

# Support Vector Machines

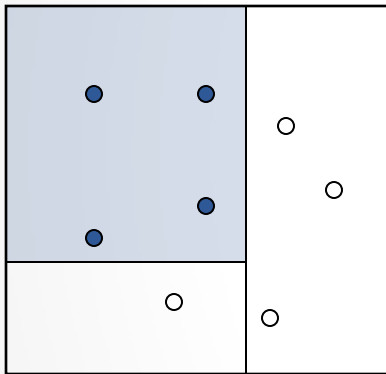
Vibhav Gogate

The University of Texas at dallas

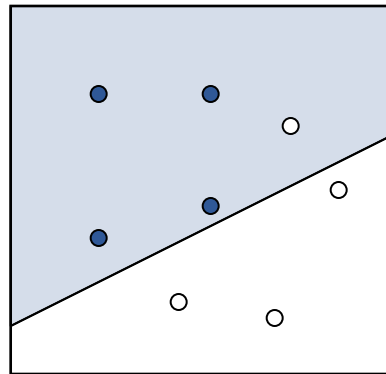
# What We have Learned So Far?

1. Decision Trees
  2. Naïve Bayes
  3. Linear Regression
  4. Logistic Regression
  5. Perceptron
  6. Neural networks
- Which of the above are linear and which are not?
  - (1) (2) and (6) are non-linear

# Decision Surfaces

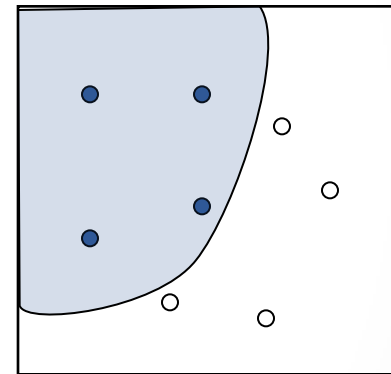


Decision  
Tree



Linear  
Functions

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



Nonlinear  
Functions  
(Neural nets)

# Today: 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-network with hand-designed features in a handwriting recognition task
- Currently, SVM is widely used in object detection & recognition, content-based image retrieval, text recognition, biometrics, speech recognition, etc.
- Also used for regression (will not cover today)
- Chapter 5.1, 5.2, 5.3, 5.11 (5.4\*) in Bishop
- SVM tutorial (start reading from Section 3)



V. Vapnik

# Outline

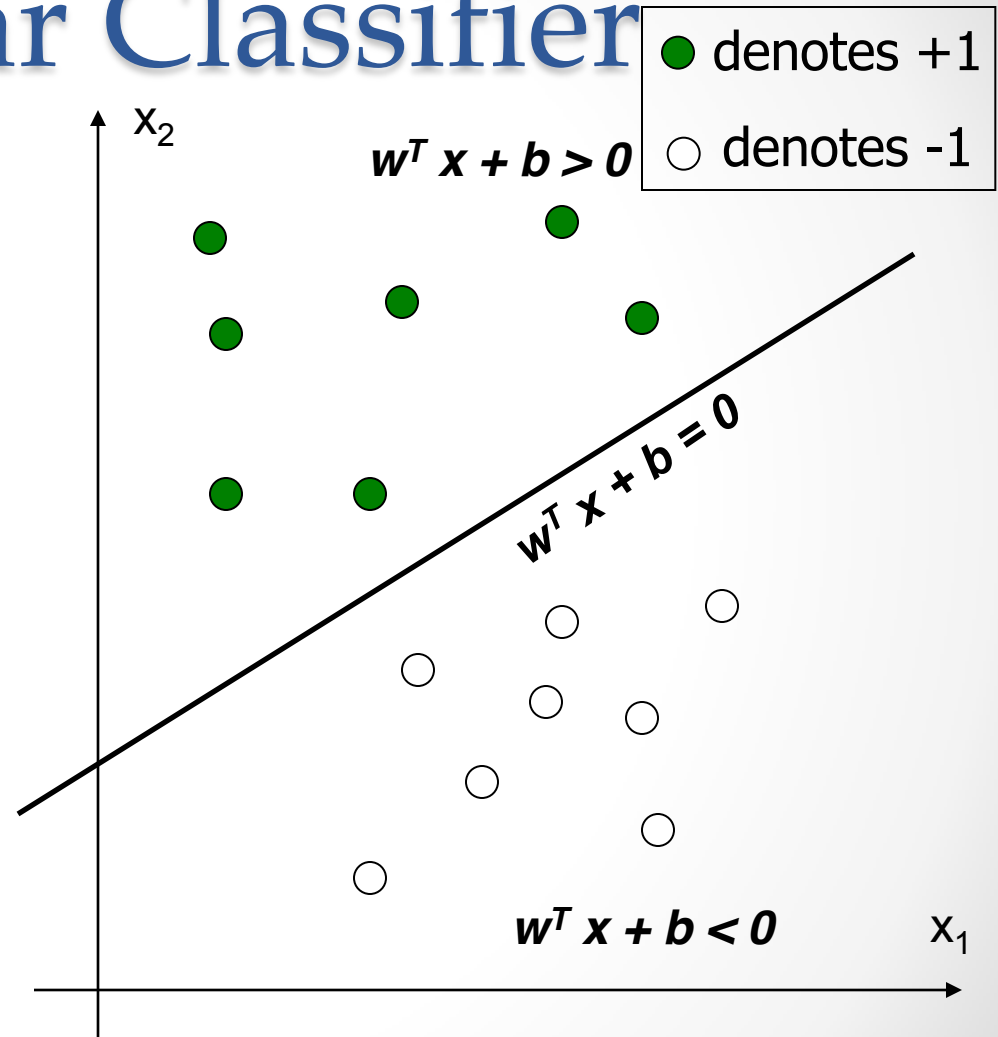
- Linear Discriminant Function
- Large Margin Linear Classifier
- Nonlinear SVM: The Kernel Trick
- Demo of SVM

# Linear Discriminant Function or a Linear Classifier

- Given data and two classes, learn a function of the form:

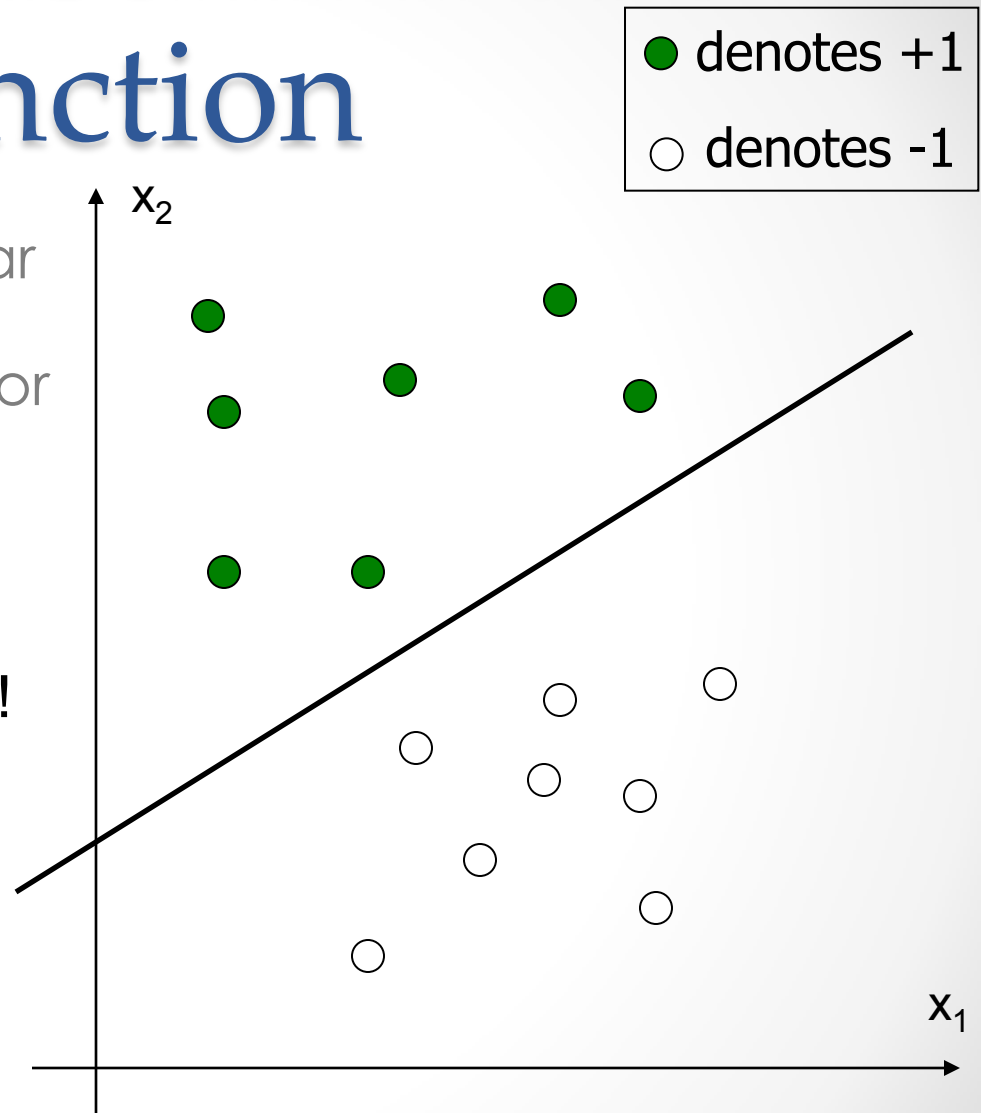
$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

- A hyper-plane in the feature space
- Decide class=1 if  $g(\mathbf{x}) > 0$  and class=-1 otherwise



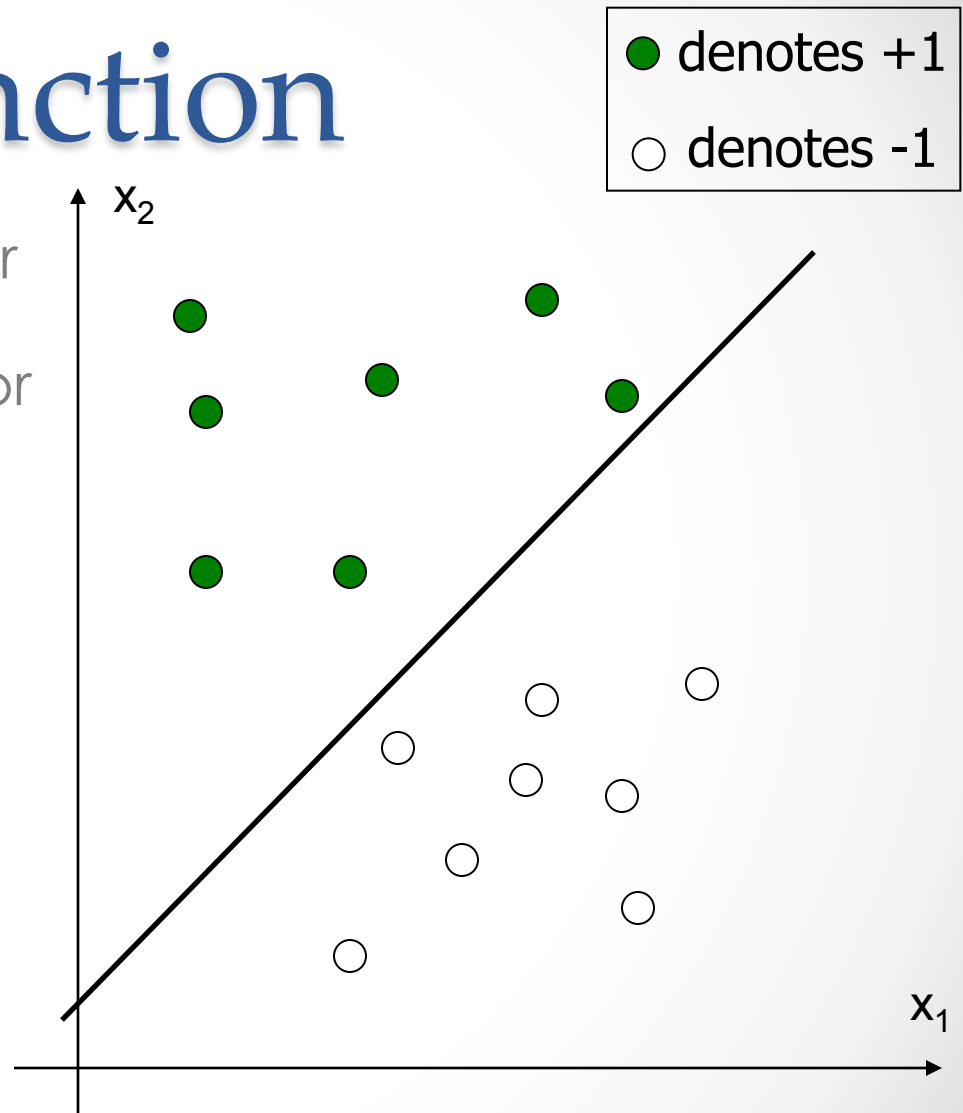
# Linear Discriminant Function

- 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!

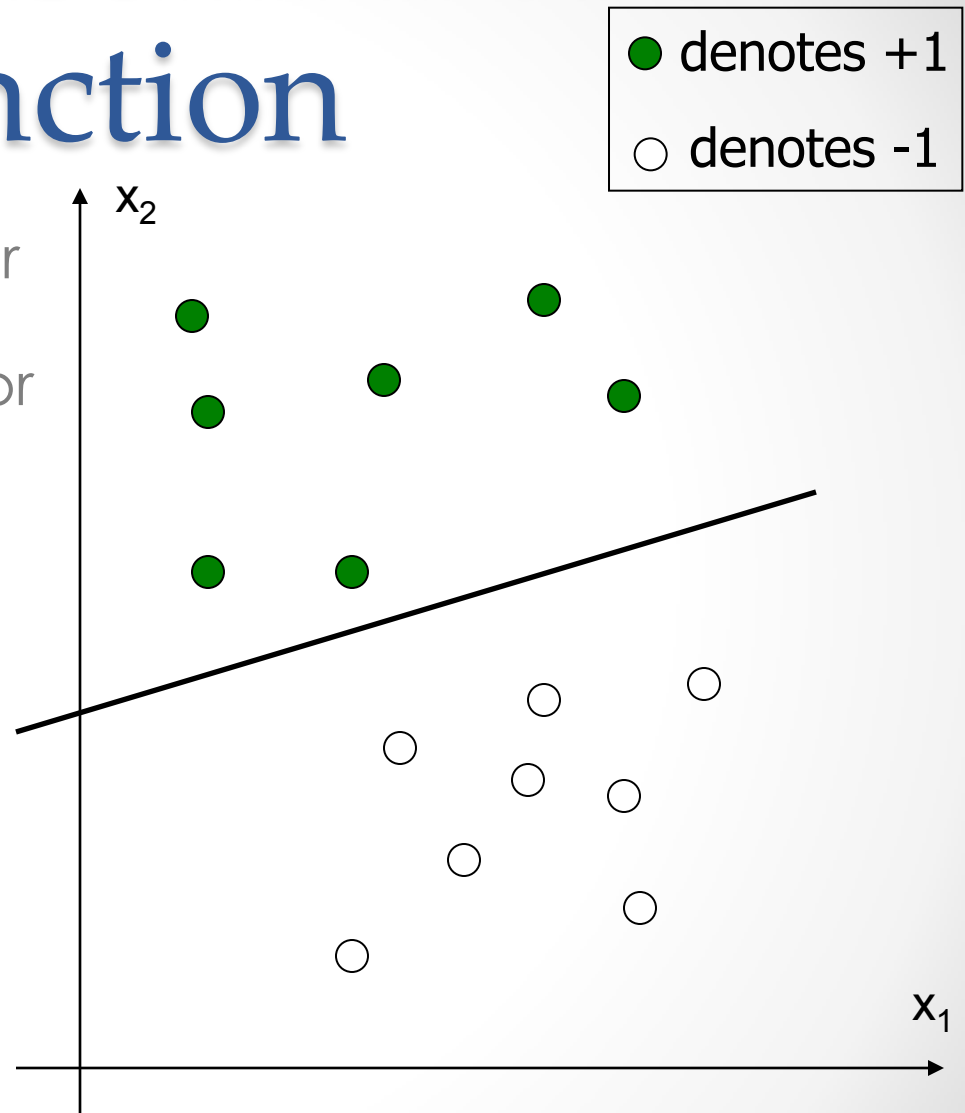




# Linear Discriminant Function

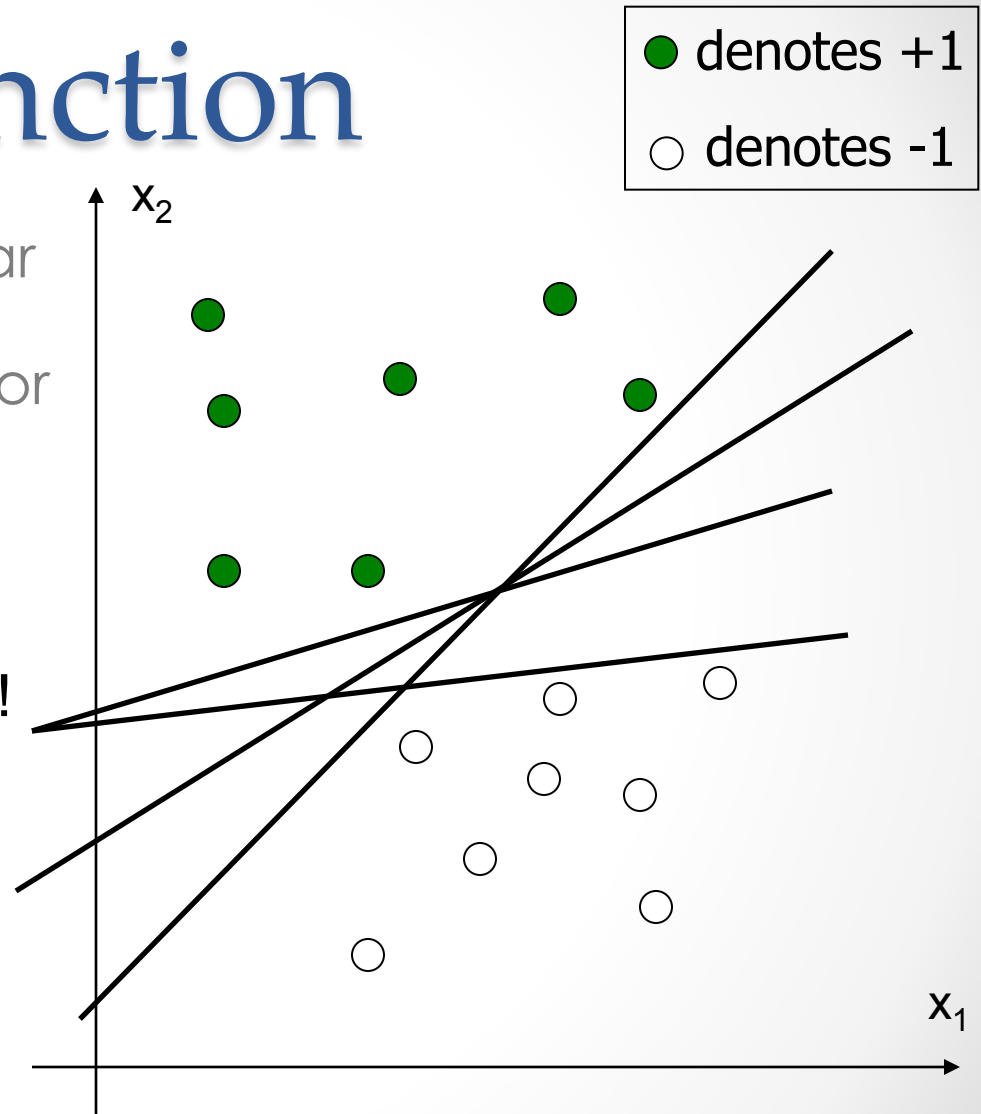
- How would you classify these points using a linear discriminant function in order to minimize the error rate?

- Infinite number of answers!



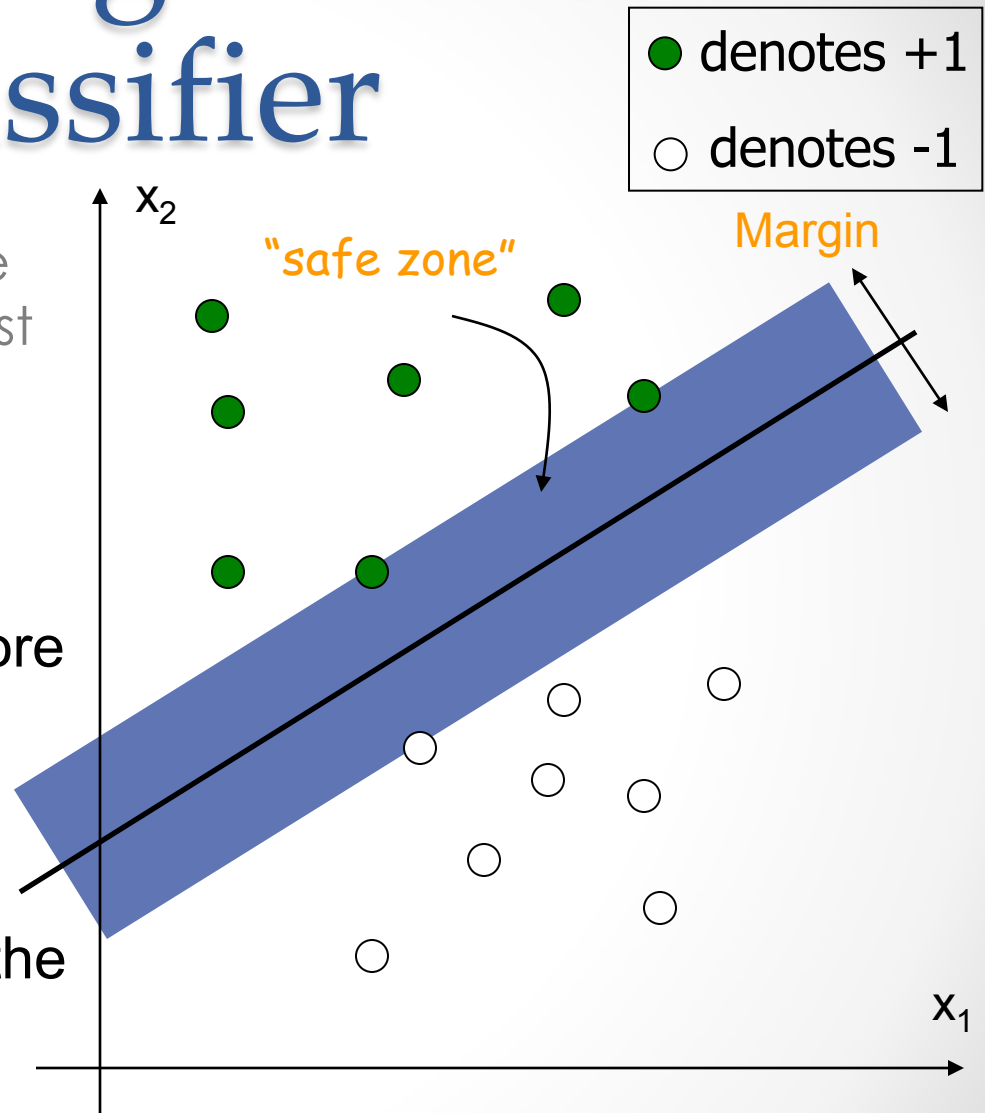
# Linear Discriminant Function

- 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?



# 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
- Why it is the best?
  - The larger the margin the better generalization
  - Robust to outliers



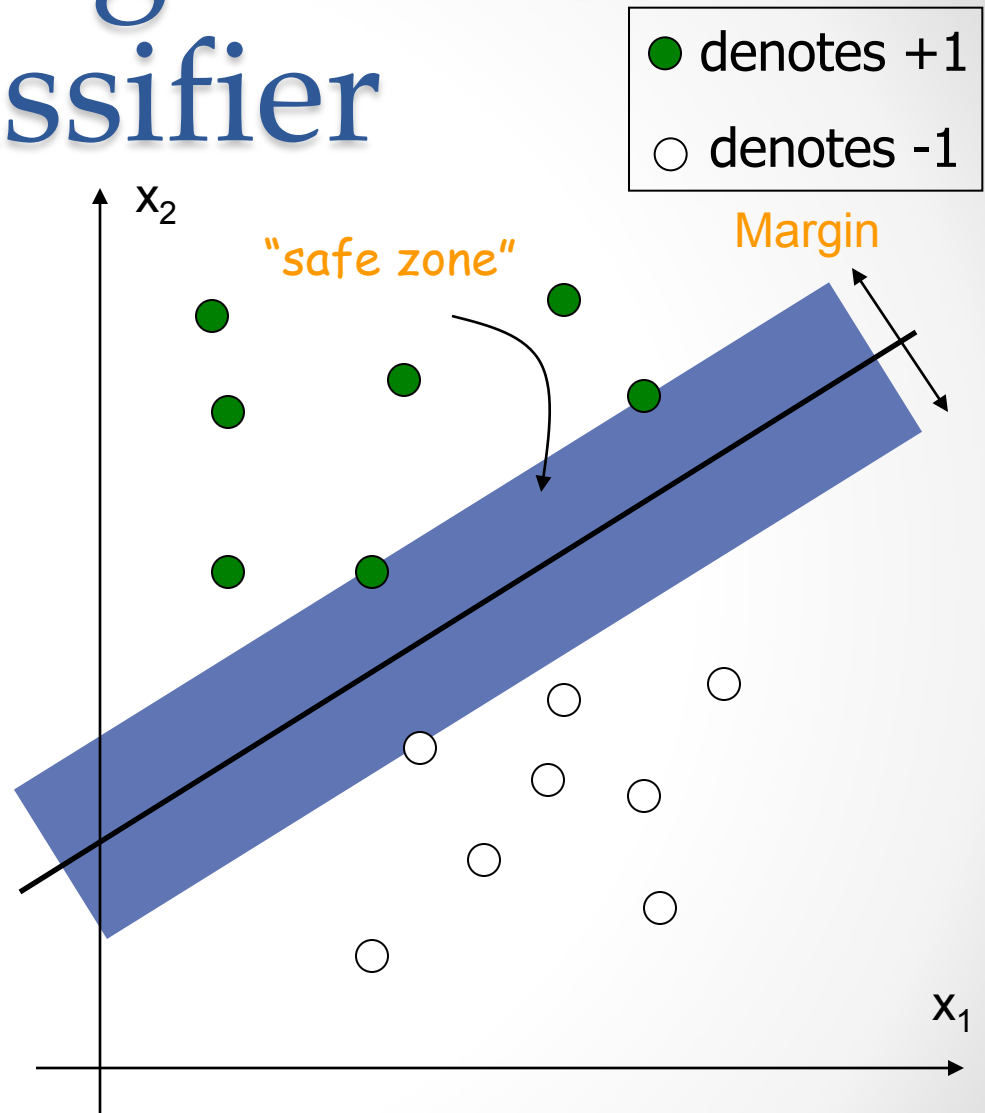
# Large Margin Linear Classifier

- Aim: Learn a large margin classifier.
- Given a set of data points, define:

For  $y_i = +1$ ,  $\mathbf{w}^T \mathbf{x}_i + b \geq 1$

For  $y_i = -1$ ,  $\mathbf{w}^T \mathbf{x}_i + b \leq -1$

- Give an algebraic expression for the width of the margin.



# Algebraic Expression for Width of a Margin

Given 2 parallel lines with equations

$$ax + by + c_1 = 0$$

and

$$ax + by + c_2 = 0$$

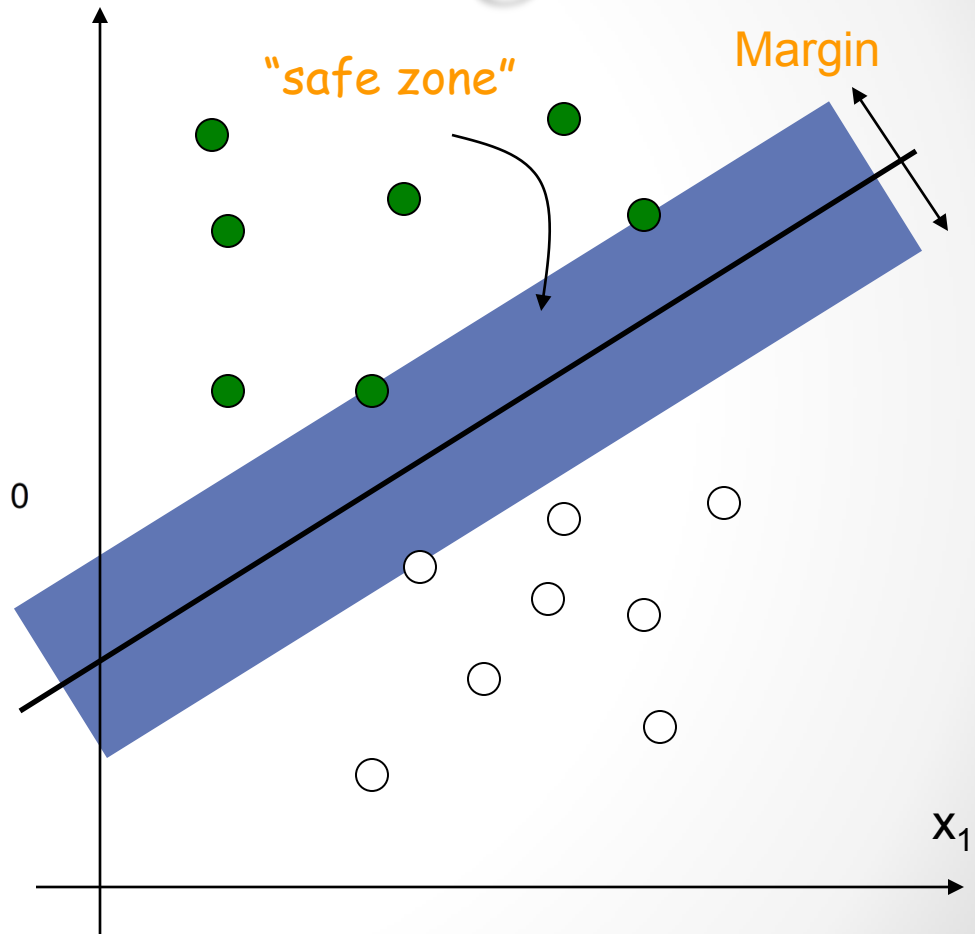
the distance between them is given by:

$$d = \frac{|c_2 - c_1|}{\sqrt{a^2 + b^2}}$$

Our lines in 2-D are:

$$w_1x_1 + w_2x_2 + b - 1 = 0 \text{ and } w_1x_1 + w_2x_2 + b + 1 = 0$$

$$\text{Distance} = \frac{|b - 1 - b - 1|}{\sqrt{w_1^2 + w_2^2}} = \frac{2}{\|\mathbf{w}\|}$$



# Large Margin Linear Classifier

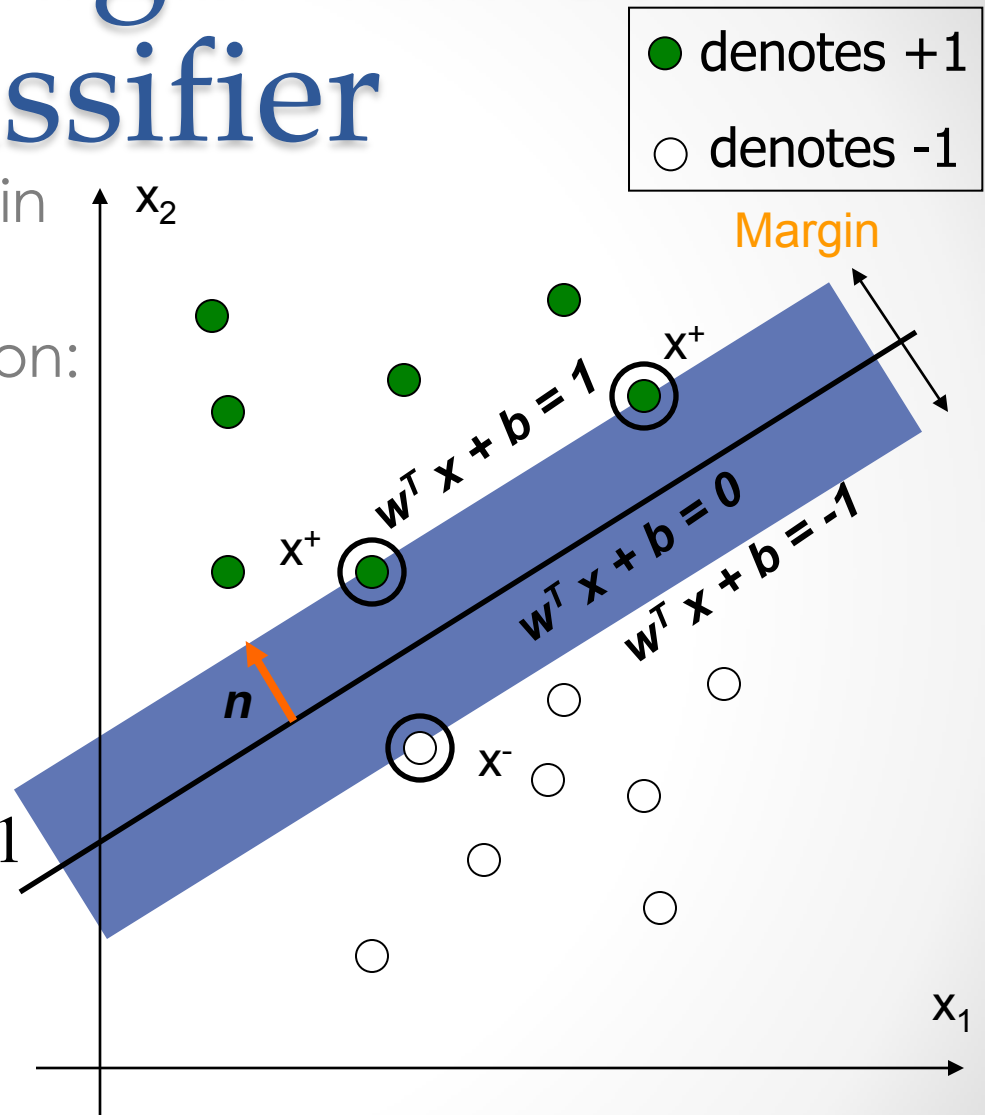
- Aim: Learn a large margin classifier
- Mathematical Formulation:

$$\text{maximize } \frac{2}{\|\mathbf{w}\|}$$

such that

$$\text{For } y_i = +1, \quad \mathbf{w}^T \mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T \mathbf{x}_i + b \leq -1$$



# Large Margin Linear Classifier

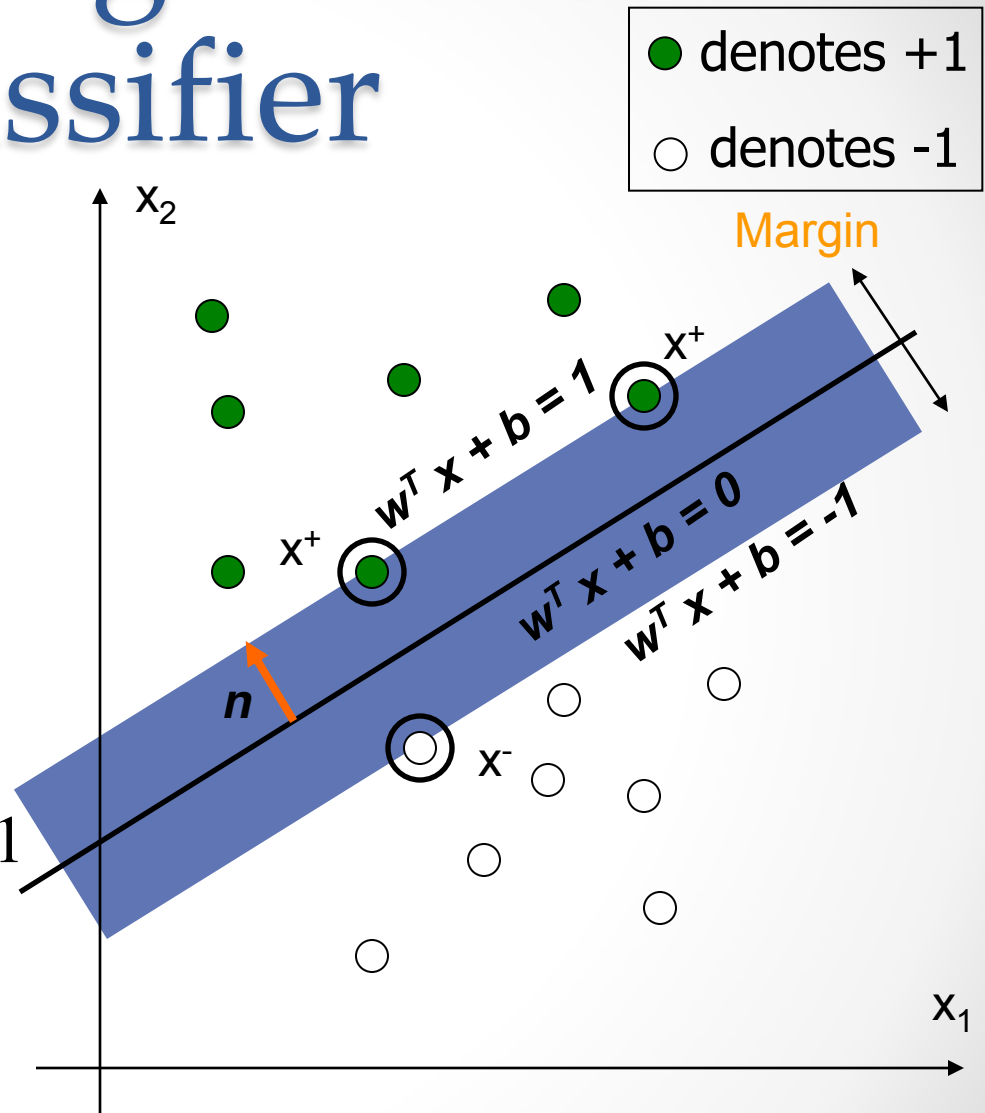
- Formulation:

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

such that

$$\text{For } y_i = +1, \quad \mathbf{w}^T \mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T \mathbf{x}_i + b \leq -1$$



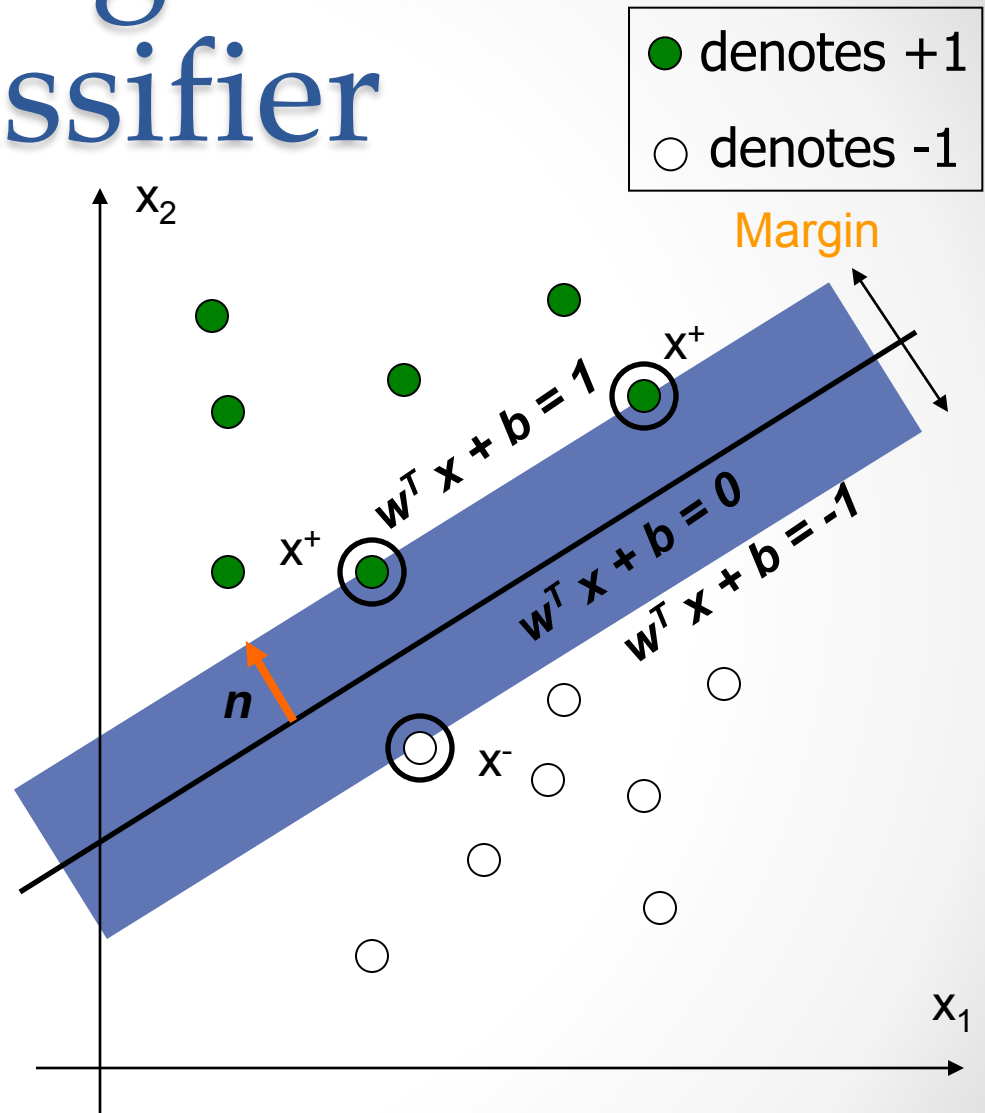
# Large Margin Linear Classifier

- Formulation:

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

such that

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

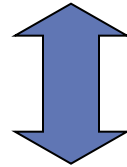




# Solving the Optimization Problem

Quadratic  
programming  
with linear  
constraints

$$\begin{array}{ll}\text{minimize} & \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{s.t.} & y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1\end{array}$$



Lagrangian  
Function

$$\begin{array}{ll}\text{minimize} & L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i (y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1) \\ \text{s.t.} & \alpha_i \geq 0\end{array}$$

# Solving the Optimization Problem

$$\begin{aligned} \text{minimize } L_p(\mathbf{w}, b, \alpha_i) &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i (y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1) \\ \text{s.t. } \alpha_i &\geq 0 \end{aligned}$$

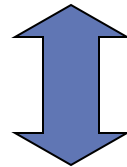
$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad \longrightarrow \quad \mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$

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

# Solving the Optimization Problem

$$\begin{aligned} \text{minimize } L_p(\mathbf{w}, b, \alpha_i) &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i (y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1) \\ \text{s.t. } \alpha_i &\geq 0 \end{aligned}$$

Lagrangian Dual  
Problem



$$\begin{aligned} \text{maximize } \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t. } \alpha_i \geq 0, \text{ and } \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$

# Solving the Optimization Problem

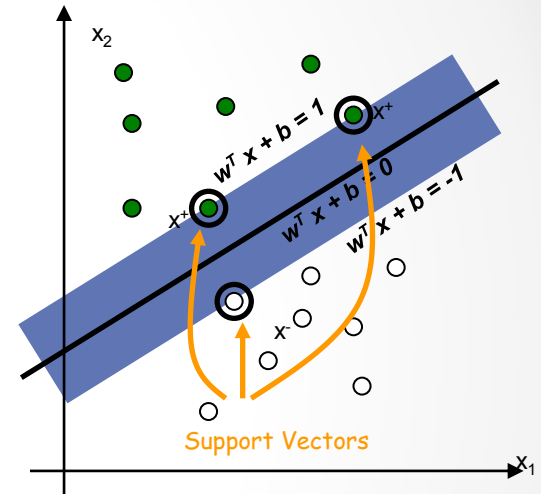
- From the equations, we can prove that: (KKT conditions):

$$\alpha_i \left( y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 \right) = 0$$

- Thus, only support vectors have  $\alpha_i \neq 0$
- The solution has the form:

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

get  $b$  from  $y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0$ ,  
where  $\mathbf{x}_i$  is support vector



# Solving the Optimization Problem

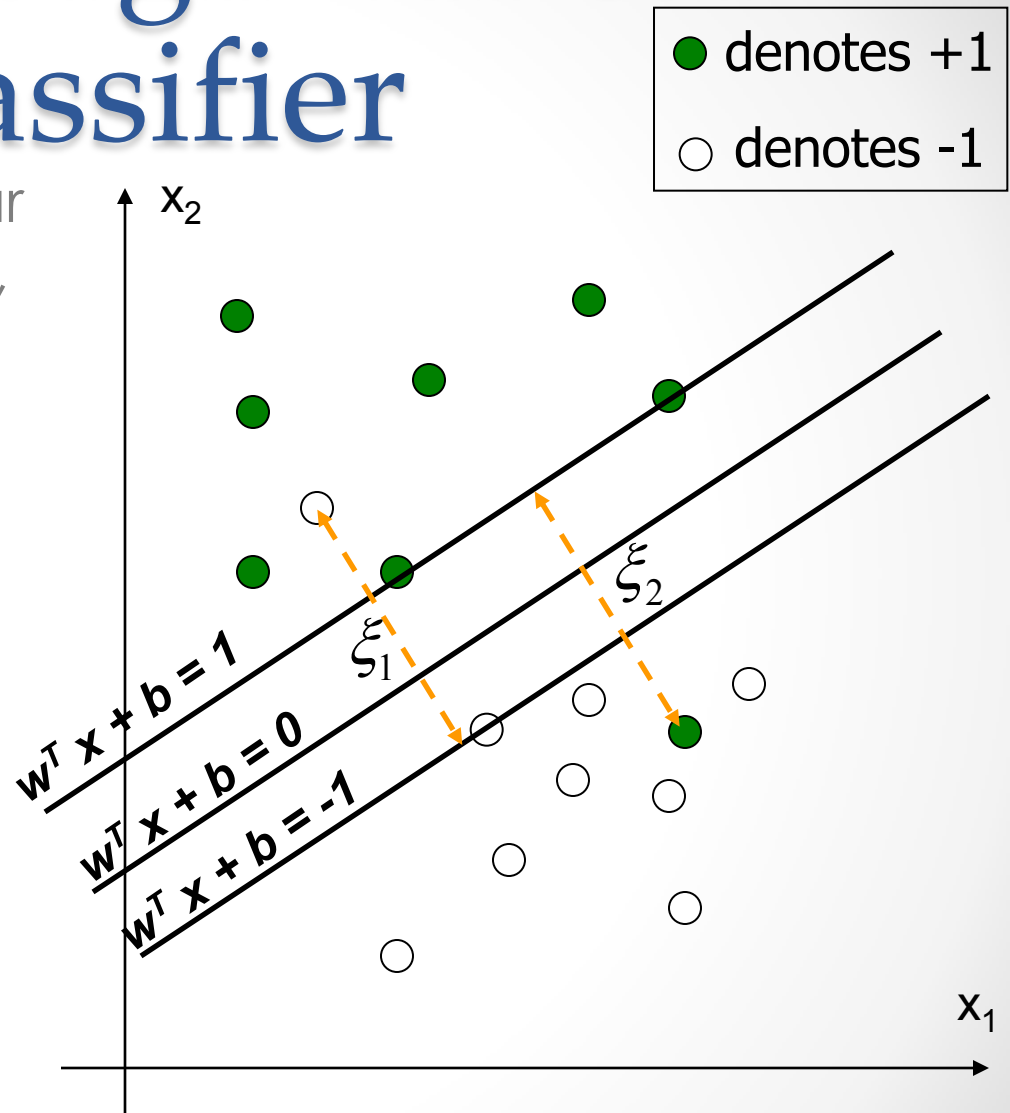
- The linear discriminant function is:

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

- Notice it relies on a *dot product* between the test point  $\mathbf{x}$  and the support vectors  $\mathbf{x}_i$
- Also keep in mind that solving the optimization problem involved computing the *dot products*  $\mathbf{x}_i^T \mathbf{x}_j$  between all pairs of training points

# Large Margin Linear Classifier

- What if data is not linear separable? (noisy data, outliers, etc.)
- Slack variables  $\xi_i$  can be added to allow misclassification of difficult or noisy data points



# Large Margin Linear Classifier

- Formulation:

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

such that

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

$$\xi_i \geq 0$$

- Parameter  $C$  can be viewed as a way to control over-fitting.

# Large Margin Linear Classifier

- Formulation: (Lagrangian Dual Problem)

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

such that

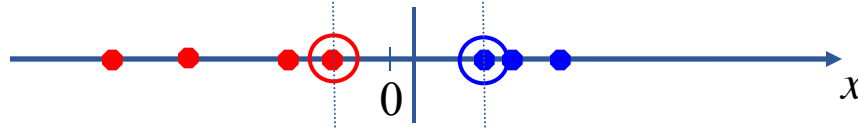
$$0 \leq \alpha_i \leq C$$

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



# Non-linear SVMs

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



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



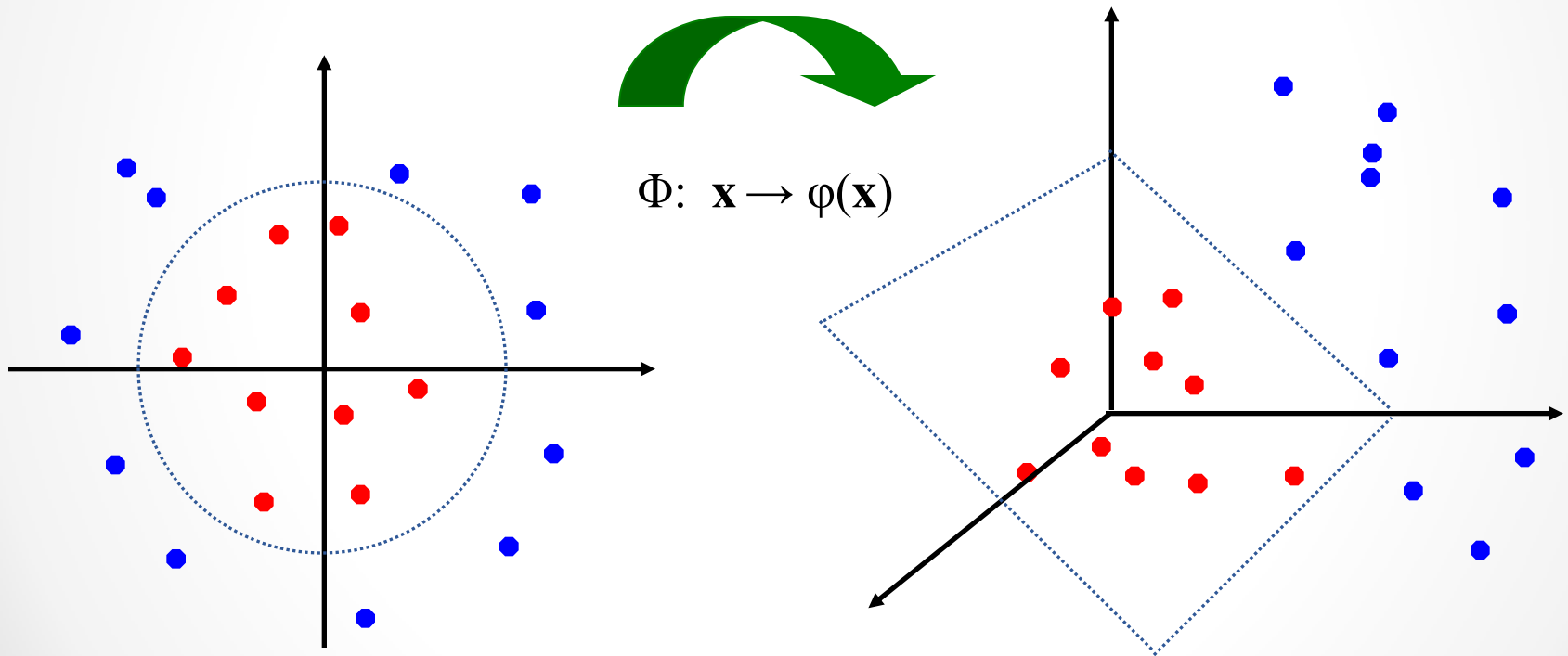
- Kernel Trick!!!
  - SVM = Linear SVM + Kernel Trick

# Kernel Trick Motivation

- **Linear classifiers** are well understood, widely-used and efficient.
- How to use linear classifiers to build non-linear ones?
- **Neural networks:** Construct non-linear classifiers by using a network of linear classifiers (perceptrons).
- **Kernels:**
  - Map the problem from the input space to a new higher-dimensional space (called the feature space) by doing a non-linear transformation using a special function called the kernel.
  - Then use a linear model in this new high-dimensional feature space. The linear model in the feature space corresponds to a non-linear model in the input space.

# 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:



# Nonlinear SVMs: The Kernel Trick

- With this mapping, our discriminant function is now:

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

- No need to know this mapping 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:

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

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

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

# 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:

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

- Sigmoid:

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

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

# Nonlinear SVM: Optimization

- Formulation: (Lagrangian Dual Problem)

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

$$\text{such that } 0 \leq \alpha_i \leq C$$
$$\sum_{i=1}^n \alpha_i y_i = 0$$

- The solution of the discriminant function is

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

- 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.  $\sigma$  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

# 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/>

# Demo of LibSVM

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