# An Introduction of
# Support Vector Machine

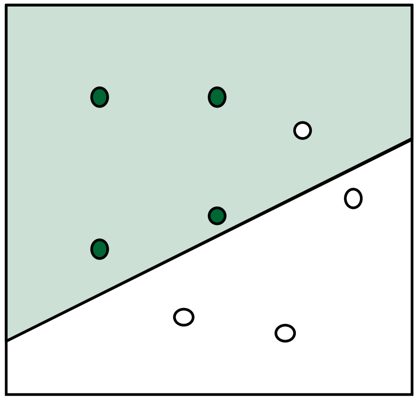
In part from of Jinwei Gu

# Support Vector Machine (SVM)

- A classifier derived from statistical learning theory by Vapnik, et al. in 1992
- SVM became famous when, using images as input, it gave accuracy comparable to neural networks in a handwriting recognition task
- Currently, SVM is widely used in object detection & recognition, content-based image retrieval, text recognition, biometrics, speech recognition, etc.
- Still one of the best non-deep methods – in many tasks, comparable performance

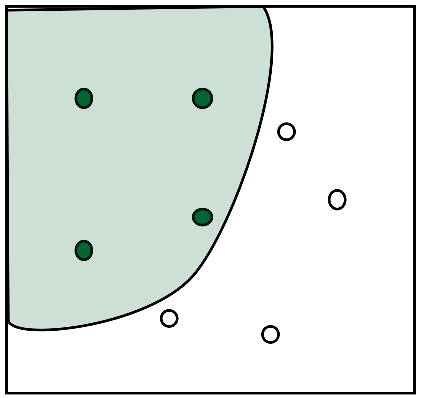
# Classification Functions learned by SVM

- It can be an arbitrary function of $y=f(x)$, such as:



Linear Functions (e.g percepron)

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$



Nonlinear Functions (e.g. feed forward neural nets)

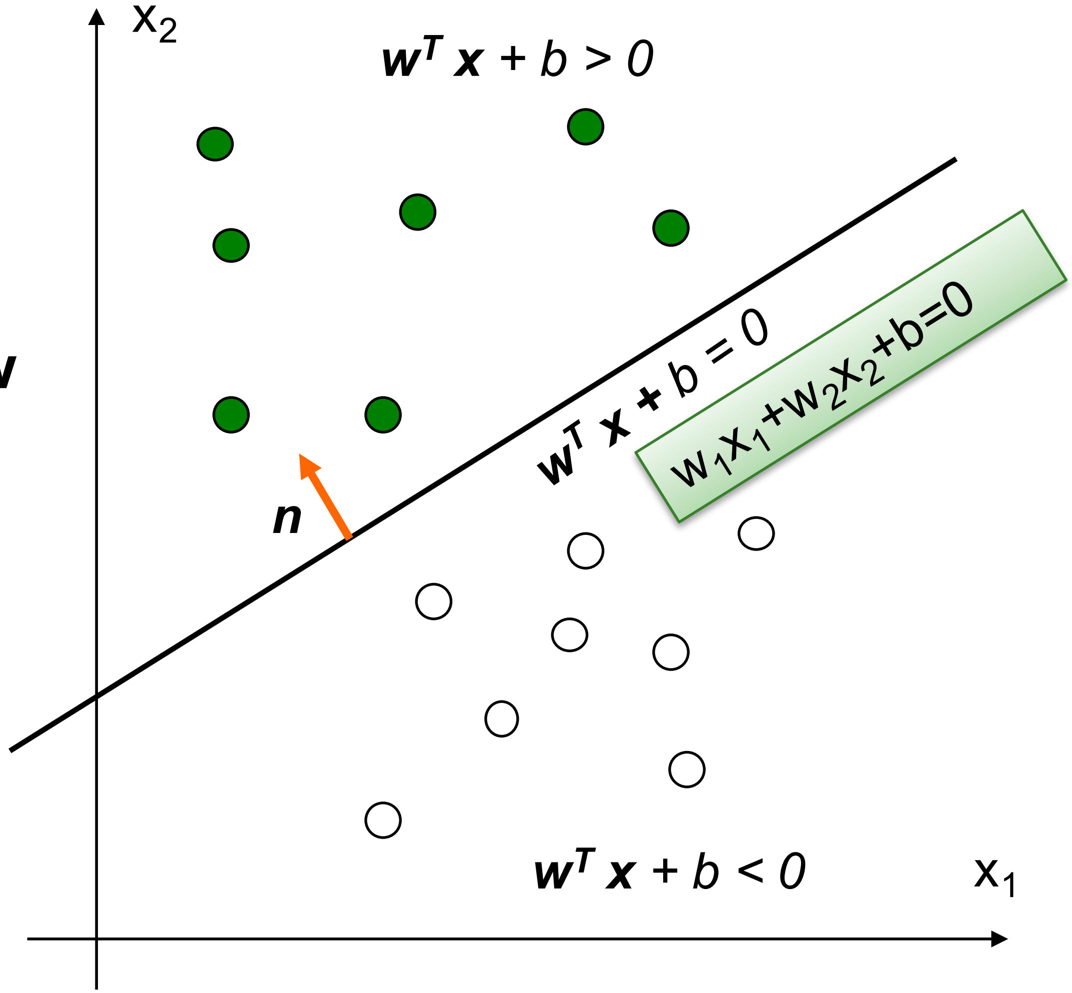
# Linear Function (Linear separator)

- f(x) is a linear function in $R^n$:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

- <u>Written in matrix/vector notation (T → traspose)</u>
- f(x) is a hyper-plane in the n-dimensional feature space, **w** and **x** are vectors (the coefficients $w_i$ and variables $x_i$ of the hyper-plane)
- (Unit-length) normal vector of the hyper-plane:

$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

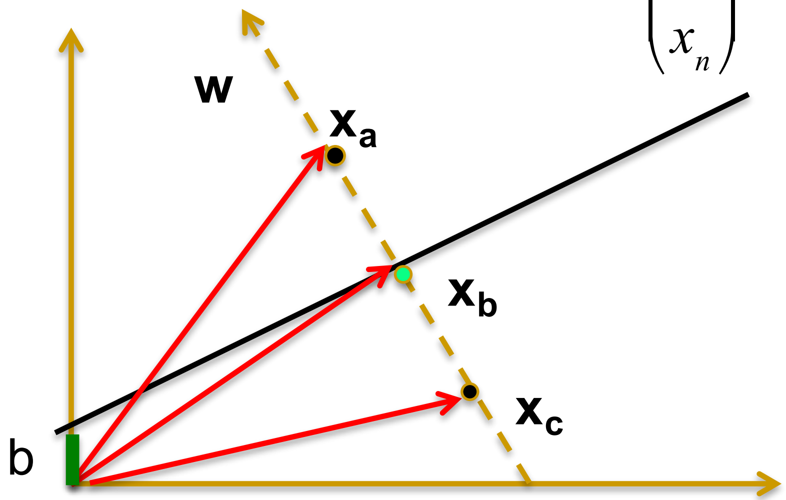



If you divide a vector by its norm, you obtain a unit vector (=with norm =1)

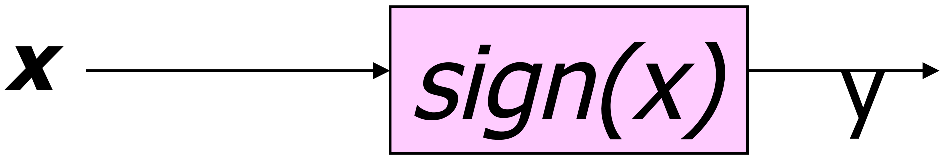
# Vector representation (just to recall..)

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

$$\mathbf{w}^T\mathbf{x} = (w_1, w_2, \ldots w_n)\begin{pmatrix} x_1 \\ x_2 \\ . \\ x_n \end{pmatrix}$$



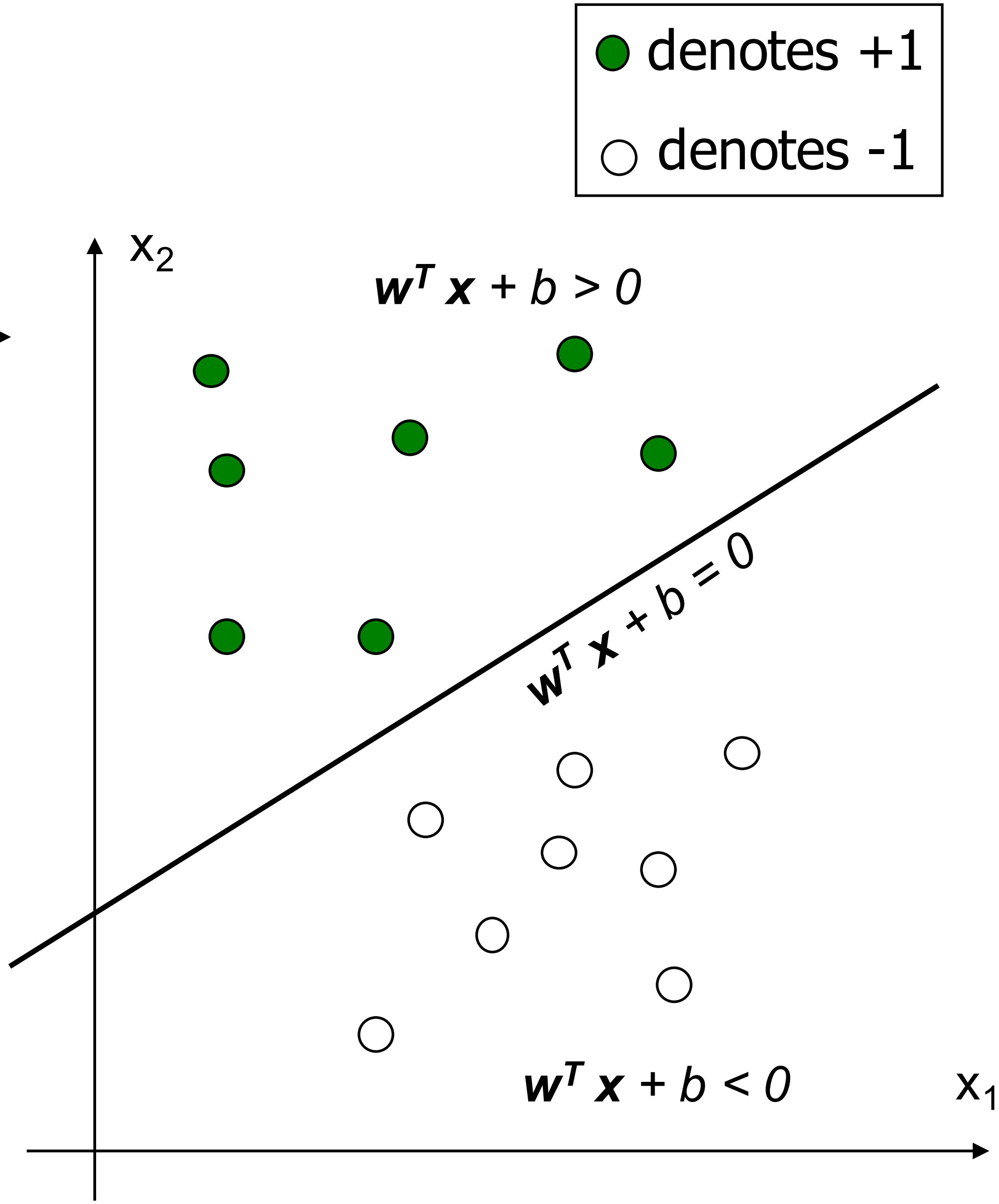
# Linear separator




*The function learnt is:*
*$y=\text{sign}(\mathbf{w}^T \mathbf{x} + b)$*

*If sign()>0 then **x** classified positive*

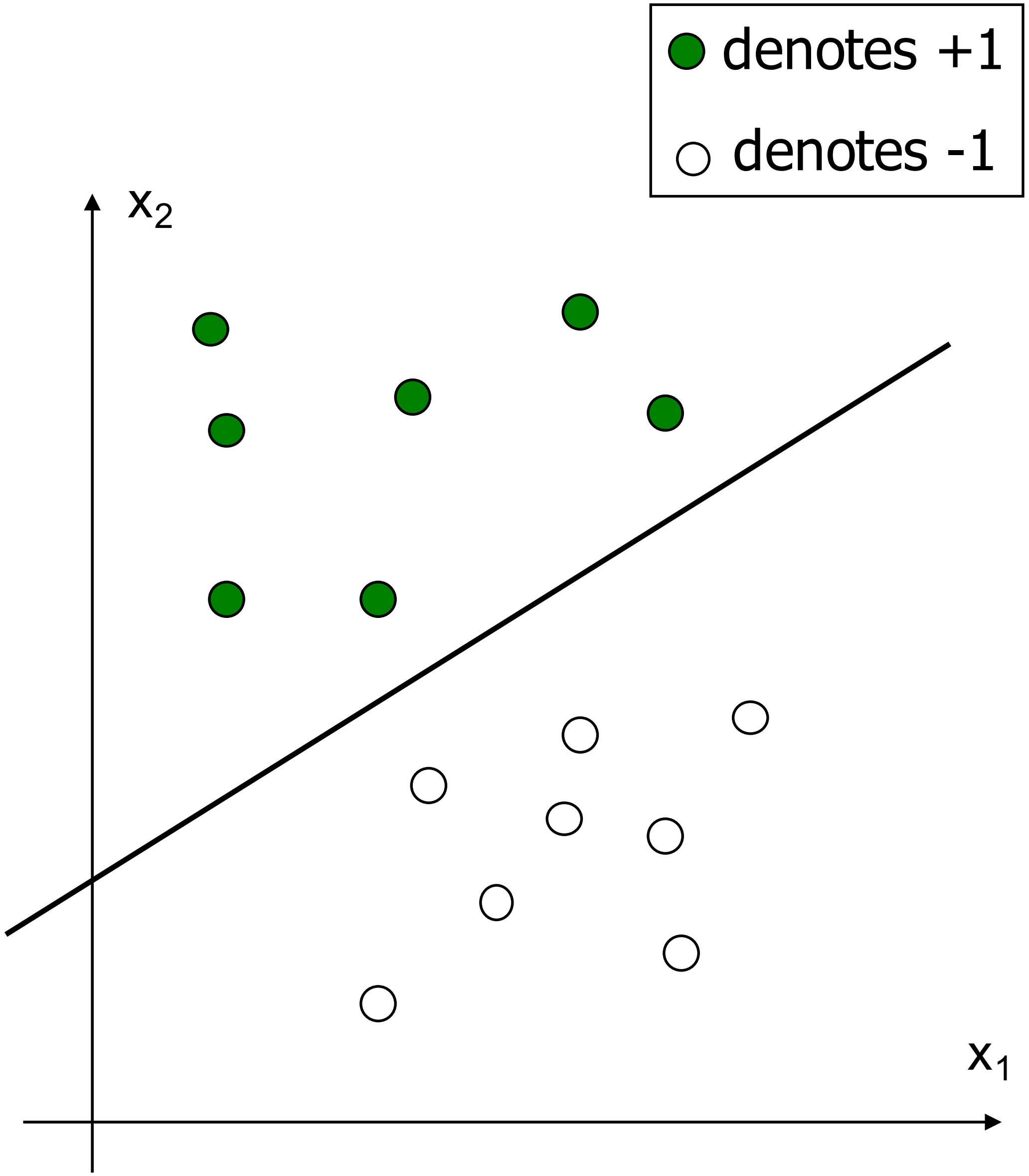
*If sign()<0 then **x** classified negative*



# Linear Function

- How would you classify these points using a linear function?

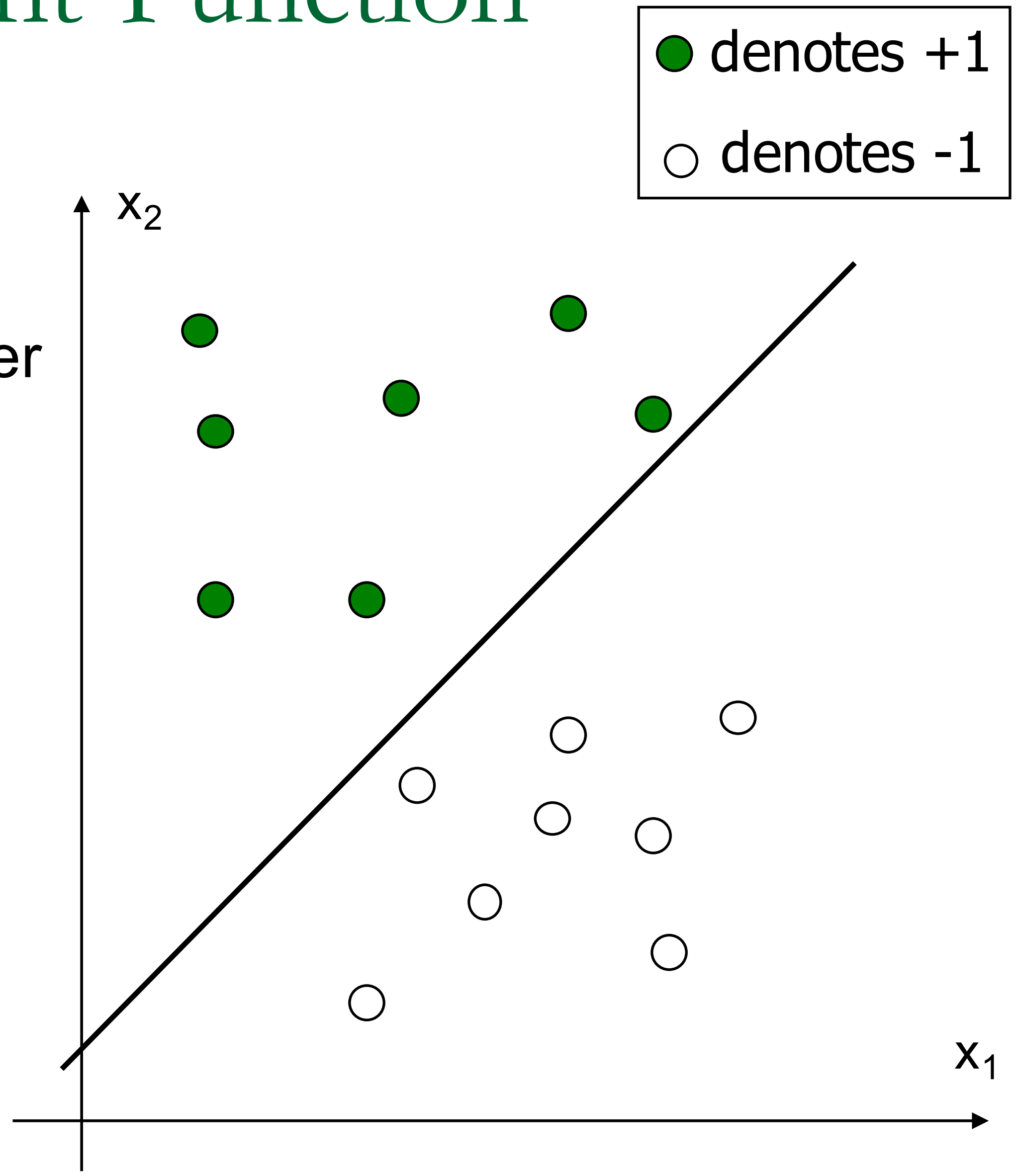
- Infinite number of answers!



# Linear Discriminant Function

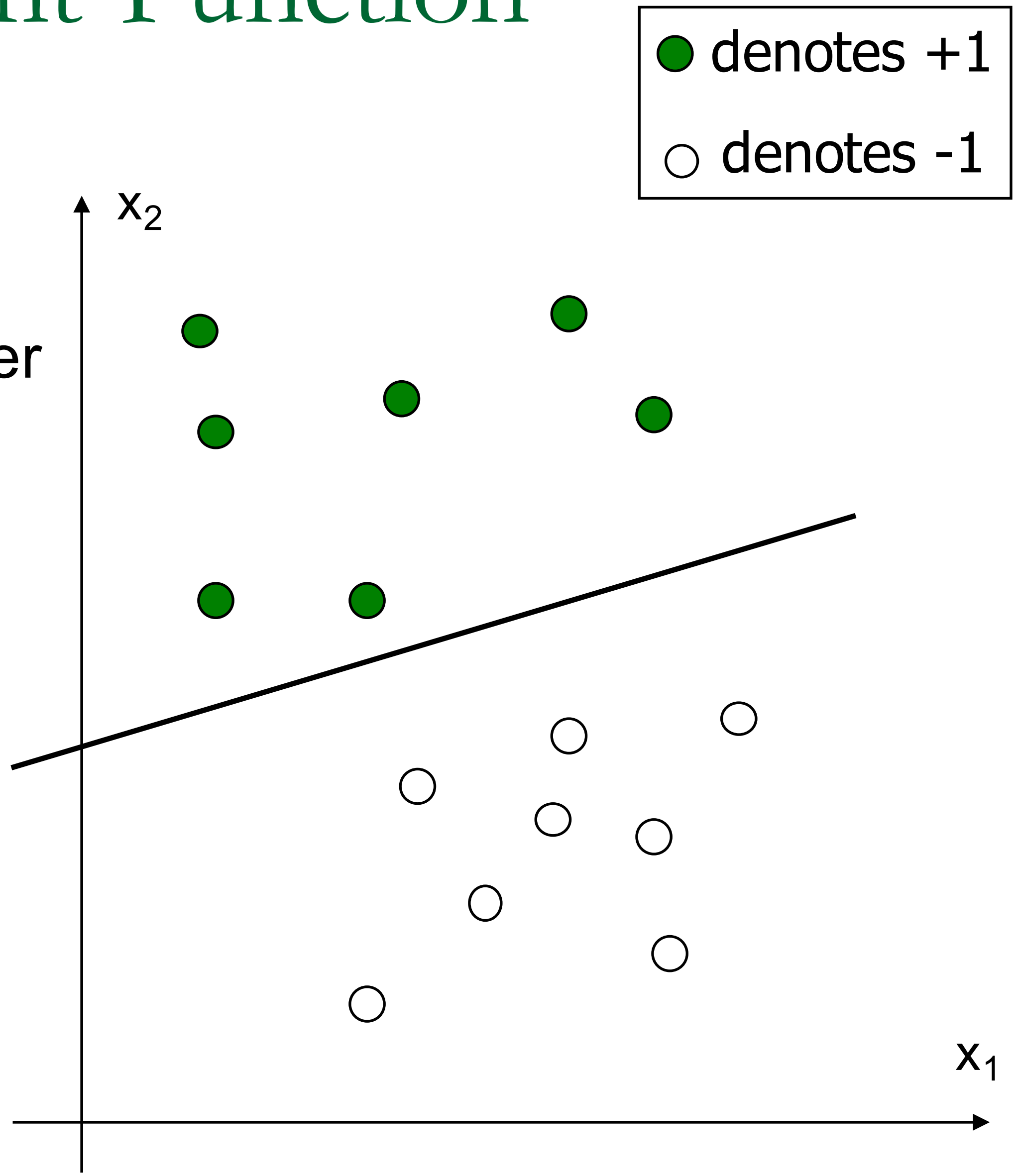
- How would you classify these points using a linear discriminant function in order to minimize the error rate?

- Infinite number of answers!



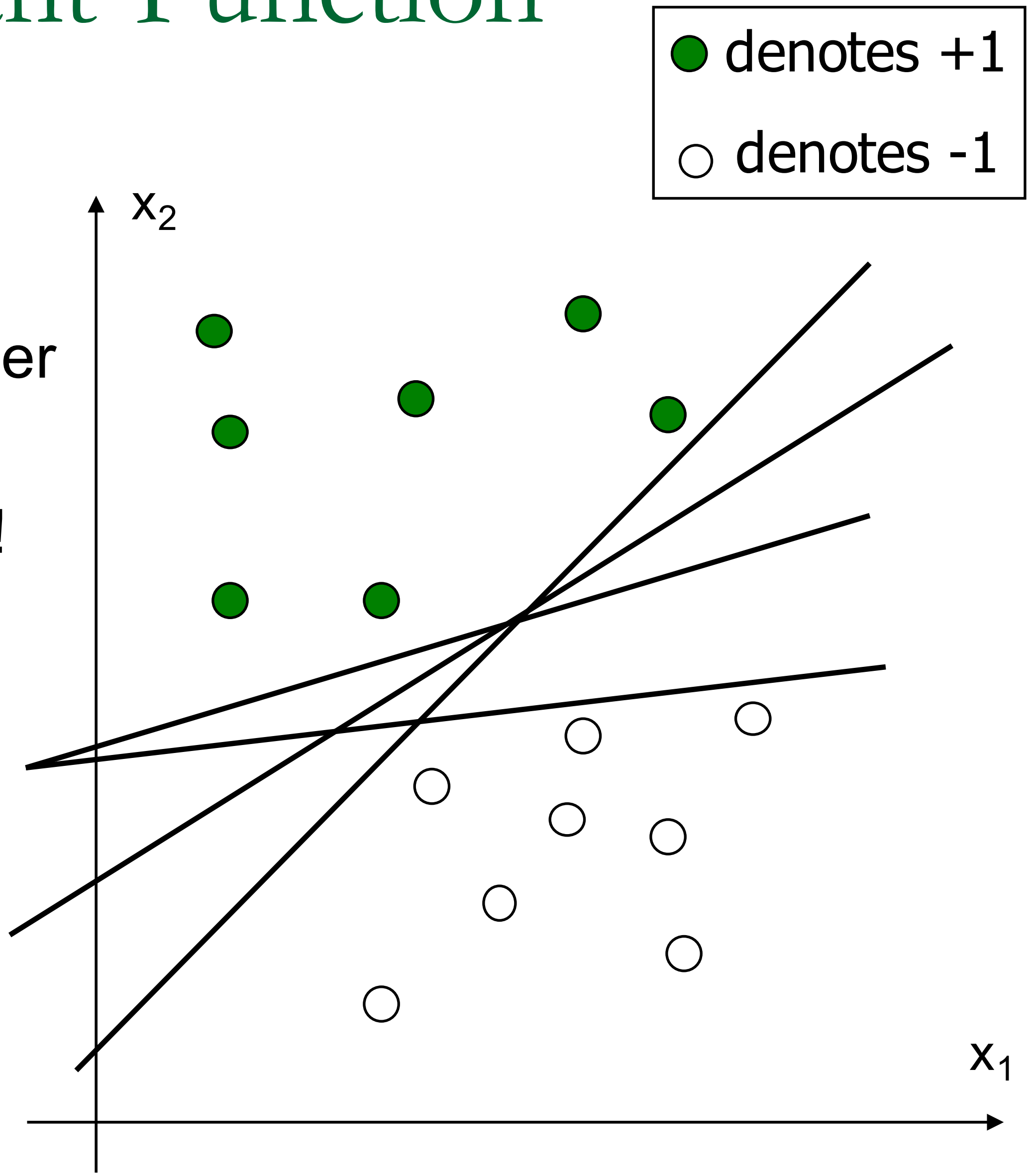
# Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?
- Infinite number of answers!

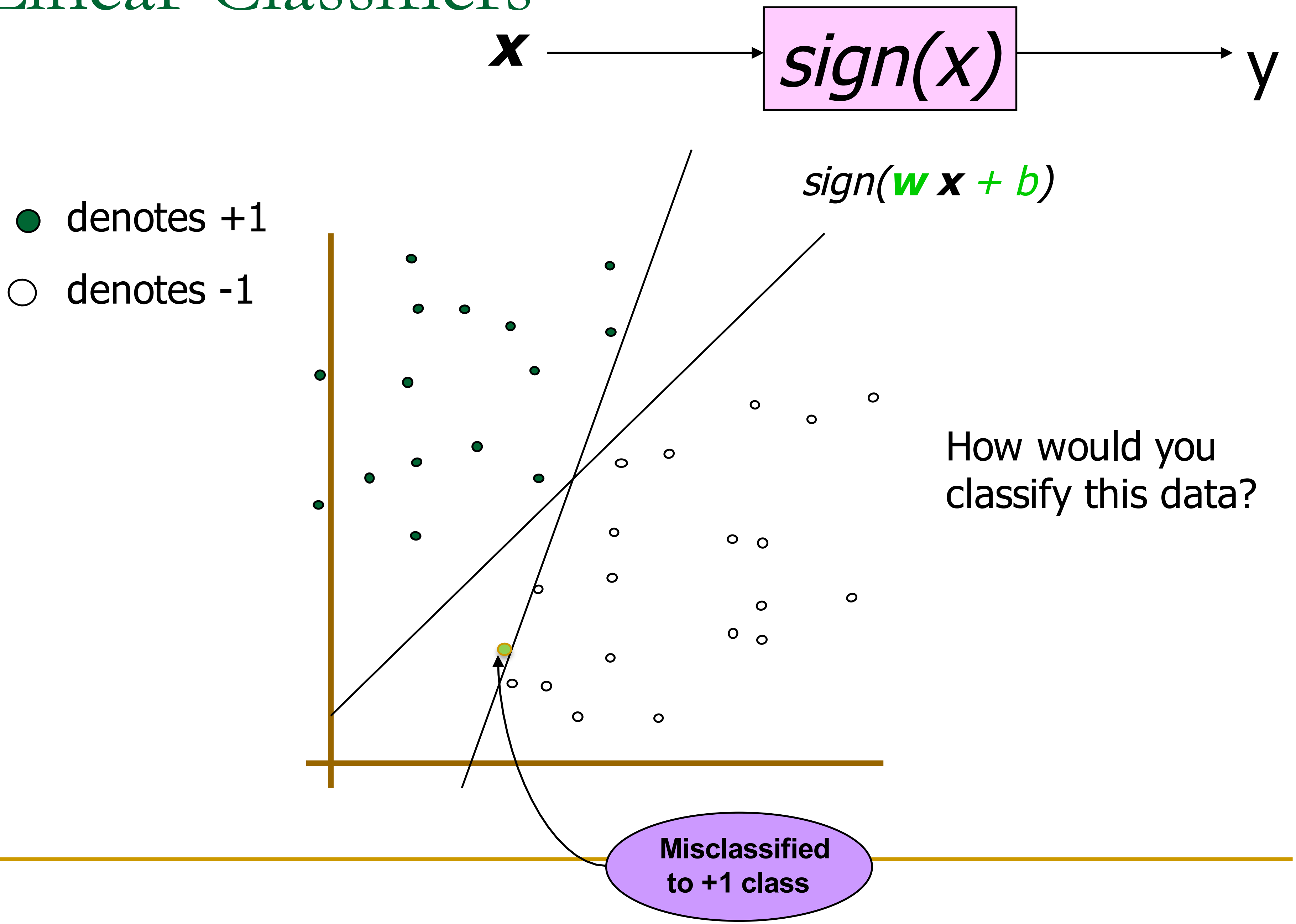


# Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?
- Infinite number of answers!
- **Which one is the best**?



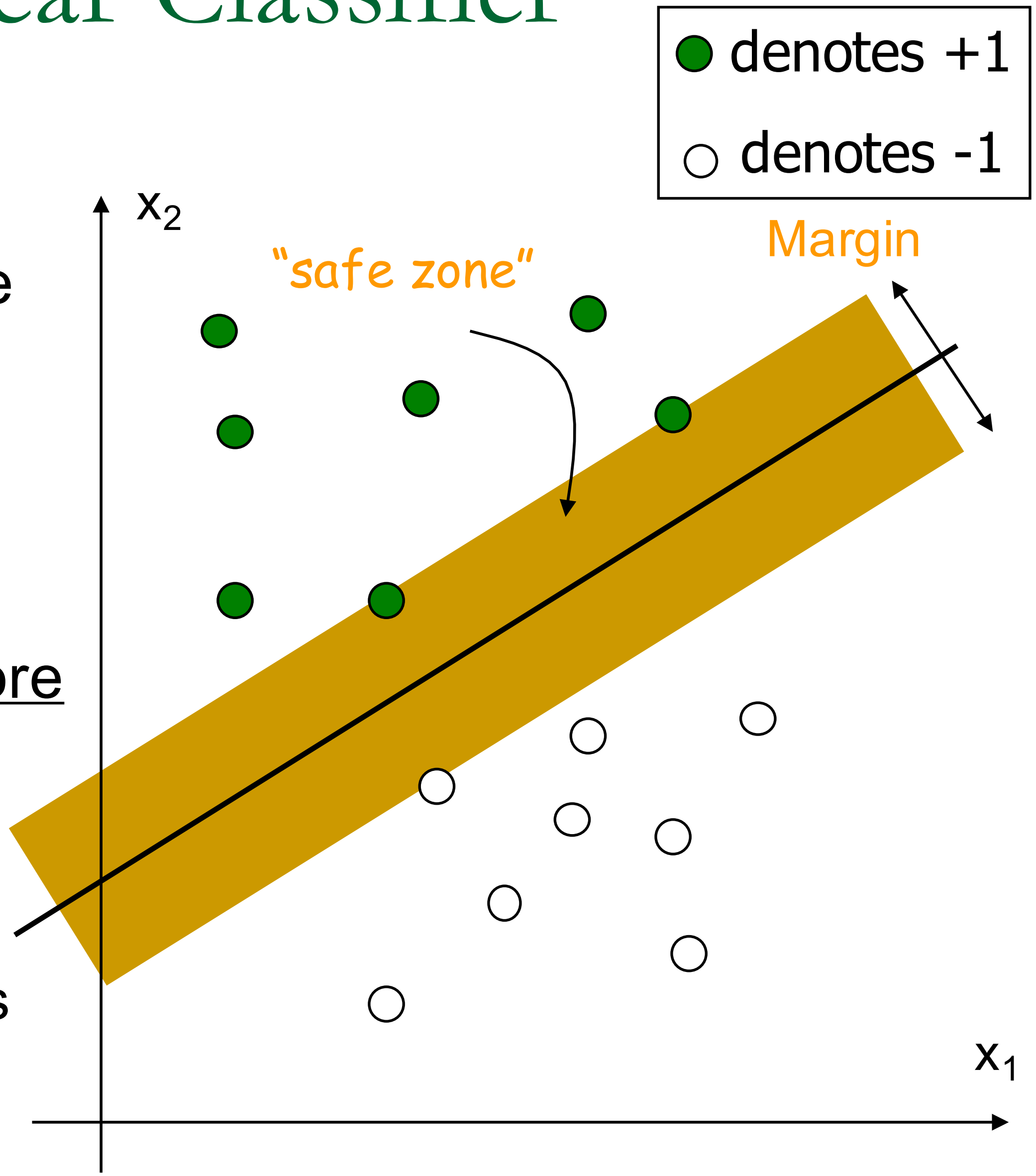
# Linear Classifiers



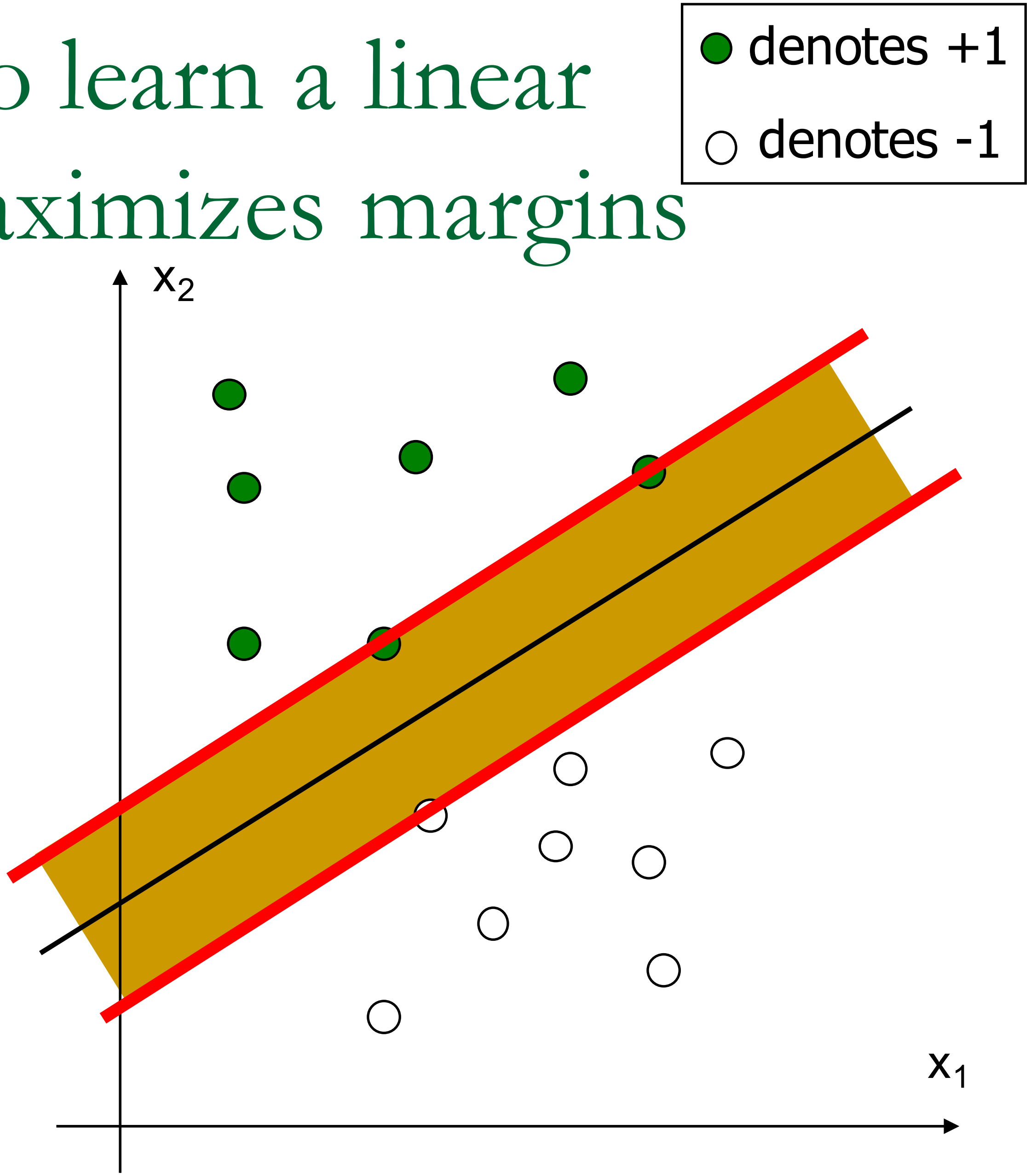

How would you classify this data?

# Large Margin Linear Classifier

- The linear discriminant function (classifier) with the maximum **margin** is the best!
- **Margin** is defined as the width that the boundary could be increased by before hitting a data point in the learning set
- Why it is the best?
  - Robust to outliers and thus stronger generalization ability



# So our target is to learn a linear separator that maximizes margins



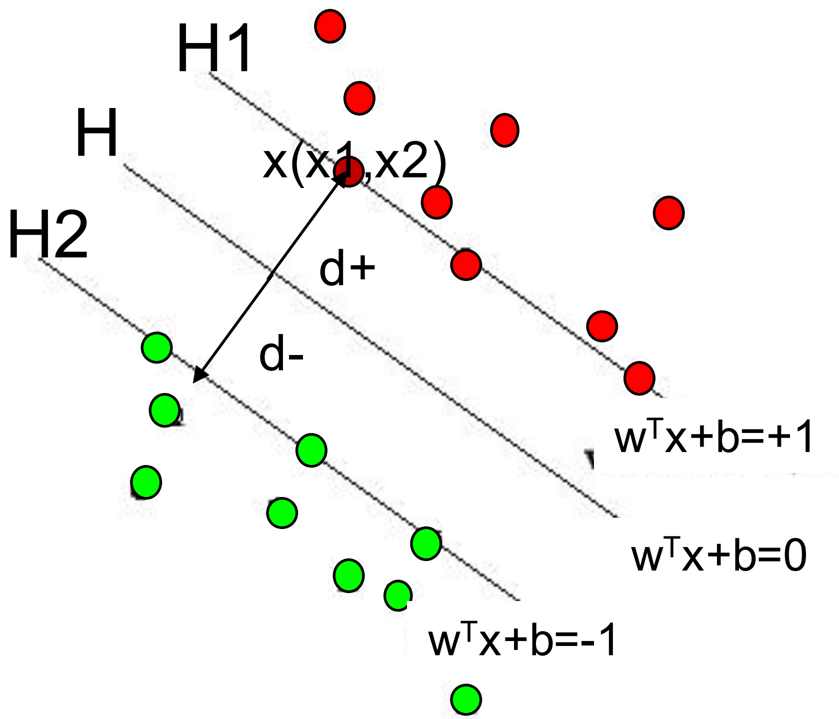

- Target of the ML task:
- Learn **w**, b (where **w** is a vector of coefficients and b is a constant) such that margins are maximized

# Maximizing the margin

We want a classifier with as big margin as possible.

Recall the distance from a point x $(x_1,x_2)$ to a line w1x+w2y+b = 0 is:

$$\frac{|w_1x_1 + w_2x_2 + b|}{\sqrt{w_1^2 + w_2^2}} = \frac{|\boldsymbol{w} \cdot \boldsymbol{x} + b|}{||w||}$$




The distance between H and H1 is then:
|**w•x**+b|/||w||=1/||w||

The distance (margin) between H1 and H2 is then: 2/||w||

**In order to maximize the margin, we need to minimize ||w||. With the condition that there must be <u>no data points between H1 and H2,</u>** e.g., for any $x_i$ in D:

$\left.\begin{array}{l} \mathbf{x}_i \bullet \mathbf{w} + b \geq +1 \text{ when } y_i = +1 \\ \mathbf{x}_i \bullet \mathbf{w} + b \leq -1 \text{ when } y_i = -1 \end{array}\right\}$ **Can be combined into $y_i(\mathbf{x}_i \bullet \mathbf{w}) \geq 1$**

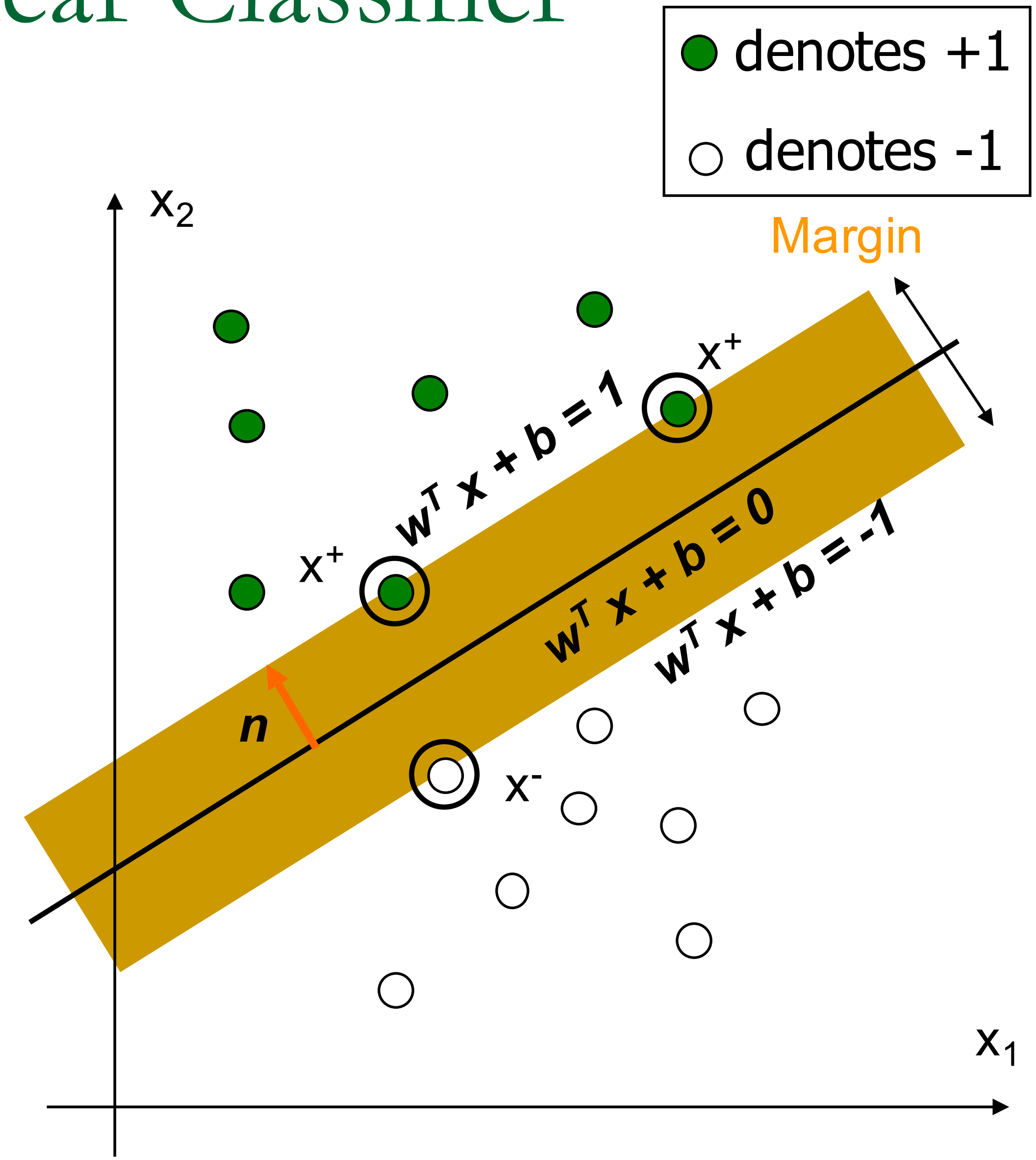
# Large Margin Linear Classifier

- Equivalent formulation:

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

such that

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$



# Solving the Optimization Problem

**Quadratic programming with linear constraints**

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$

Lagrangian Method:

$$\text{minimize} \quad L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left( y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.} \quad \alpha_i \geq 0$$

# Lagrangian

Given a function f(x) and a set of constraints $c_1..c_n$, a *Lagrangian* is a function L(f,c1,..cn, $\alpha_1$,.. $\alpha_n$) that "incorporates" constraints in the optimization problem

$$L(x,\alpha) = f(x) - \sum \alpha_i c_i(x)$$

The optimum is at a point where (Karush-Kuhn-Tucker (KKT) conditions):

$$1) \nabla f(x) - \sum \alpha_i \nabla c_i(x) = 0$$

$$2) \alpha_i \geq 0$$

$$3) \alpha_i c_i(x) = 0 \ \ \forall i$$

Derivative is zero

The third condition is known as **complementarity condition**

# Solving the Optimization Problem

$$\text{minimize} \quad L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \boxed{\sum_{i=1}^{n} \alpha_i \left( y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)}$$

Note: our variables here are **w**, b and the $\alpha_i$

s.t. $\alpha_i \geq 0$ $\quad c_i(x) = (y_i(w^T x_i + b) - 1)$

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \frac{2}{2}\mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \ \rightarrow \ \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L_p}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$2)\alpha_i \geq 0$$

$$3)\alpha_i(y_i(w^T x_i + b) - 1)) = 0 \ \ \forall i$$

To minimize *L* we need to maximize the red box

# Dual Problem Formulation (2)

$$\text{minimize} \quad L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left( y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.} \quad \alpha_i \geq 0$$

Since

$\| w \| = w^T w$

and

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{s.t.} \quad \alpha_i \geq 0 \text{ , and } \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

# All the steps

$$\text{minimize } L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left( y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

$$||w||^2 = w^T w$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\frac{1}{2}\sum\sum \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j - \sum\sum \alpha_i y_i \alpha_j y_j \boldsymbol{x}_i^T \boldsymbol{x}_j - b \cancel{\sum \alpha_i y_i} + \sum \alpha_i$$

$$= \boxed{\sum \alpha_i - \frac{1}{2}\sum\sum \alpha_i y_i \alpha_j y_j \boldsymbol{x}_i^T \boldsymbol{x}_j}$$

s.t. $\alpha_i \geq 0$

Each non-zero $\alpha_i$ indicates that corresponding $\mathbf{x_i}$ is a support vector SV.

# Why only SV have α>0?

$$\begin{aligned} \alpha_i &\geq 0 \\ y_i(w^\top x_i + b) - 1 &\geq 0 \\ \boxed{\alpha_i(y_i(w^\top x_i + b) - 1) = 0.} \end{aligned}$$

The *complementarity condition* (the **third one of KKT**, see previous definition of Lagrangians) implies that one of the 2 multiplicands is zero. However, $y_i(w^T x_i + b) > 1$ for non-SV **x**i.
Therefore **only data points on the margin (=SV)** will have a non-zero α (because they are the only ones for wich **y**i (**w**$^{T}$**x**i + **b** ) = 1 )

# Final Formulation

- To summarize:

Find $\alpha_1 \ldots \alpha_n$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all $\alpha_i$

- Again, remember that the constraint (2) can be verified with a non-equality to zero only for the SV!!
- $\mathbf{Q}(\boldsymbol{\alpha})$ can be computed since we know the $y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ *(they are the pairs* $\langle \mathbf{x}_i, y_i \rangle$ *of the training set D!!)*
- *So, in this final formulation the only variables to be computed are the* $\alpha_i$ *of the support vectors. Wrt the original formulation, b and* $w_i$ *have disappeared!*

# The Optimization Problem Solution

- Given a solution to the dual problem **Q(*α*)** (i.e. computing the $\alpha_1 \ldots \alpha_n$ ), **solution to the primal** (i.e. computing **w** and b) is:

$$\boxed{\mathbf{w} = \Sigma \alpha_i y_i \mathbf{x}_i \qquad b = y_k - \Sigma \alpha_i y_i \mathbf{x}_i^{\mathbf{T}} \mathbf{x}_k \quad \text{for any } \alpha_k > 0}$$

- Each non-zero $\alpha_i$ indicates that corresponding $\mathbf{x}_i$ is a support vector.
- Then the classifying function for a **new** point x is (note that we don't need to compute **w** explicitly):

$$\boxed{\Phi(\mathbf{x}) = \Sigma \alpha_i y_i \mathbf{x}_i^{\mathbf{T}} \mathbf{x} + b \quad (xi \text{ are SV})}$$

- Notice to predict the class of **x** we perform an *inner (dot) product* $\mathbf{x}_i^{\mathbf{T}}\mathbf{x}$ between the test point **x** and the support vectors $\mathbf{x}_i$ . **Only SV are useful for classification of new instances!!** (since the other *α values are zero)* **.**
- Also keep in mind that solving the optimization problem involved computing the inner (dot) products $\mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$ **between all training points**.

# Inner or dot product between two vectors (example)

**a**: $(a_1, a_{2,} a_3)$ **b**: $(b_1, b_2, b_3)$

Dot product: $\Sigma a_i b_i$

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3.$$

Example:

Let a=(1,2,3) and b=(4,−5,6)

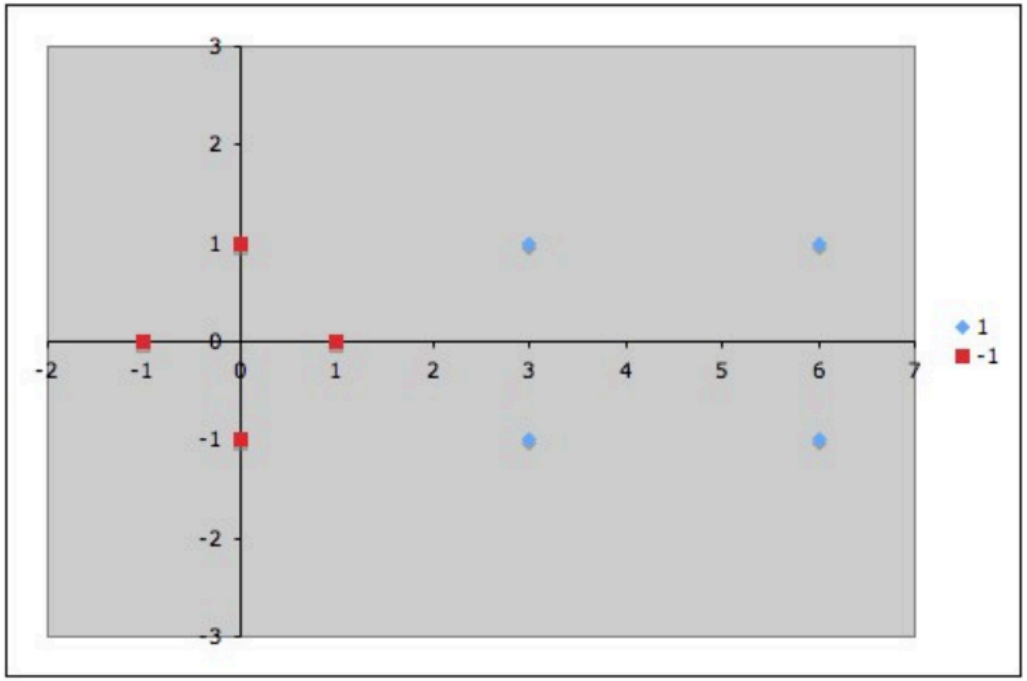
$\mathbf{a} \cdot \mathbf{b}$=1(4)+2(−5)+3(6)=4−10+18=12.

# Example

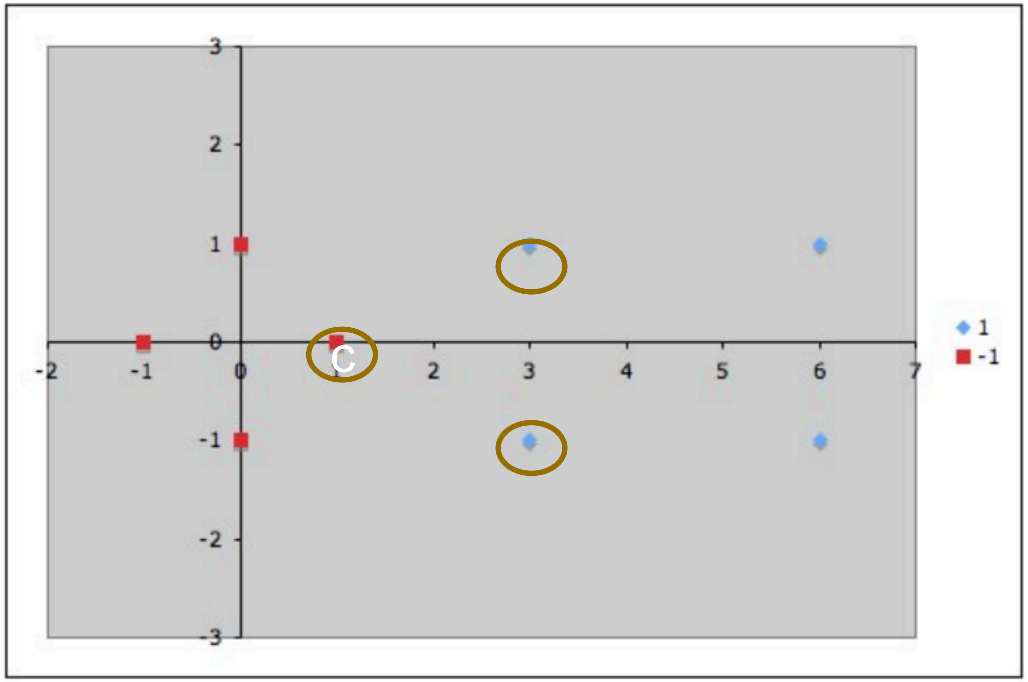
- Suppose we have the dataset:

$$\left\{ \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ -1 \end{pmatrix}, \begin{pmatrix} 6 \\ 1 \end{pmatrix}, \begin{pmatrix} 6 \\ -1 \end{pmatrix} \right\} \quad \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right\}$$

positive negative



# By simple inspection, we can identify 3 SVs




$$\left\{ s_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, s_2 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}, s_3 = \begin{pmatrix} 3 \\ -1 \end{pmatrix} \right\}$$

# First step is to write system of equations to find the "alfas"

- We know that $\mathbf{w}^T\mathbf{x}+b=\Sigma\alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b = -1$ for negative SVs ($s_1$) and $\mathbf{w}^T\mathbf{x}+b=\Sigma\alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b = 1$ for positive SVs ($s_2$,$s_3$)
- Furthermore, $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ (first contraint from derivative) and $\boldsymbol{\alpha}_i$ are non-zero only for SVs.
- we can write a system of equations for each of the 3

$$
\begin{aligned}
\alpha_1 k(s_1, s_1) + \alpha_2 k(s_2, s_1) + \alpha_3 k(s_3, s_1) &= -1 \\
\alpha_1 k(s_1, s_2) + \alpha_2 k(s_2, s_2) + \alpha_3 k(s_3, s_2) &= 1 \\
\alpha_1 k(s_1, s_3) + \alpha_2 k(s_2, s_3) + \alpha_3 k(s_3, s_3) &= 1
\end{aligned}
$$

**Where k(x,y) is the dot product $x^Ty$**

# Then we compute the dot products

$$k(s_1, s_1) = (1\ 0)\begin{pmatrix}1\\0\end{pmatrix} = 1$$
$$k(s_1, s_2) = (1\ 0)\begin{pmatrix}3\\1\end{pmatrix} = 3$$
$$k(s_2, s_3) = (3\ 1)\begin{pmatrix}3\\-1\end{pmatrix} = 9 - 1 = 8$$

Etc. (do it yourself)

# Finally, we obtain the system of equations

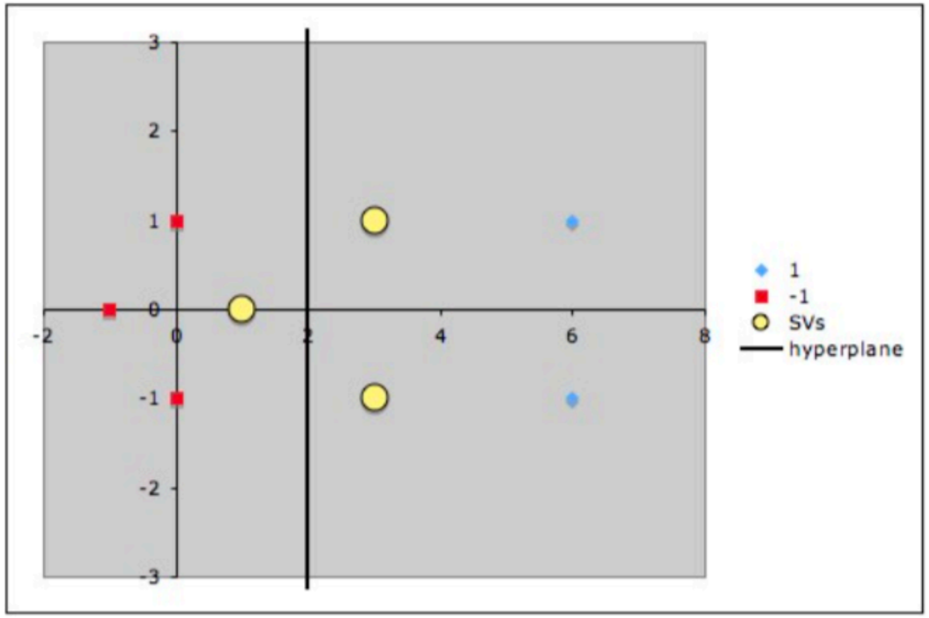
$$\alpha_1 + 3\alpha_2 + 3\alpha_3 = -1$$
$$3\alpha_1 + 10\alpha_2 + 8\alpha_3 = 1$$
$$3\alpha_1 + 8\alpha_2 + 10\alpha_3 = 1$$

From the optimization conditions, we also know that: $\sum_i \alpha_i y_i = 0$

$$-\alpha_1 + \alpha_2 + \alpha_3 = 0$$

$$\alpha_2 = \alpha_3 = \frac{1}{8};\ \alpha_1 = \frac{1}{4};\ b = -2$$

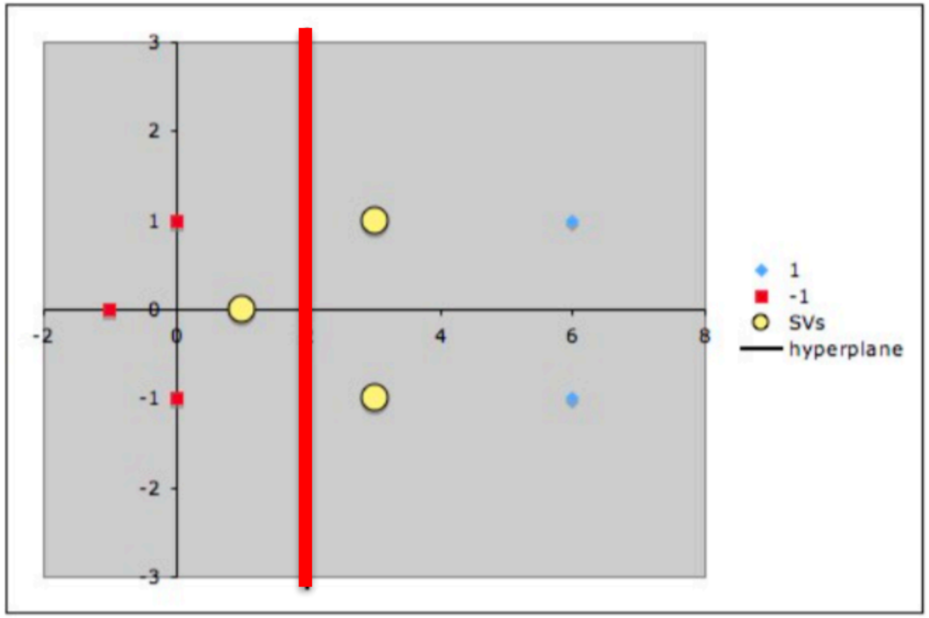
# The solution (graphically)




The $w_i$ are computed using $\mathbf{w} = \Sigma \alpha_i y_i \mathbf{x}_i$

$$w_1 = 1, w_2 = 0, b = -2$$

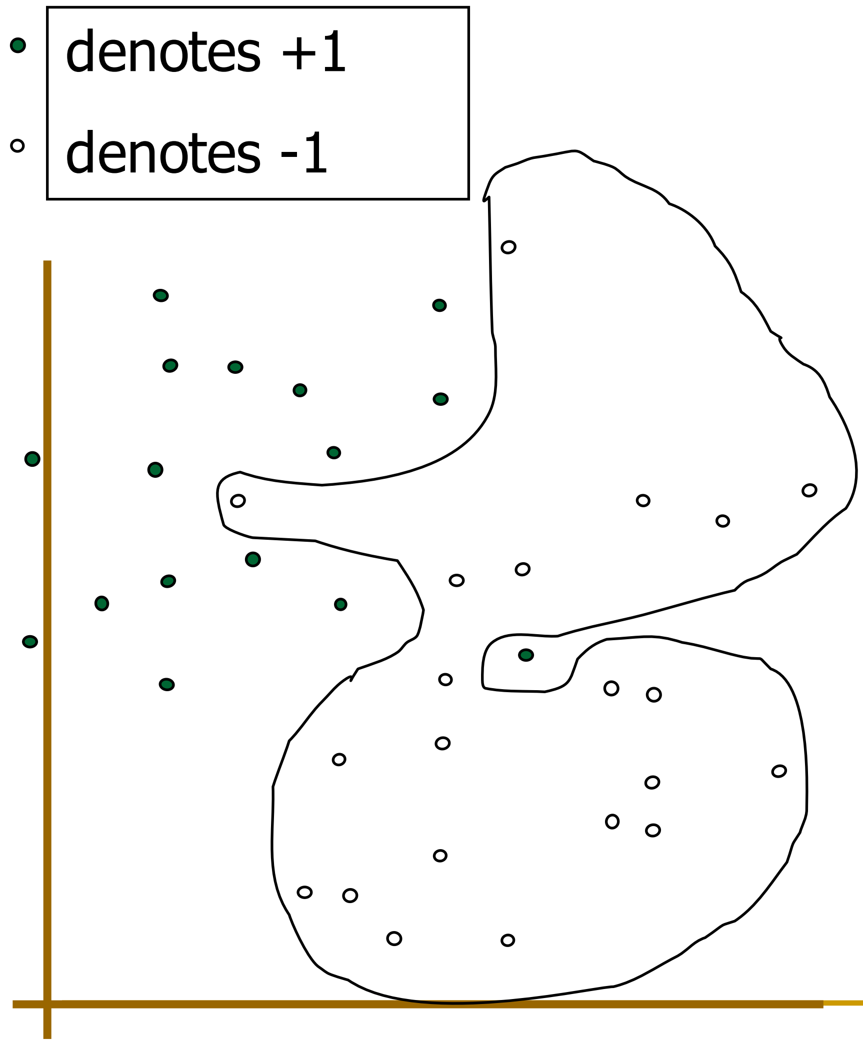
# The solution (graphically)




The $w_i$ are computed using $\mathbf{w} = \Sigma \alpha_i y_i \mathbf{x}_i$

$$w_1 = 1, w_2 = 0, b = -2$$

# Dataset with noise




- **Hard Margin: So far we require all data points be classified correctly**

  **- No training error**

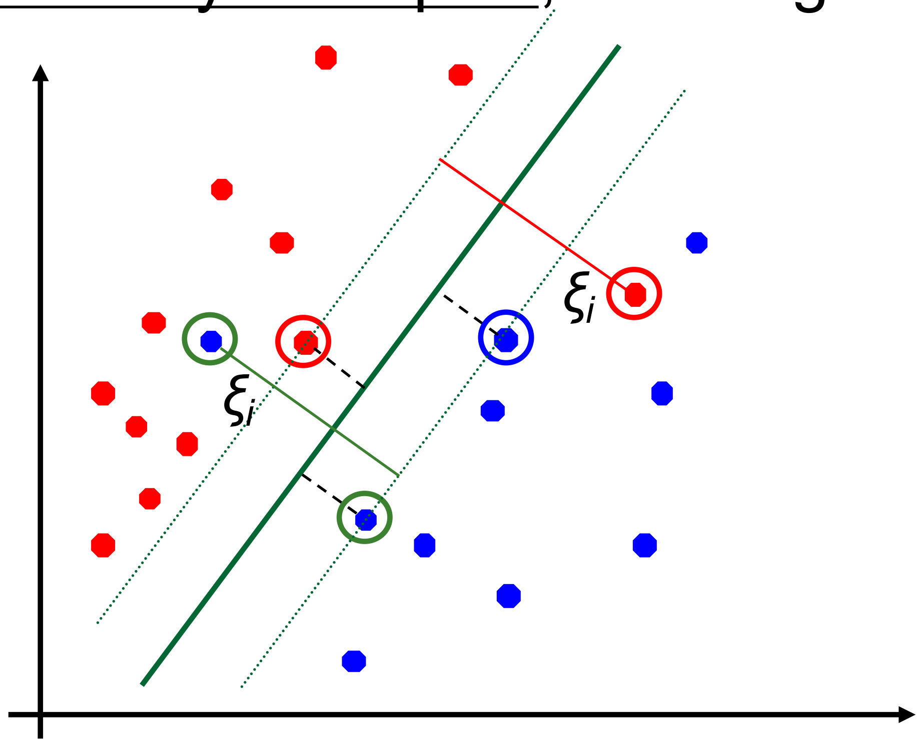
- **What if the training set is noisy?**

  **- Solution 1: use more complex separator**

**OVERFITTING!**

# Soft Margin Classification

- A better solution: slack variables
- *Slack variables* $\xi_i$ can be added to allow misclassification of outliers or noisy examples, resulting margins are called *soft*.




$$\xi_i = \max\left(0, \gamma - y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b)\right)$$

$x_i$ are the misclassified examples

# Large Margin Linear Classifier

- New Formulation:

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

such that

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \boxed{- \xi_i}$$

$$\xi_i \geq 0$$

Slack variables allow an example in the margin $0 \leq \xi_i \leq 1$ to be misclassified, while if $\xi_i > 1$ it is an error

- Parameter $C$ can be viewed as a way to control over-fitting. For large C a large penalty is assigned to errors.

# Hard Margin v.s. Soft Margin

- **The old formulation:**

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

$\mathbf{\Phi}(\mathbf{w}) = ½\, \mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i\, (\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + \mathrm{b}) \geq 1$

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

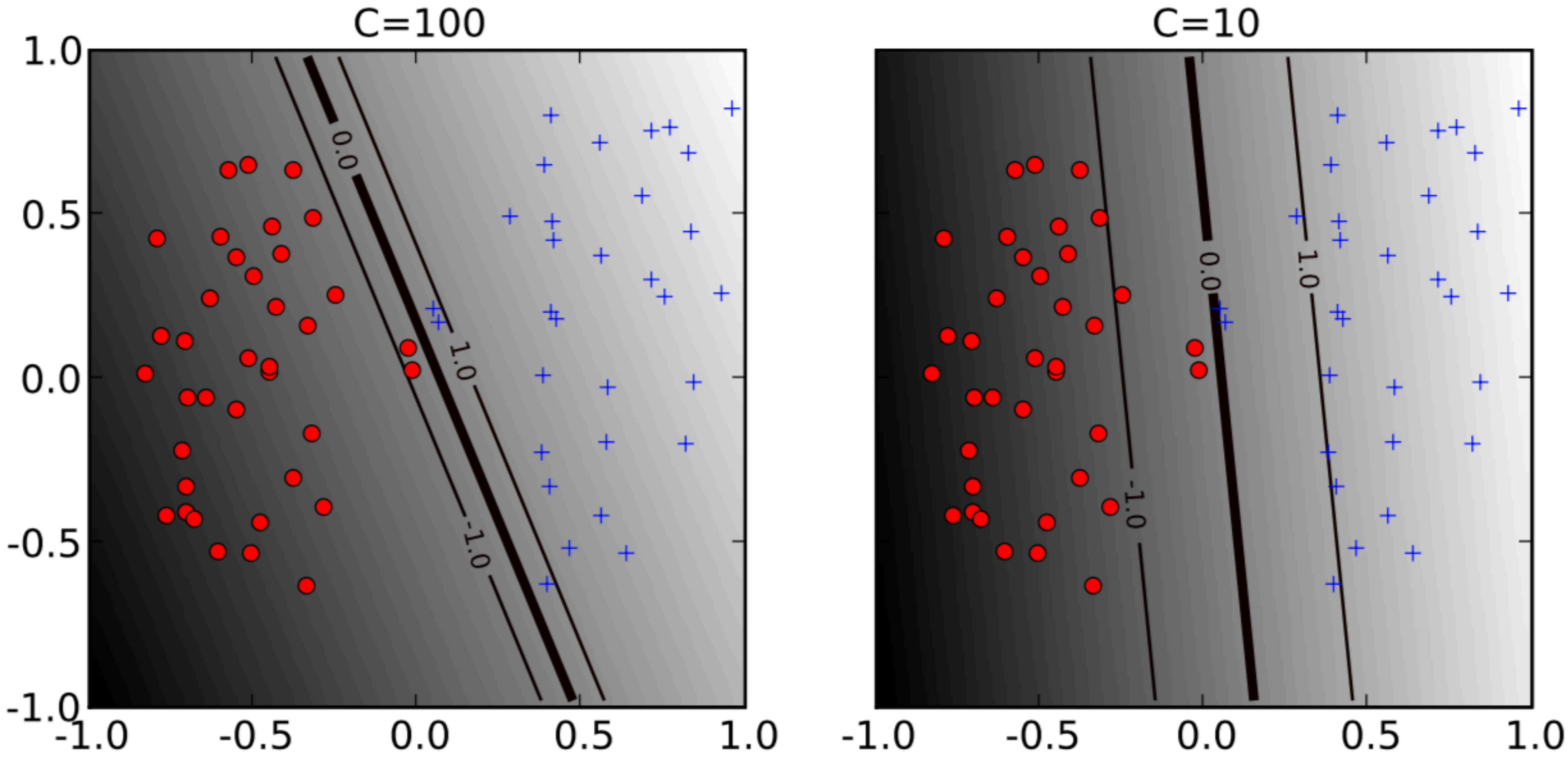
Find $\mathbf{w}$ and $b$ such that

$\mathbf{\Phi}(\mathbf{w}) = ½\, \mathbf{w}^{\mathrm{T}}\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i\, (\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all $i$

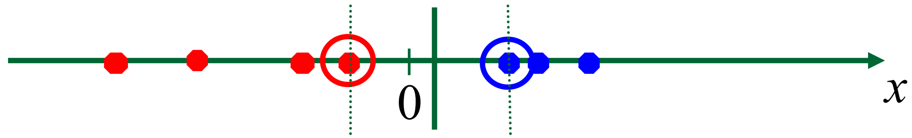
- **Parameter *C* can be viewed as a way to control overfitting.**

# Effect of soft-margin constant C



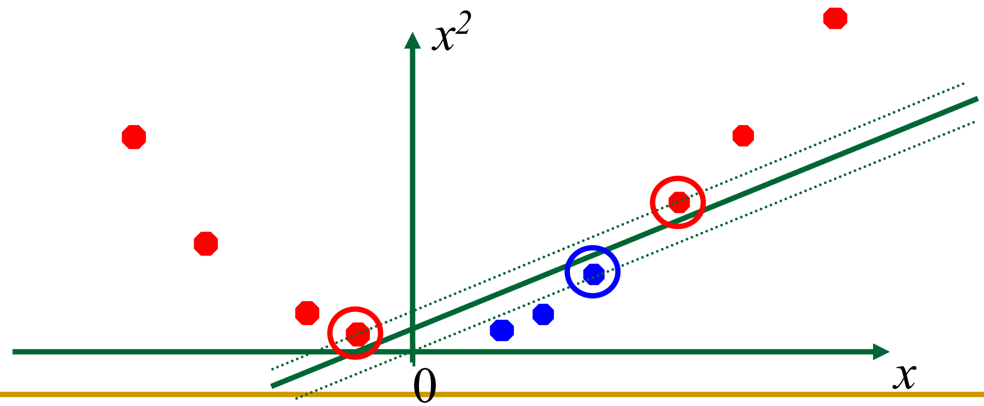
# Non-linear SVMs

- Datasets that are linearly separable (possibly with noise) work out great:




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

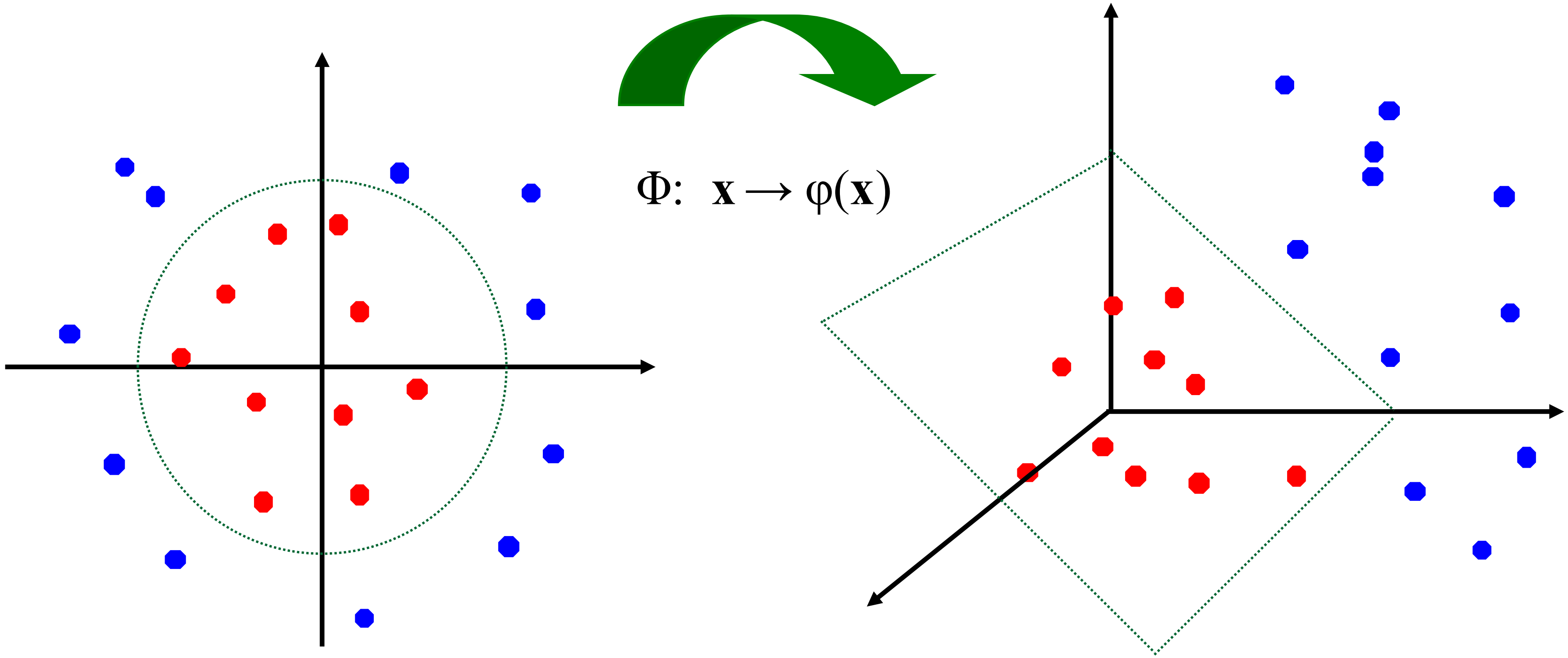

- How about… mapping data to a higher-dimensional space:




This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

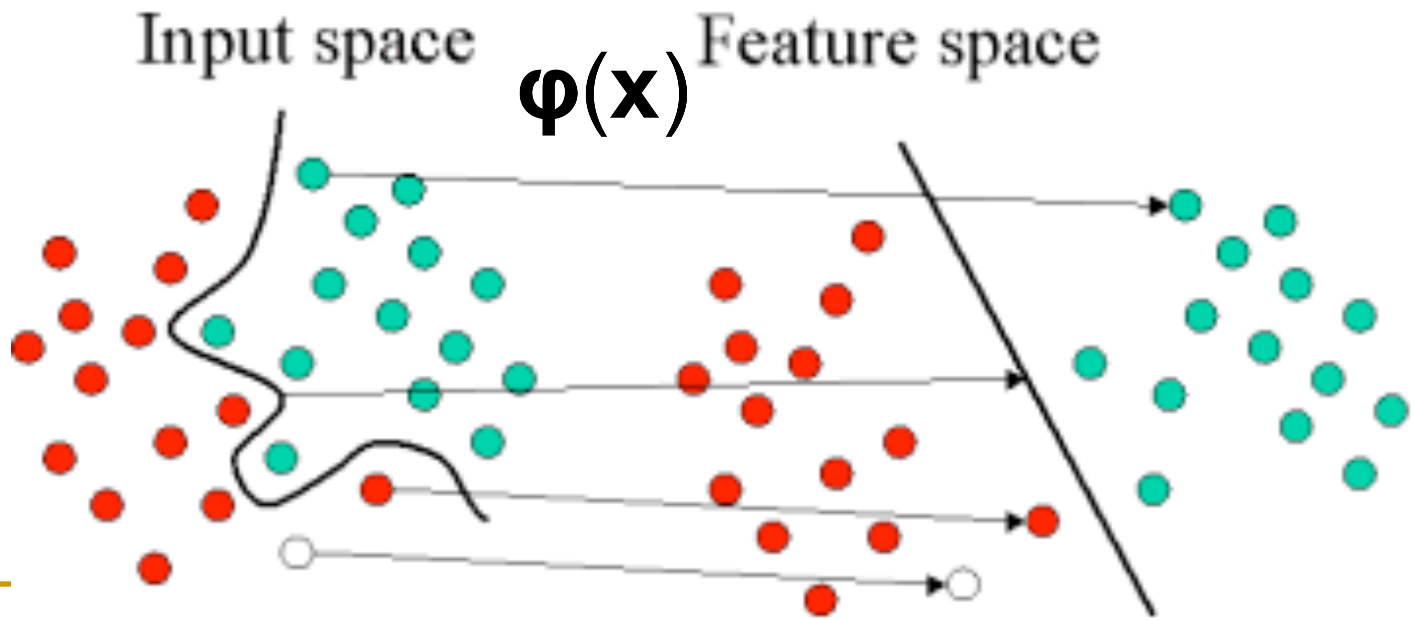
# Non-linear SVMs: Feature Space

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




This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

The original points (left side of the schematic) are mapped, i.e., rearranged, using a set of mathematical functions, known as **kernels**.



SVM with a polynomial Kernel visualization

Created by:
Udi Aharoni

# Nonlinear SVMs: The Kernel Trick

- With this mapping, our discriminant function is now:

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = \sum_{i \in \mathrm{SV}} \alpha_i \boxed{\phi(\mathbf{x}_i)^T \phi(\mathbf{x})} + b$$

*Original formulation was* $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \Sigma \alpha_i y_i \boxed{\mathbf{x}_i^T \mathbf{x}} + b$

- No need to know this mapping function explicitly, because we only use the dot product of feature vectors in both the training and test.
- A *kernel function* is defined as a function that corresponds to a dot product of two feature vectors **in some expanded feature space**: $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

  - Note that, obviously, the dot product $\mathbf{x}_i^T\mathbf{x} = \mathbf{x}_i \bullet \mathbf{x}$ IS a kernel function!

# Nonlinear SVMs: The Kernel Trick

- An 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 $K(\mathrm{x_i},\mathrm{x_j}) = \boldsymbol{\varphi}(\mathbf{x_i})^T\boldsymbol{\varphi}(\mathbf{x_j})$, e.g., that it is a kernel function:

$$
\begin{aligned}
K(\mathbf{x_i},\mathbf{x_j}) &= (1+\mathbf{x_i}^T\mathbf{x_j})^2, \\
&= 1+x_{i1}^2x_{j1}^2+2\,x_{i1}x_{j1}\,x_{i2}x_{j2}+x_{i2}^2x_{j2}^2+2x_{i1}x_{j1}+2x_{i2}x_{j2} \\
&= [1\ \ x_{i1}^2\ \ \sqrt{2}\,x_{i1}x_{i2}\ \ x_{i2}^2\ \ \sqrt{2}x_{i1}\ \ \sqrt{2}x_{i2}]^T[1\ \ x_{j1}^2\ \ \sqrt{2}\,x_{j1}x_{j2}\ \ x_{j2}^2\ \ \sqrt{2}x_{j1}\ \ \sqrt{2}x_{j2}] \\
&= \boldsymbol{\varphi}(\mathbf{x_i})^T\boldsymbol{\varphi}(\mathbf{x_j}), \quad \text{where } \boldsymbol{\varphi}(\mathbf{x}) = [1\ \ x_1^2\ \ \sqrt{2}\,x_1x_2\ \ x_2^2\ \ \sqrt{2}x_1\ \ \sqrt{2}x_2]
\end{aligned}
$$

**Ex: x:(1,2)** $\rightarrow \boldsymbol{\varphi}(\mathbf{x})=(1, 1^2, \sqrt{2}(1\times 2), 2^2, \sqrt{2}\times 1, \sqrt{2}\times 2)=(1,1,2,4,\sqrt{2},2)$

This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

# Nonlinear SVMs: The Kernel Trick

- Examples of commonly-used kernel functions:
  - Linear kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
  - Polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
  - Gaussian (Radial-Basis Function (RBF) ) kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}\right)$$

  - Sigmoid:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^T \mathbf{x}_j + \beta_1)$$

- In general, functions that satisfy *Mercer's condition* can be kernel functions.

# Nonlinear SVM: Optimization

- Formulation: (Lagrangian Dual Problem) find α1, α2,.. $\alpha_n$ such that:

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

such that

$$0 \le \alpha_i \le C$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

- The solution of the discriminant function is

$$g(\mathbf{x}) = \sum_{i \in \mathrm{SV}} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$$

- The optimization technique is the same.

# Support Vector Machine: Algorithm

- 1. Choose a kernel function
- 2. Choose a value for *C*
- 3. Solve the quadratic programming problem (many software packages available)
- 4. Construct the discriminant function from the support vectors

# 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. σ in Gaussian kernel
  - σ 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

This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

# Issues

- Large margin classifiers are known to be sensitive to the way features are scaled. Therefore it is essential to **normalize** the data.
- Also sensible to unbalanced data
- Hyper-parameter tuning (C, kernel): read

A User's Guide to Support Vector Machines

Asa Ben-Hur
Department of Computer Science
Colorado State University

Jason Weston
NEC Labs America
Princeton, NJ 08540 USA

# Summary: Support Vector Machine

- 1. Large Margin Classifier
    - Better generalization ability & less over-fitting
- 2. The Kernel Trick
    - Map data points to higher dimensional space in order to make them linearly separable.
    - Since only dot product is used, we do not need to represent the mapping explicitly.

# Additional Resource

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

# LibSVM (best implementation for SVM)

http://www.csie.ntu.edu.tw/~cjlin/libsvm/