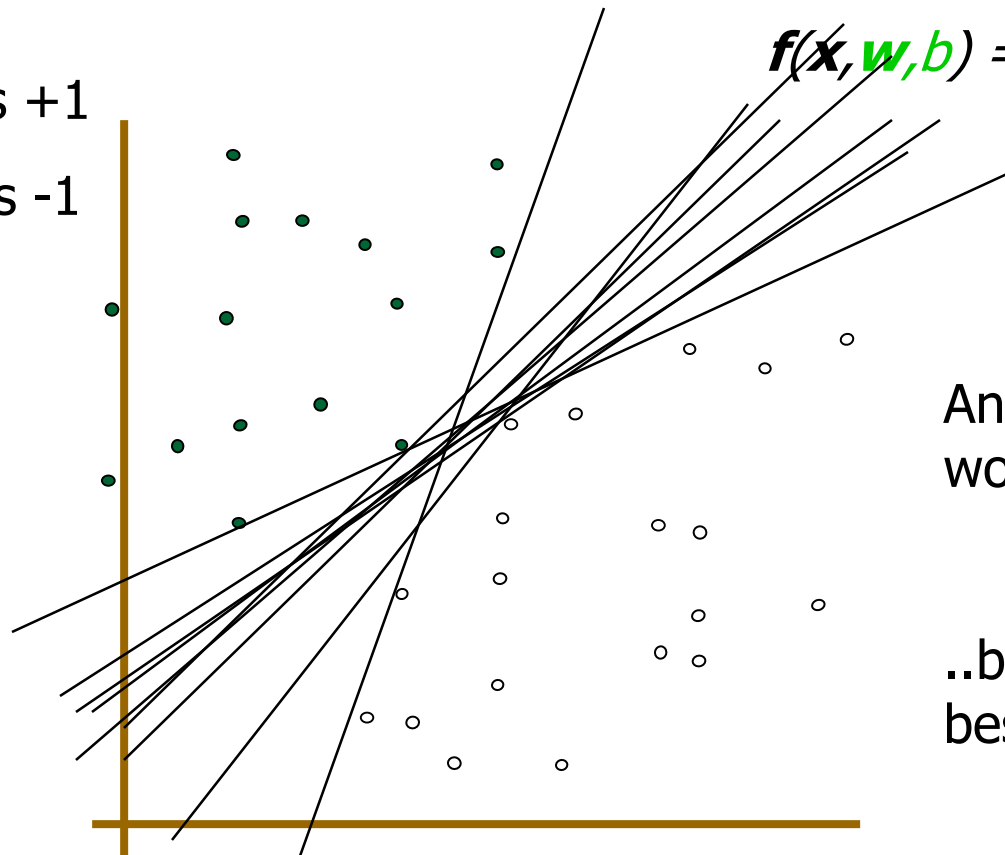
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \mathbf{x} + b)$$

How would you classify this data?

# Linear Classifiers



- denotes +1
- denotes -1

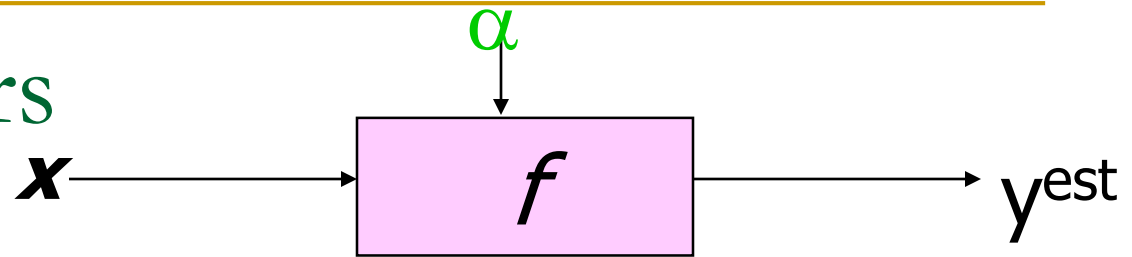


$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \mathbf{x} + b)$$

Any of these  
would be fine..

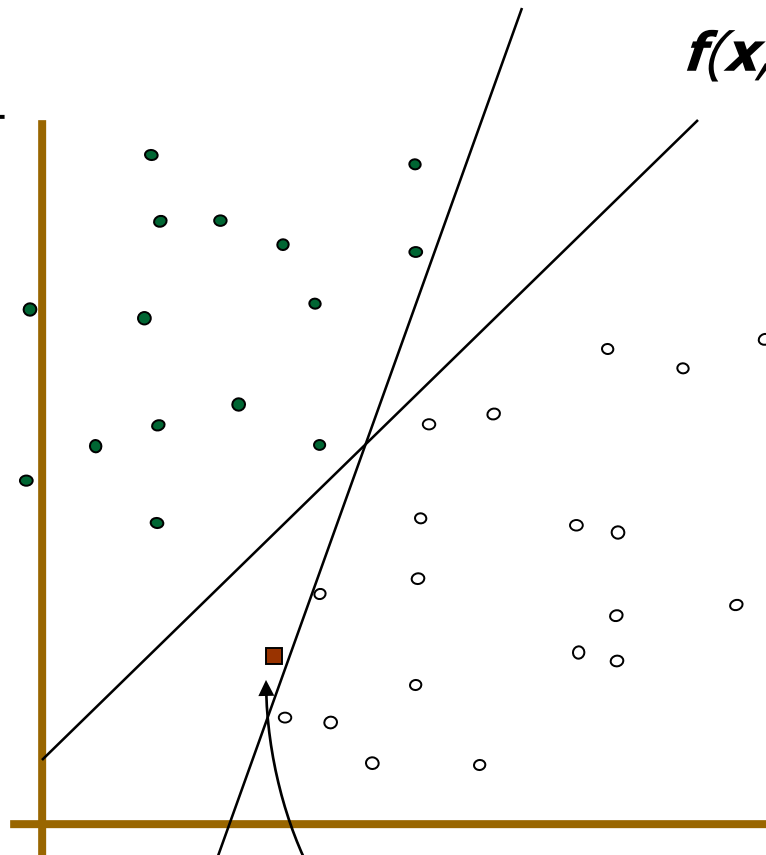
..but which is  
best?

# Linear Classifiers



- denotes +1
- denotes -1

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \mathbf{x} + b)$$

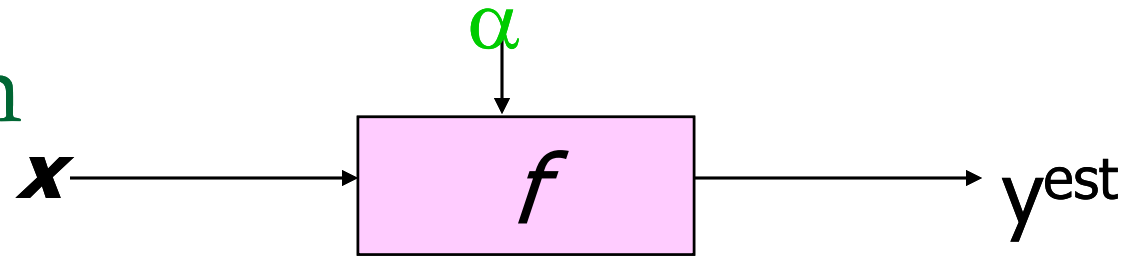


How would you classify this data?

Misclassified  
to +1 class

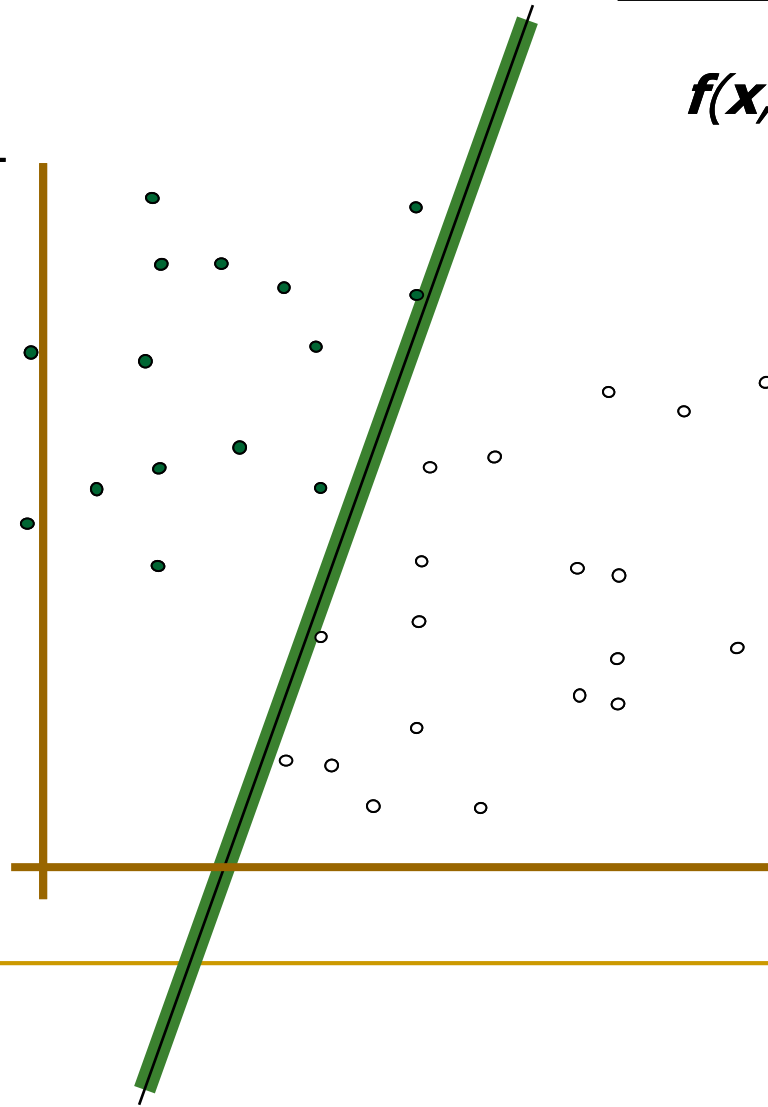


# Classifier Margin



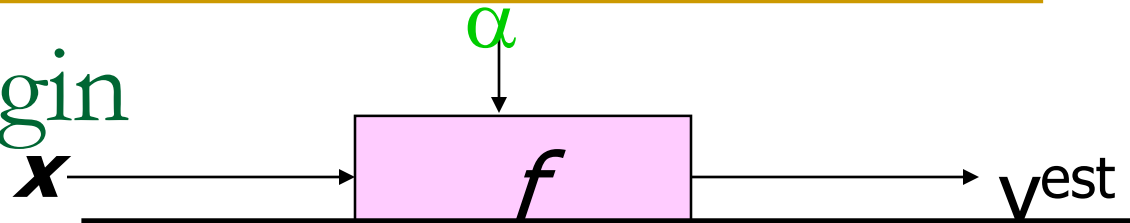
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \mathbf{x} + b)$$

- denotes +1
- denotes -1



Define the **margin** of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

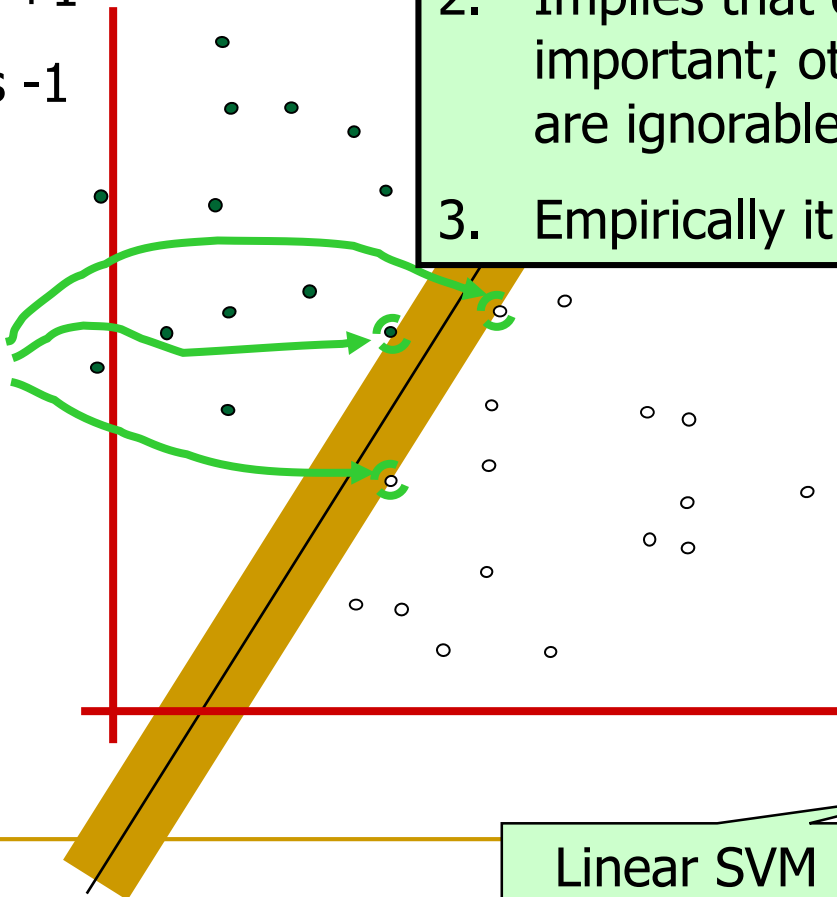
# Maximum Margin



1. Maximizing the margin is good according to intuition and PAC theory
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very very well.

- denotes +1
- denotes -1

**Support Vectors**  
are those datapoints that the margin pushes up against

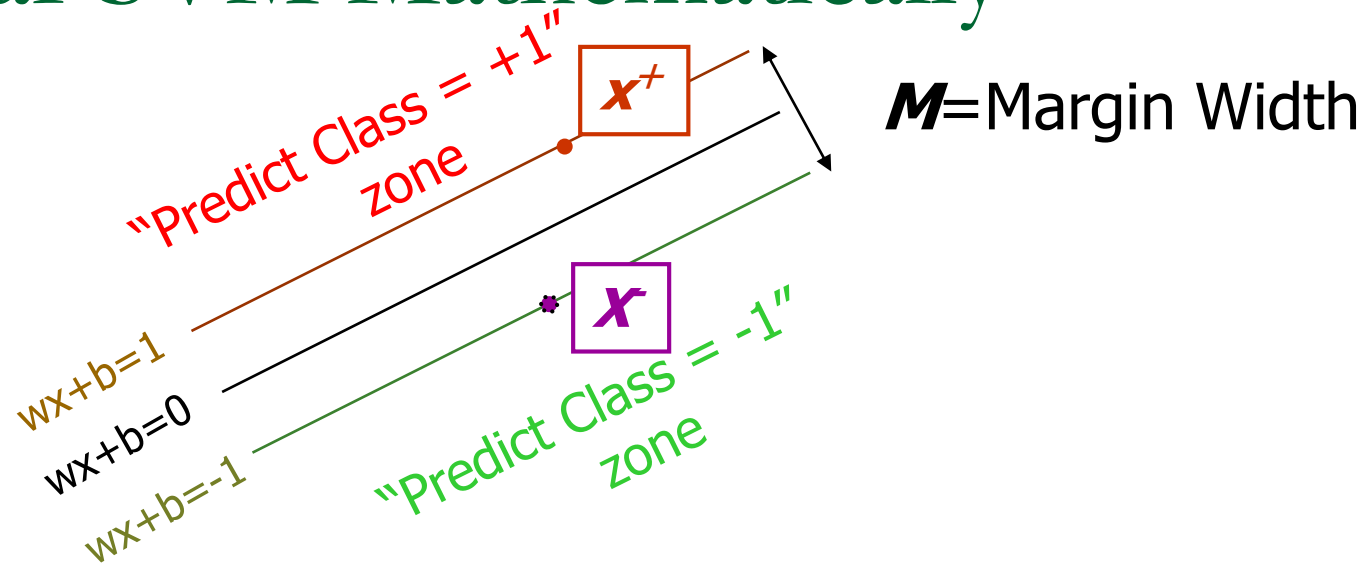


linear classifier  
with the, um,  
maximum margin.

This is the  
simplest kind of  
SVM (Called an  
LSVM)

Linear SVM

# Linear SVM Mathematically



What we know:

- $\mathbf{w} \cdot \mathbf{x}^+ + b = +1$
- $\mathbf{w} \cdot \mathbf{x}^- + b = -1$
- $\mathbf{w} \cdot (\mathbf{x}^+ - \mathbf{x}^-) = 2$

$$M = \frac{(\mathbf{x}^+ - \mathbf{x}^-) \cdot \mathbf{w}}{|\mathbf{w}|} = \frac{2}{|\mathbf{w}|}$$

# Linear SVM Mathematically

■ Goal: 1) **Correctly classify all training data**

$$wx_i + b \geq 1 \quad \text{if } y_i = +1$$

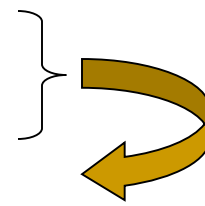
$$wx_i + b \leq -1 \quad \text{if } y_i = -1$$

$$y_i(wx_i + b) \geq 1 \quad \text{for all } i$$

2) **Maximize the Margin**

**same as minimize**

$$M = \frac{2}{|w|}$$
$$\frac{1}{2} w^t w$$



■ We can formulate a **Quadratic Optimization Problem** and solve for **w** and **b**

■ Minimize  $\Phi(w) = \frac{1}{2} w^t w$

subject to  $y_i(wx_i + b) \geq 1 \quad \forall i$

# Solving the Optimization Problem

Find  $\mathbf{w}$  and  $b$  such that

$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$  is minimized;

and for all  $\{(\mathbf{x}_i, y_i)\}$ :  $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

- Need to optimize a *quadratic* function subject to *linear* constraints.
- Quadratic optimization problems are a well-known class of mathematical programming problems, and many (rather intricate) algorithms exist for solving them.
- The solution involves constructing a *dual problem* where a *Lagrange multiplier*  $\alpha_i$  is associated with every constraint in the primary problem:

Find  $\alpha_1 \dots \alpha_N$  such that

$Q(\boldsymbol{\alpha}) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$  is maximized and

(1)  $\sum \alpha_i y_i = 0$

(2)  $\alpha_i \geq 0$  for all  $\alpha_i$

# The Optimization Problem Solution

- The solution has the form:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \quad b = y_k - \mathbf{w}^T \mathbf{x}_k \text{ for any } \mathbf{x}_k \text{ such that } \alpha_k \neq 0$$

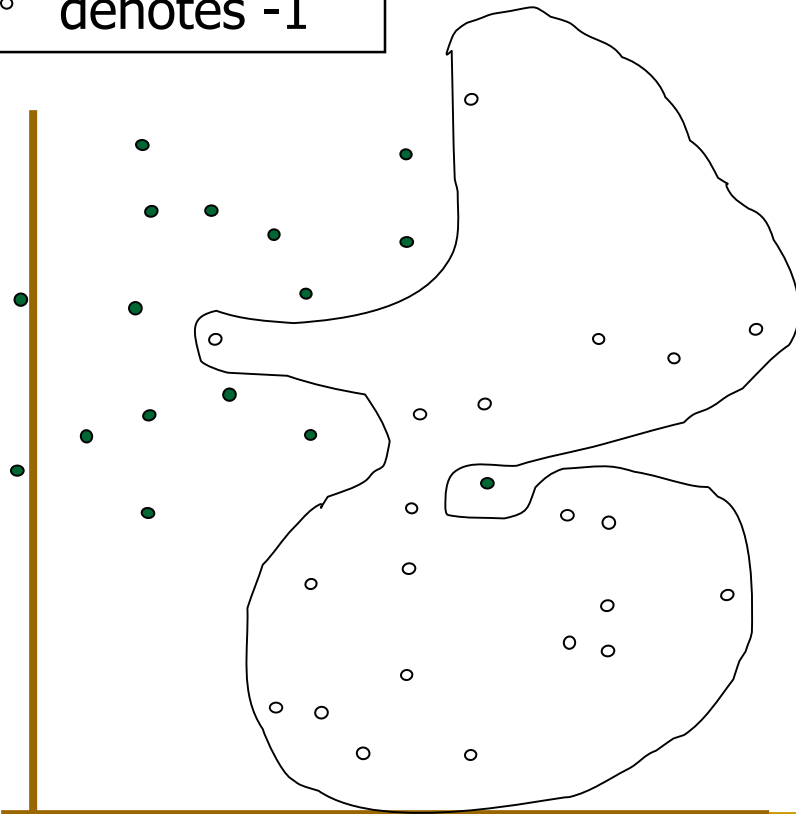
- Each non-zero  $\alpha_i$  indicates that corresponding  $\mathbf{x}_i$  is a support vector.
- Then the **classifying function** will have the form:

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point  $\mathbf{x}$  and the support vectors  $\mathbf{x}_i$  – we will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products  $\mathbf{x}_i^T \mathbf{x}_j$  between all pairs of training points.

# Dataset with noise

- denotes +1
- denotes -1

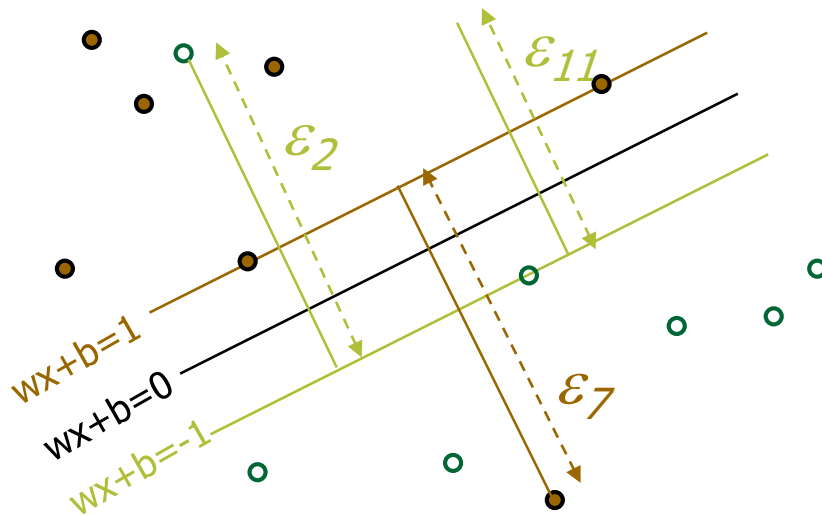


- **Hard Margin:** So far we require all data points be classified correctly
  - No training error
- **What if the training set is noisy?**
  - **Solution 1:** use very powerful kernels

**OVERFITTING!**

# Soft Margin Classification

**Slack variables  $\xi_i$**  can be added to allow misclassification of difficult or noisy examples.



What should our quadratic optimization criterion be?

Minimize

$$\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{k=1}^R \xi_k$$



# Hard Margin v.s. Soft Margin

- **The old formulation:**

Find  $\mathbf{w}$  and  $b$  such that

$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$  is minimized and for all  $\{(\mathbf{x}_i, y_i)\}$   
 $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

- **The new formulation incorporating slack variables:**

Find  $\mathbf{w}$  and  $b$  such that

$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_i$  is minimized and for all  $\{(\mathbf{x}_i, y_i)\}$   
 $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$  and  $\xi_i \geq 0$  for all  $i$

- **Parameter  $C$  can be viewed as a way to control overfitting.**

# Linear SVMs: Overview

- The classifier is a *separating hyperplane*.
- Most “important” training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points  $\mathbf{x}_i$  are support vectors with non-zero Lagrangian multipliers  $\alpha_i$ .
- Both in the dual formulation of the problem and in the solution training points appear only inside dot products:

Find  $\alpha_1 \dots \alpha_N$  such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$  is maximized and

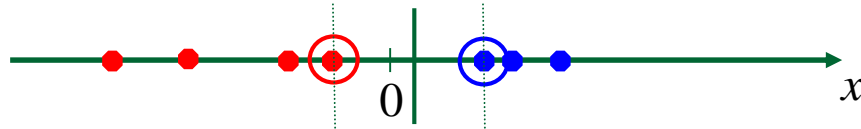
(1)  $\sum \alpha_i y_i = 0$

(2)  $0 \leq \alpha_i \leq C$  for all  $\alpha_i$

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

# Non-linear SVMs

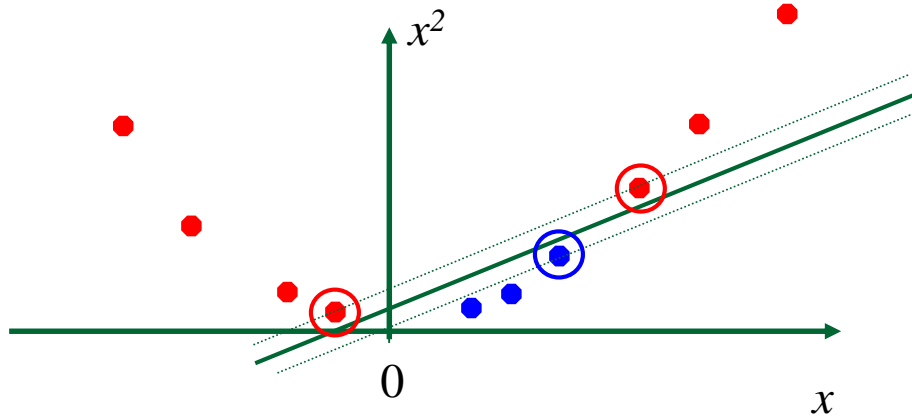
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

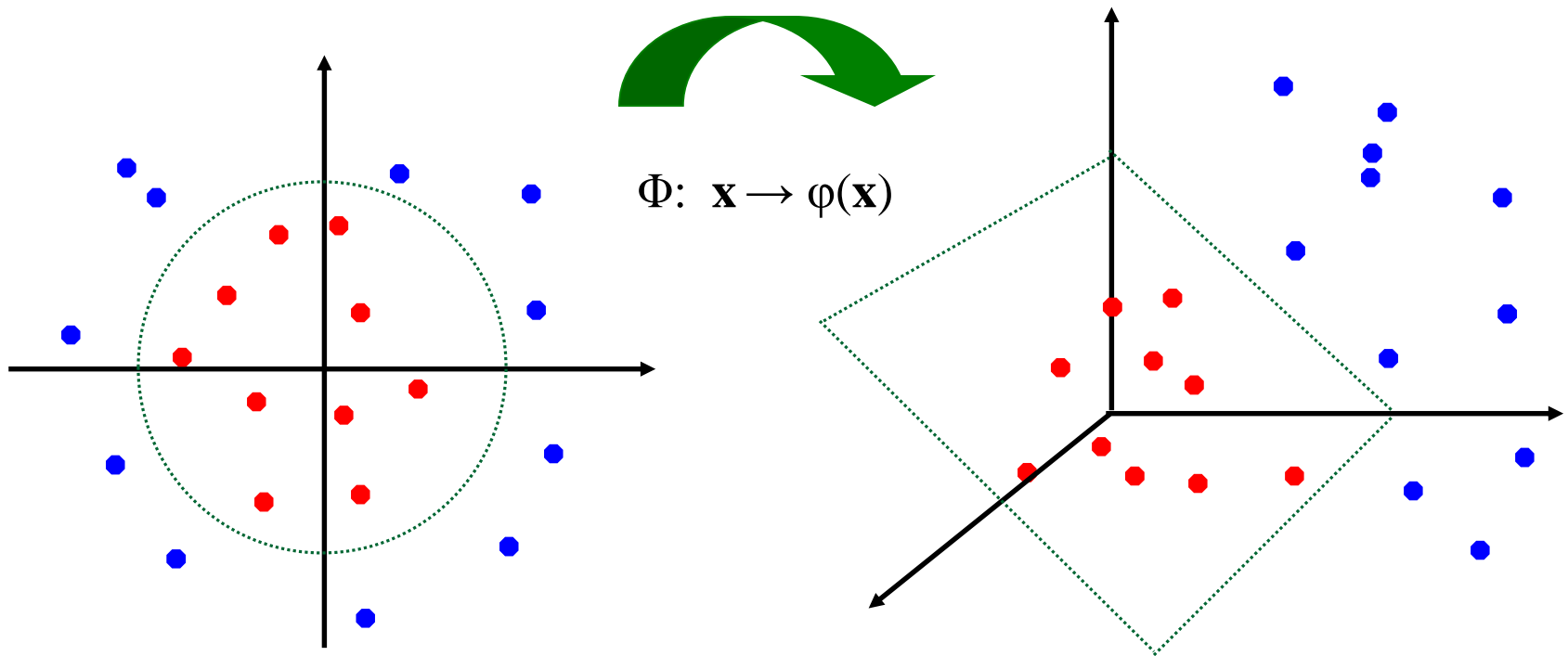


- How about... mapping data to a higher-dimensional space:



# Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



# The “Kernel Trick”

- The linear classifier relies on dot product between vectors  $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- If every data point is mapped into high-dimensional space via some transformation  $\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$ , the dot product becomes:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- Example:

2-dimensional vectors  $\mathbf{x} = [x_1 \ x_2]$ ; let  $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$ ,

Need to show that  $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ :

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2, \\ &= 1 + x_{i1}^2 x_{j1}^2 + 2 x_{i1} x_{j1} x_{i2} x_{j2} + x_{i2}^2 x_{j2}^2 + 2 x_{i1} x_{j1} + 2 x_{i2} x_{j2} \\ &= [1 \ x_{i1}^2 \ \sqrt{2} x_{i1} x_{i2} \ x_{i2}^2 \ \sqrt{2} x_{i1} \ \sqrt{2} x_{i2}]^T [1 \ x_{j1}^2 \ \sqrt{2} x_{j1} x_{j2} \ x_{j2}^2 \ \sqrt{2} x_{j1} \ \sqrt{2} x_{j2}] \\ &= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j), \quad \text{where } \phi(\mathbf{x}) = [1 \ x_1^2 \ \sqrt{2} x_1 x_2 \ x_2^2 \ \sqrt{2} x_1 \ \sqrt{2} x_2] \end{aligned}$$

# What Functions are Kernels?

- For some functions  $K(\mathbf{x}_i, \mathbf{x}_j)$  checking that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x}_j) \text{ can be cumbersome.}$$

- Mercer's theorem:

*Every semi-positive definite symmetric function is a kernel*

- Semi-positive definite symmetric functions correspond to a semi-positive definite symmetric Gram matrix:

$K =$

$K(\mathbf{x}_1, \mathbf{x}_1)$	$K(\mathbf{x}_1, \mathbf{x}_2)$	$K(\mathbf{x}_1, \mathbf{x}_3)$	$\dots$	$K(\mathbf{x}_1, \mathbf{x}_N)$
$K(\mathbf{x}_2, \mathbf{x}_1)$	$K(\mathbf{x}_2, \mathbf{x}_2)$	$K(\mathbf{x}_2, \mathbf{x}_3)$		$K(\mathbf{x}_2, \mathbf{x}_N)$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$K(\mathbf{x}_N, \mathbf{x}_1)$	$K(\mathbf{x}_N, \mathbf{x}_2)$	$K(\mathbf{x}_N, \mathbf{x}_3)$	$\dots$	$K(\mathbf{x}_N, \mathbf{x}_N)$

# Examples of Kernel Functions

- Linear:  $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Polynomial of power  $p$ :  $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
- Gaussian (radial-basis function network):
$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$
- Sigmoid:  $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^T \mathbf{x}_j + \beta_1)$

# Non-linear SVMs Mathematically

- **Dual problem formulation:**

Find  $\alpha_1 \dots \alpha_N$  such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$  is maximized and

(1)  $\sum \alpha_i y_i = 0$

(2)  $\alpha_i \geq 0$  for all  $\alpha_i$

- **The solution is:**

$$f(\mathbf{x}) = \sum \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b$$

- **Optimization techniques for finding  $\alpha_i$ 's remain the same!**



---

# Nonlinear SVM - Overview

- SVM locates a separating hyperplane in the feature space and classify points in that space
  - It does not need to represent the space explicitly, simply by defining a kernel function
  - The kernel function plays the role of the dot product in the feature space.
-

# Properties of SVM

- **Flexibility in choosing a similarity function**
- **Sparseness of solution when dealing with large data sets**
  - only support vectors are used to specify the separating hyperplane
- **Ability to handle large feature spaces**
  - complexity does not depend on the dimensionality of the feature space
- **Overfitting can be controlled by soft margin approach**
- **Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution**
- **Feature Selection**

---

# SVM Applications

- **SVM has been used successfully in many real-world problems**
    - text (and hypertext) categorization
    - image classification
    - bioinformatics (Protein classification, Cancer classification)
    - hand-written character recognition
-

# Application 1: Cancer Classification

- High Dimensional

- $p > 1000$ ;  $n < 100$

- Imbalanced

- less positive samples

$$K[x, x] = k(x, x) + \lambda \frac{n^+}{N}$$

- Many irrelevant features

- Noisy

SVM is sensitive to noisy (mis-labeled) data ☹

Genes				
Patients	g-1	g-2	.....	g-p
P-1				
p-2				
.....				
p-n				

## FEATURE SELECTION

In the linear case,  
 $w_i^2$  gives the ranking of dim  $i$

# Weakness of SVM

- **It is sensitive to noise**

- A relatively small number of mislabeled examples can dramatically decrease the performance

- **It only considers two classes**

- how to do multi-class classification with SVM?

- Answer:

1) with output arity  $m$ , learn  $m$  SVM's

- SVM 1 learns “Output==1” vs “Output != 1”
- SVM 2 learns “Output==2” vs “Output != 2”
- :
- SVM  $m$  learns “Output== $m$ ” vs “Output !=  $m$ ”

2) To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

---

# Application 2: Text Categorization

- Task: The classification of natural text (or hypertext) documents into a fixed number of predefined categories based on their content.
    - email filtering, web searching, sorting documents by topic, etc..
  - A document can be assigned to more than one category, so this can be viewed as a series of binary classification problems, one for each category
-

# Representation of Text

IR's vector space model (aka bag-of-words representation)

- A doc is represented by a vector indexed by a pre-fixed set or dictionary of terms
- Values of an entry can be binary or weights

$$\phi_i(x) = \frac{tf_i \log(idf_i)}{\kappa},$$

- Normalization, stop words, word stems
- Doc  $x \Rightarrow \boldsymbol{\varphi}(x)$

# Text Categorization using SVM

- The distance between two documents is  $\phi(x) \cdot \phi(z)$
- $K(x,z) = \langle \phi(x) \cdot \phi(z) \rangle$  is a valid kernel, SVM can be used with  $K(x,z)$  for discrimination.
- Why SVM?
  - High dimensional input space
  - Few irrelevant features (dense concept)
  - Sparse document vectors (sparse instances)
  - Text categorization problems are linearly separable



# Some Issues

## ■ Choice of kernel

- Gaussian or polynomial kernel is default
- if ineffective, more elaborate kernels are needed
- domain experts can give assistance in formulating appropriate similarity measures

## ■ Choice of kernel parameters

- e.g.  $\sigma$  in Gaussian kernel
- $\sigma$  is the distance between closest points with different classifications
- In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.

## ■ Optimization criterion – Hard margin v.s. Soft margin

- a lengthy series of experiments in which various parameters are tested

---

# Additional Resources

- **An excellent tutorial on VC-dimension and Support Vector Machines:**  
C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):955-974, 1998.
- **The VC/SRM/SVM Bible:**  
Statistical Learning Theory by Vladimir Vapnik, Wiley-Interscience; 1998

<http://www.kernel-machines.org/>

---

# Reference

- **Support Vector Machine Classification of Microarray Gene Expression Data**, Michael P. S. Brown William Noble Grundy, David Lin, Nello Cristianini, Charles Sugnet, Manuel Ares, Jr., David Haussler
- [www.cs.utexas.edu/users/mooney/cs391L/svm.ppt](http://www.cs.utexas.edu/users/mooney/cs391L/svm.ppt)
- **Text categorization with Support Vector Machines:  
learning with many relevant features**  
T. Joachims, ECML - 98