An Introduction of Support Vector Machine

Junying (Joanna) Zhang

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)

V. Vapnik

Outline

- Linear Discriminant Function

- Large Margin Linear Classifier

- Nonlinear SVM: The Kernel Trick

- Demo of SVM

Discriminant Function

- the classifier is said to assign a feature vector *x* to class *w_i* if

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \qquad \text{for all} \quad j \neq i$$

- For two-category case, $g(\mathbf{x}) \equiv g_1(\mathbf{x}) - g_2(\mathbf{x})$
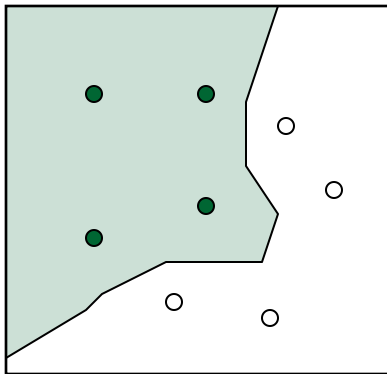
$$\text{Decide } \omega_1 \text{ if } g(\mathbf{x}) > 0; \text{ otherwise decide } \omega_2$$

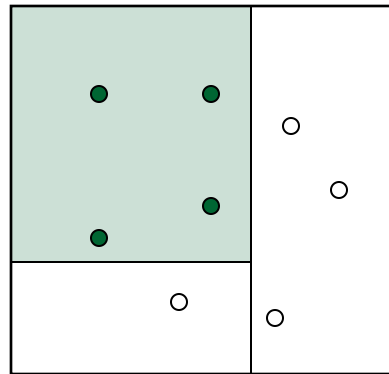- An example we've learned before:
  - Minimum-Error-Rate Classifier

$$g(\mathbf{x}) \equiv p(\omega_1 \mid \mathbf{x}) - p(\omega_2 \mid \mathbf{x})$$
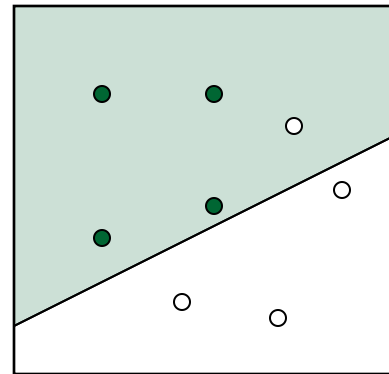
Discriminant Function

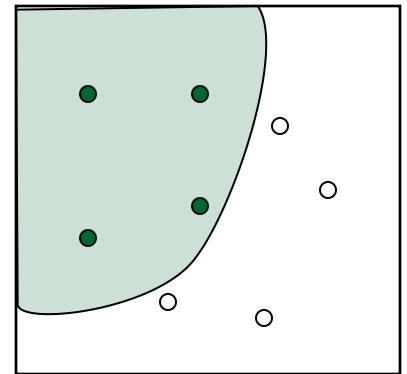- It can be arbitrary functions of *x*, such as:

Nearest Neighbor	Decision Tree	Linear Functions	Nonlinear Functions

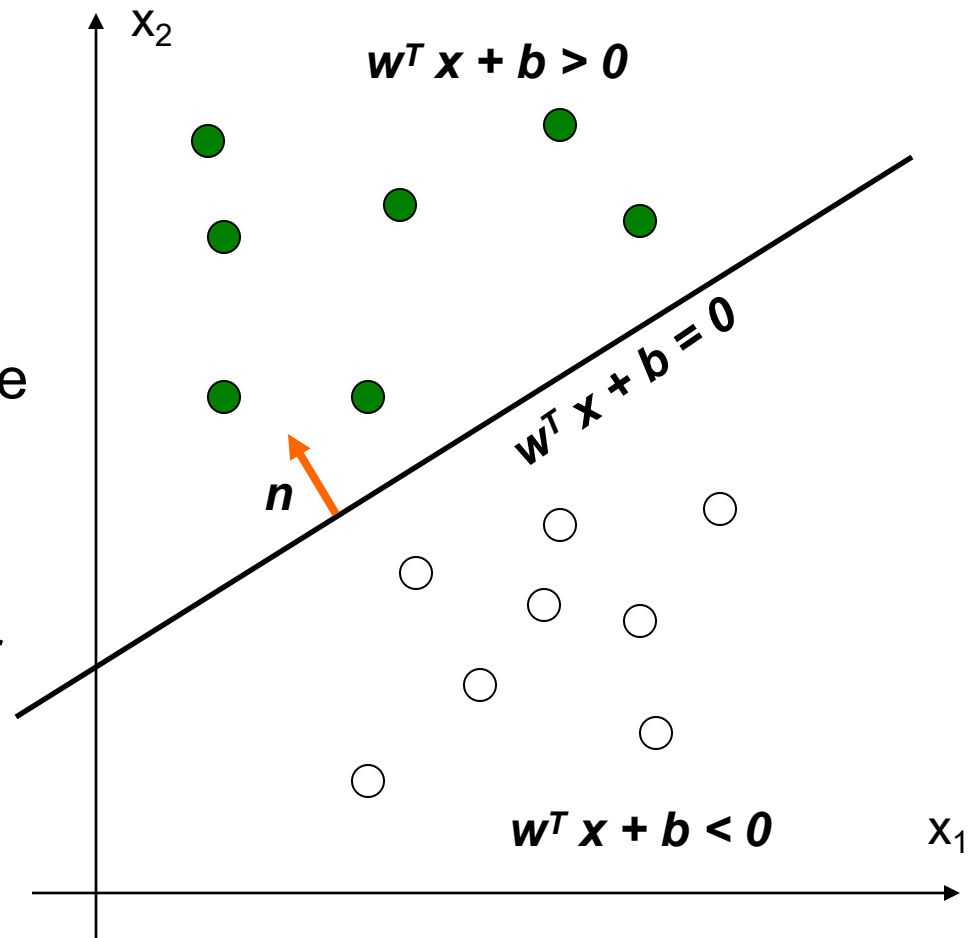$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

Linear Discriminant Function

- g(x) is a linear function:

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

- A hyper-plane in the feature space

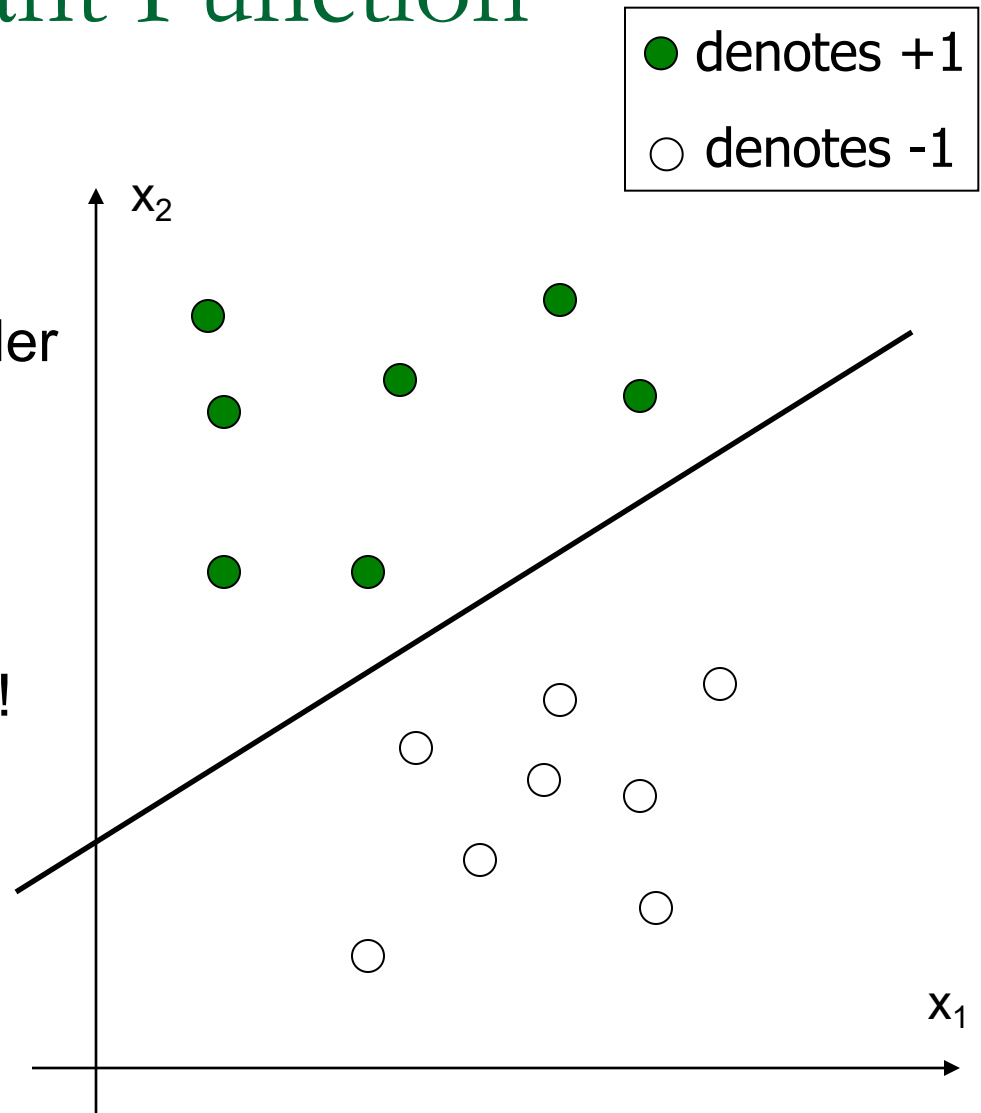- (Unit-length) normal vector of the hyper-plane:

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

x_2

$w^T x + b > 0$

$w^T x + b = 0$

n

$w^T x + b < 0$

x_1

Linear Discriminant Function

denotes +1
denotes -1

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

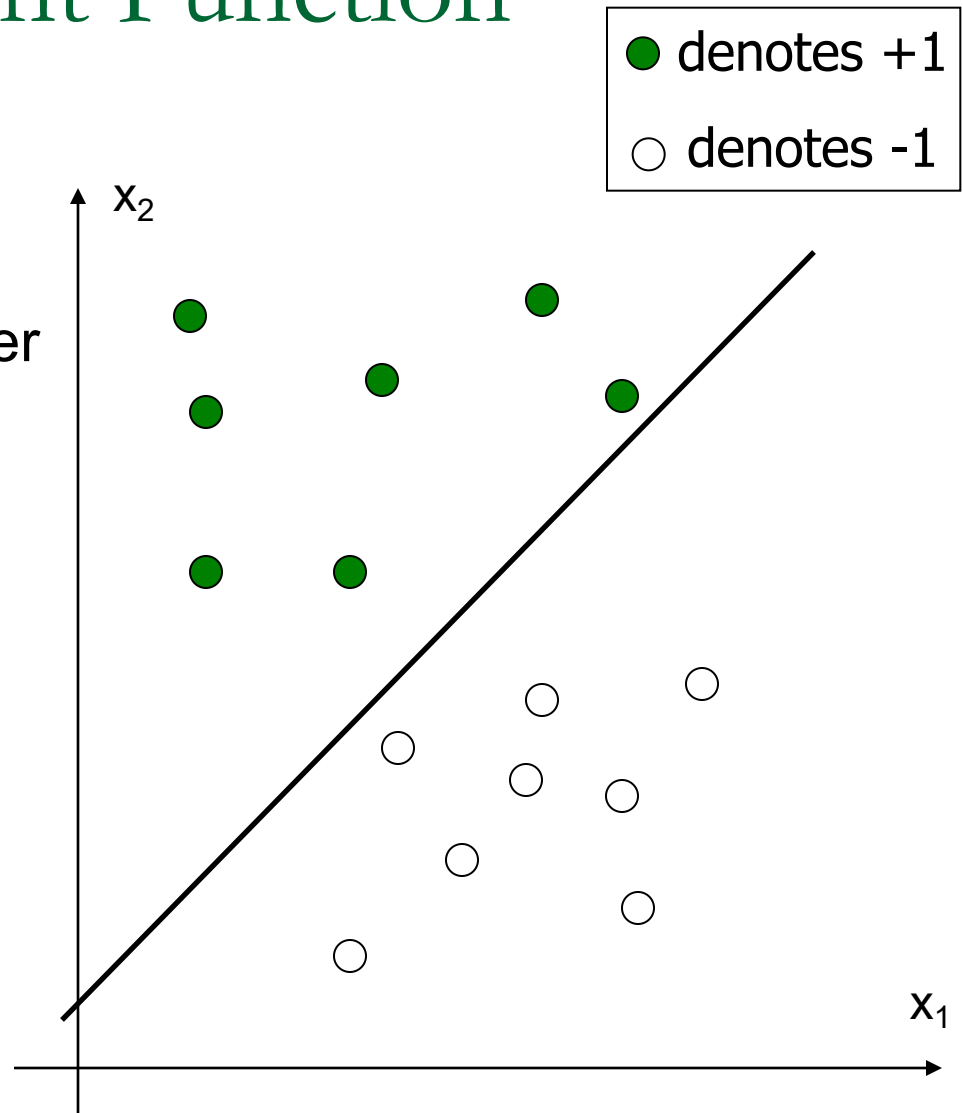- Infinite number of answers!

x_2

x_1

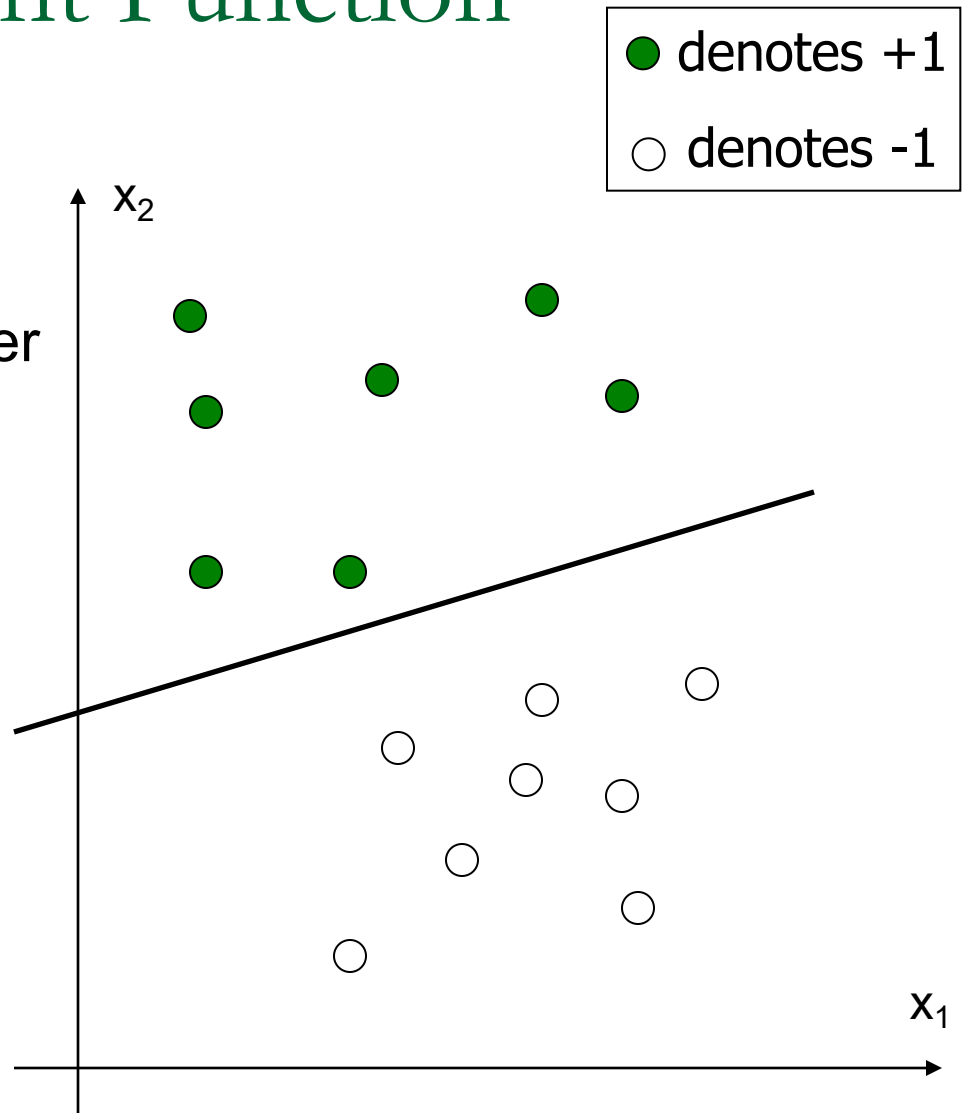# Linear Discriminant Function

- 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

●	denotes +1
○	denotes -1

- 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

● denotes +1

○ denotes -1

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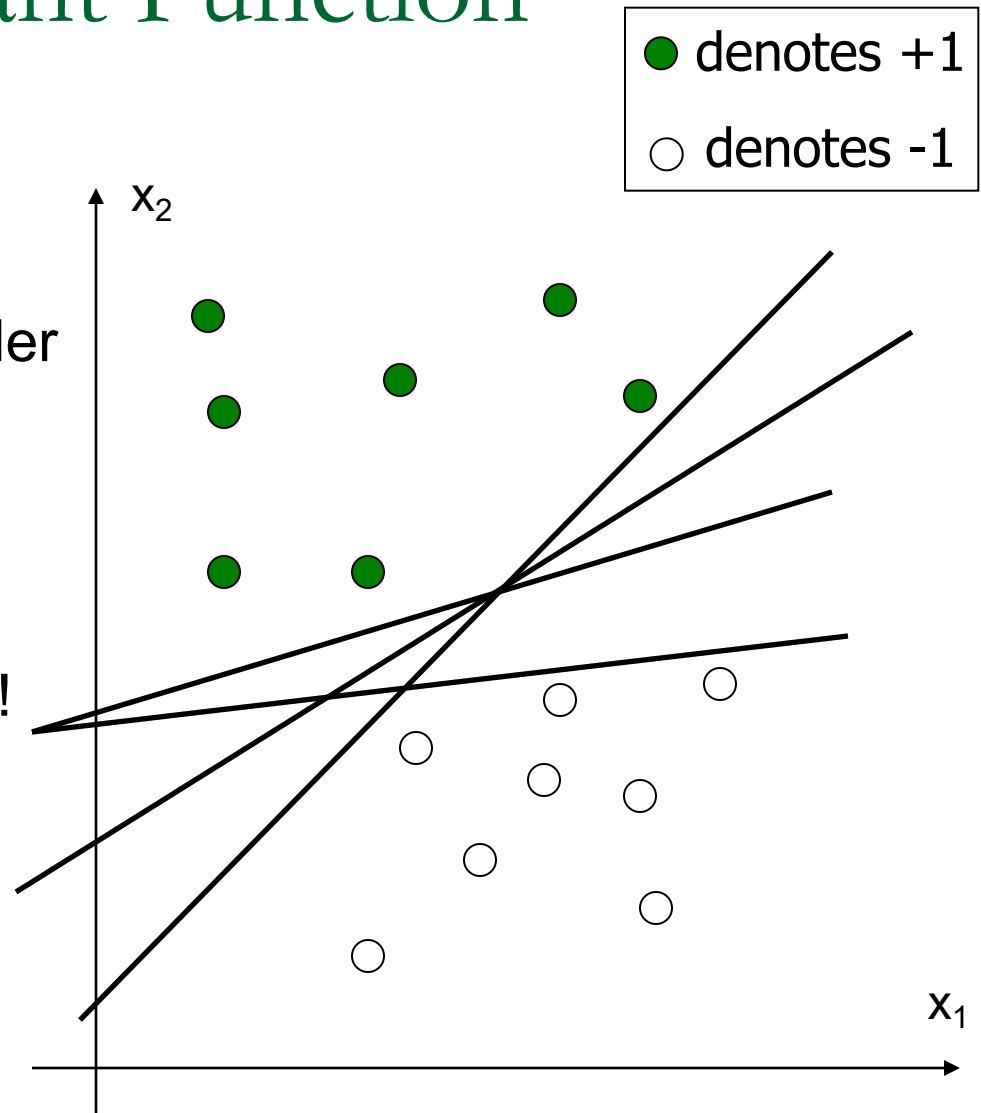
- Infinite number of answers!

- Which one is the best?

x_2

x_1

Large Margin Linear Classifier

- **denotes +1**
- ○ **denotes -1**

- 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?
  - Robust to outliners and thus strong generalization ability

x_2

"safe zone"

Margin

x_1
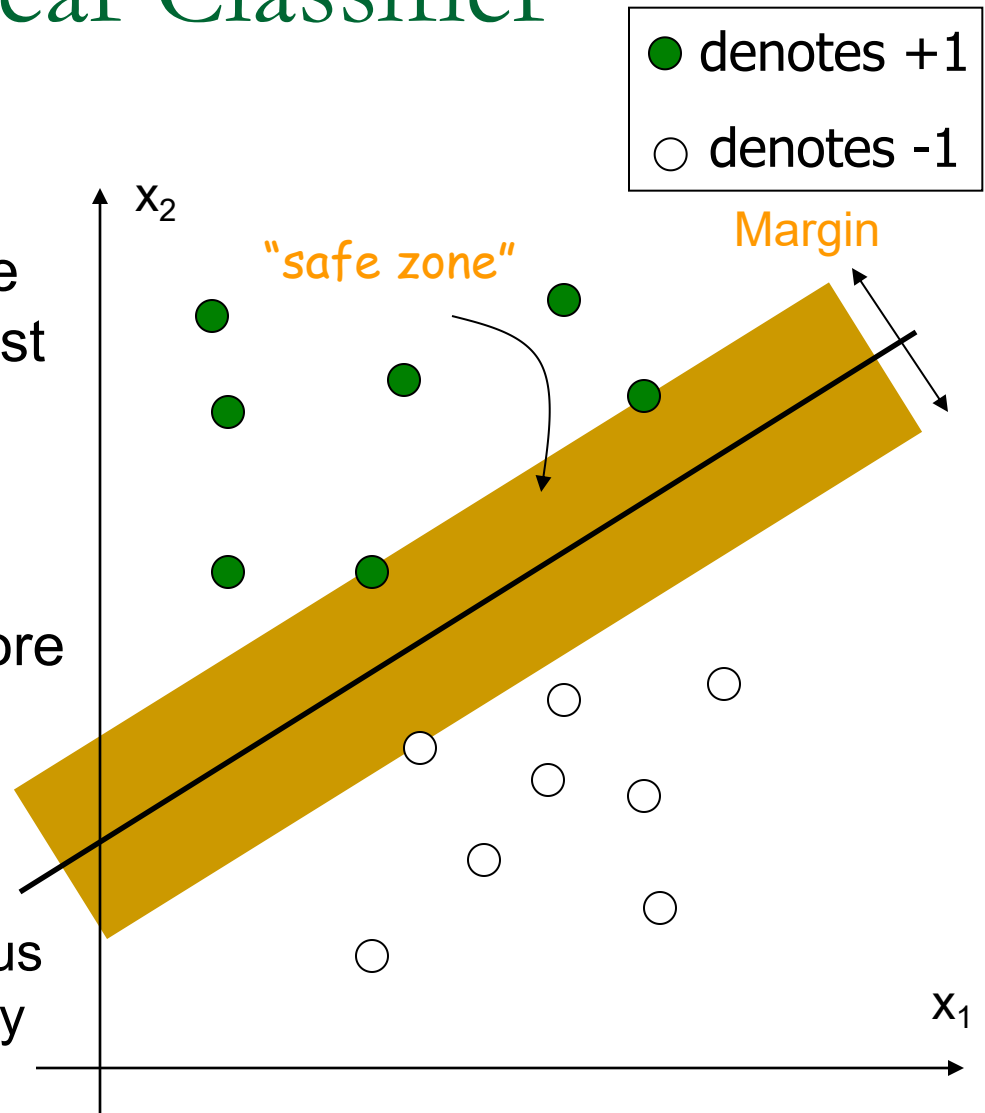
Large Margin Linear Classifier

● denotes +1
○ denotes -1

- Given a set of data points:

$$\{(\mathbf{x}_i, y_i)\}, \ i = 1, 2, \cdots, n, \text{ where}$$
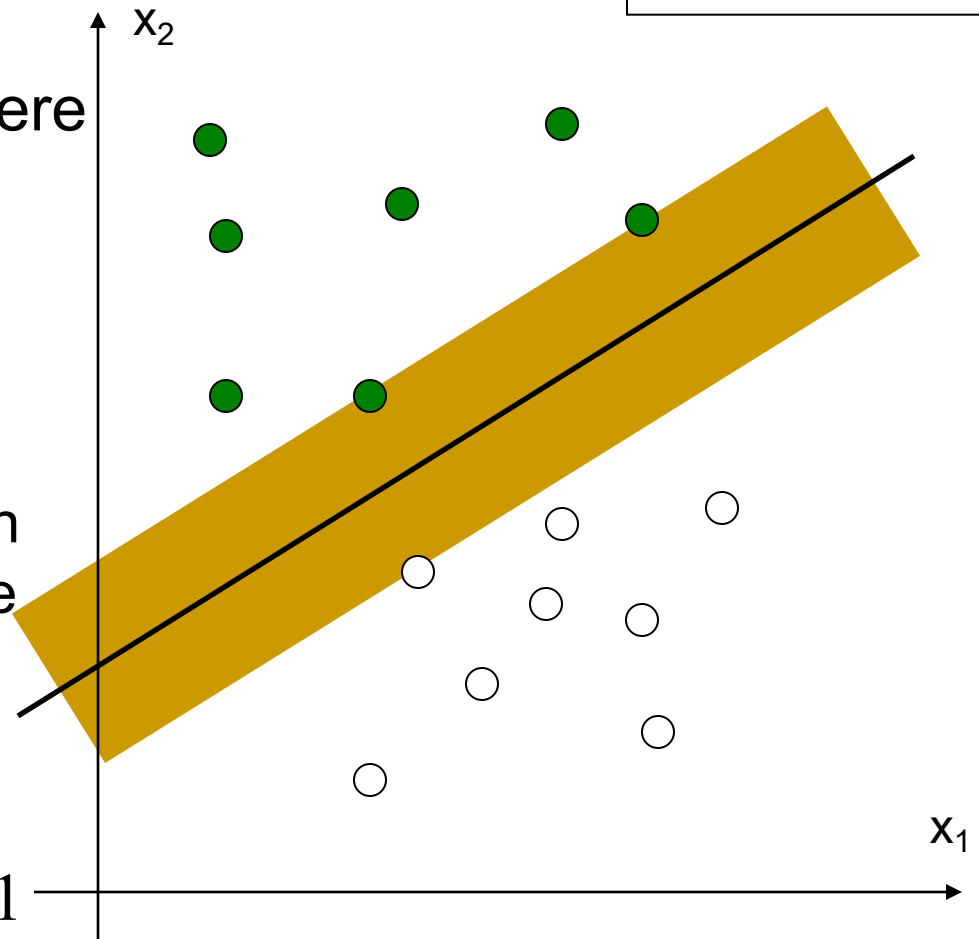
$$\text{For } y_i = +1, \ \mathbf{w}^T \mathbf{x}_i + b > 0$$

$$\text{For } y_i = -1, \ \mathbf{w}^T \mathbf{x}_i + b < 0$$

- With a scale transformation on both *w* and *b*, the above is equivalent to

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

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

x_2

x_1

Large Margin Linear Classifier

- We know that

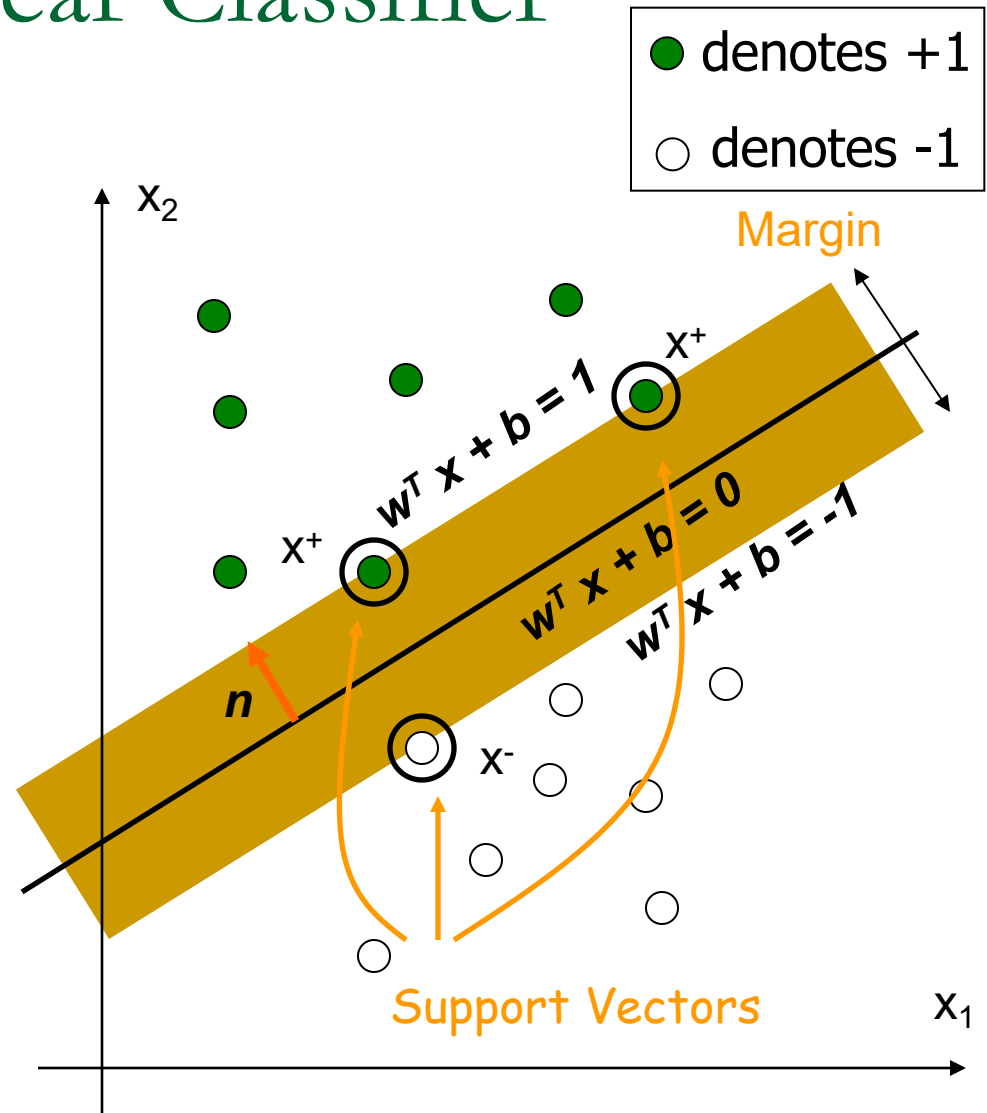$$\mathbf{w}^T\mathbf{x}^+ + b = 1$$

$$\mathbf{w}^T\mathbf{x}^- + b = -1$$

- The margin width is:

$$M = (\mathbf{x}^+ - \mathbf{x}^-) \cdot \mathbf{n}$$

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

- denotes +1
- denotes -1

Margin

x_2

$w^T x + b = 1$

x^+

$w^T x + b = 0$

x^+

$w^T x + b = -1$

n

x^-

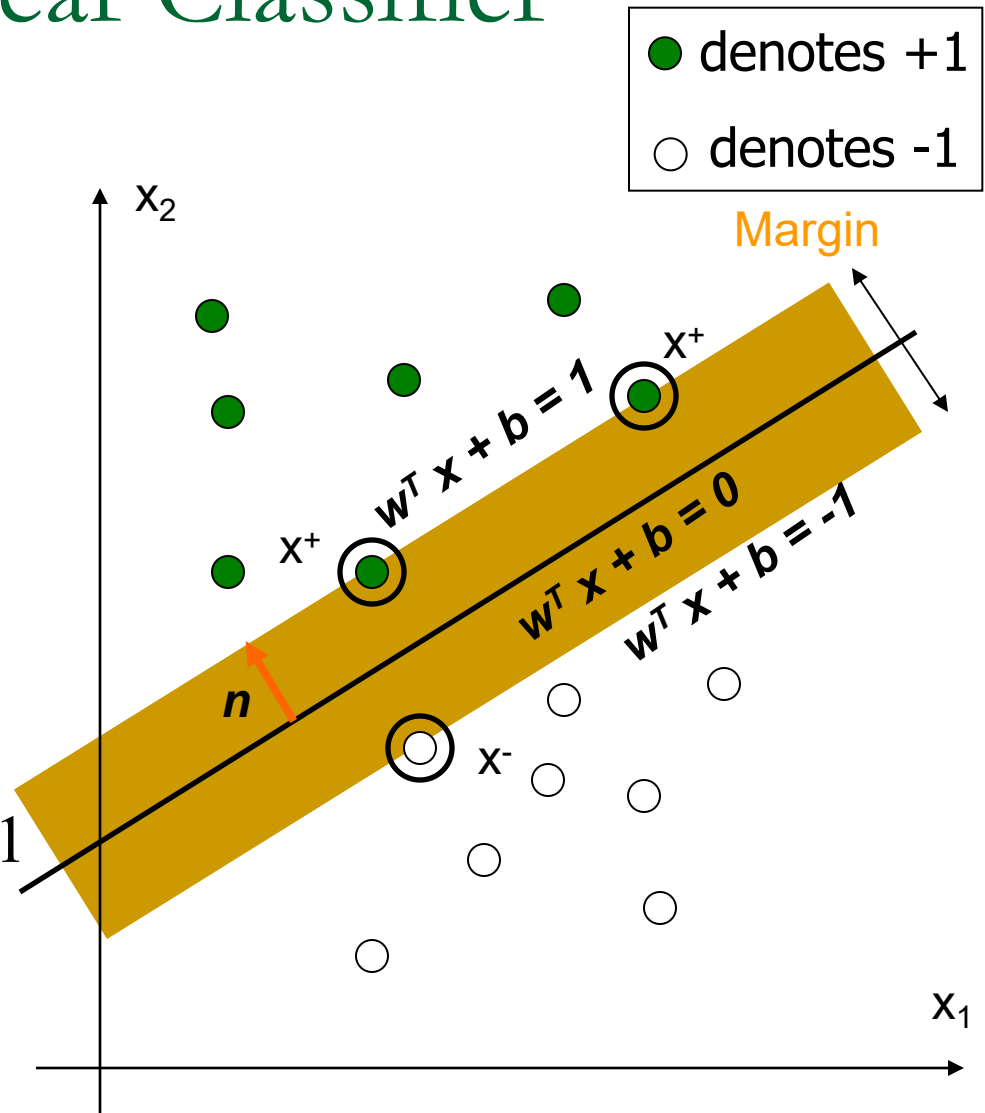Support Vectors

x_1

Large Margin Linear Classifier

- Formulation:

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

such that

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

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

● denotes +1

○ denotes -1

Margin

x_2

$w^T x + b = 1$

$w^T x + b = 0$

$w^T x + b = -1$

x^+

x^+

x^-

n
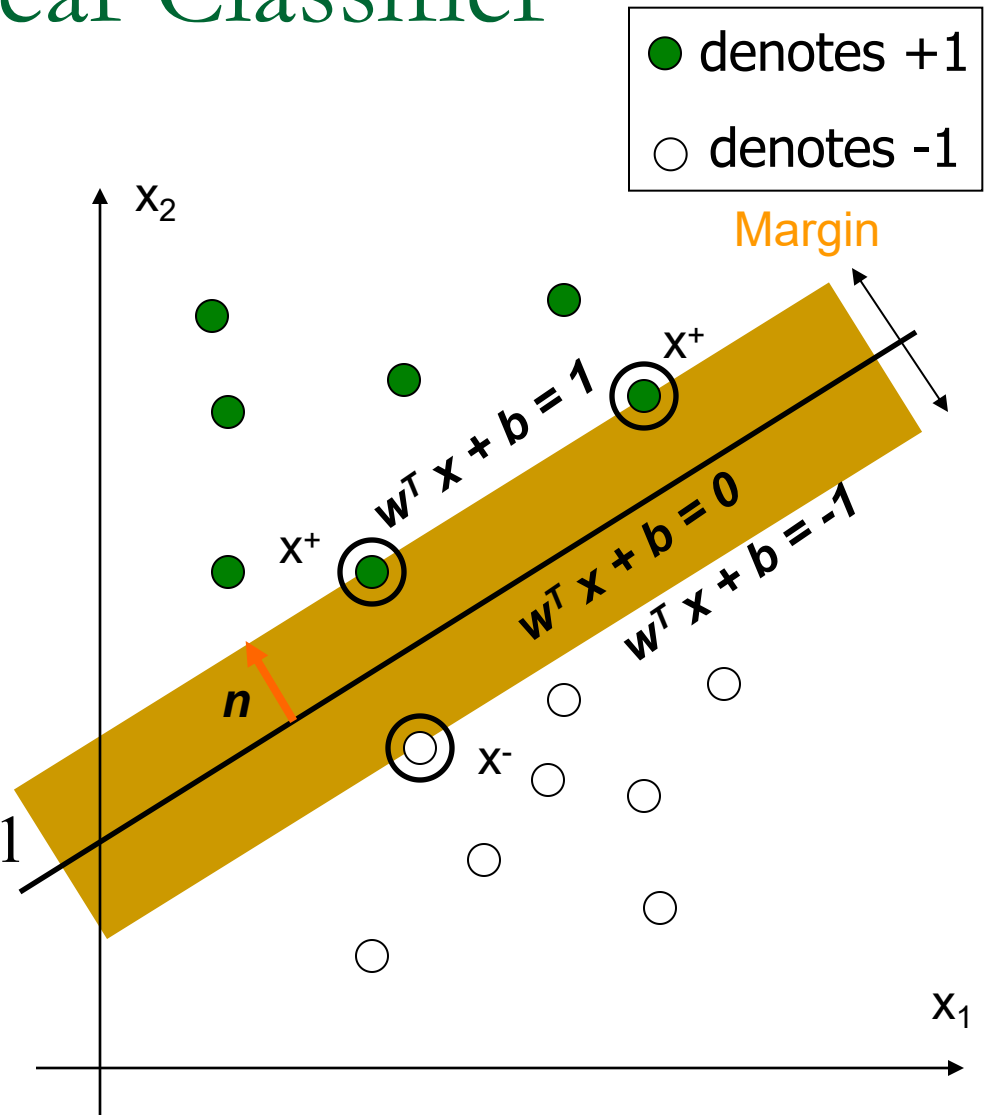
x_1

Large Margin Linear Classifier

- Formulation:

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2$$

such that

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

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

denotes +1
denotes -1

Margin

x_2

x^+

$w^T x + b = 1$

$w^T x + b = 0$

$w^T x + b = -1$
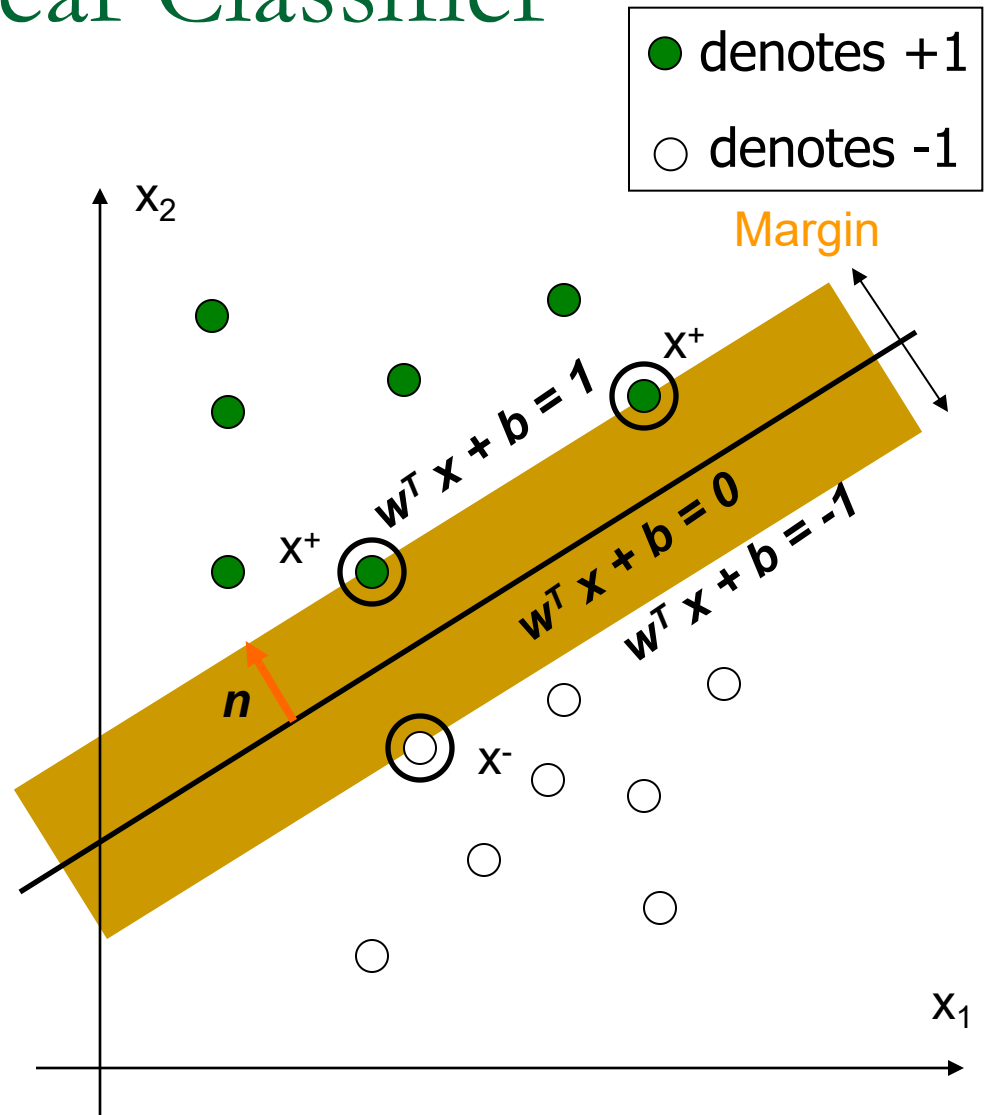
x^+

n

x^-

x_1

Large Margin Linear Classifier

- denotes +1
- denotes -1

Formulation:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2$$

such that

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

Margin

x_2

x_1

$w^T x + b = 1$

$w^T x + b = 0$

$w^T x + b = -1$

x^+

x^+

x^-

n

Solving the Optimization Problem

Quadratic programming with linear constraints

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

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

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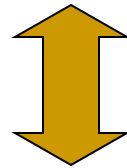
$$\min_{\alpha} \quad L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i\left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right)$$

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

Solving the Optimization Problem

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

$$\text{s.t.} \quad \alpha_i \geq 0$$
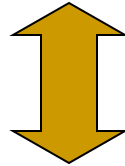
$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \implies \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

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

Solving the Optimization Problem

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

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

Lagrangian Dual
Problem

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

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

Solving the Optimization Problem
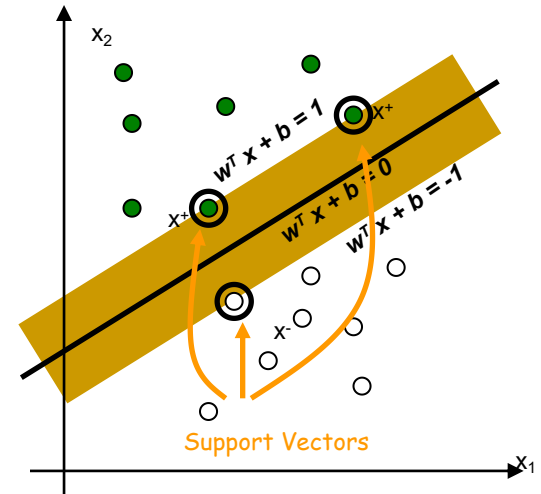
- From KKT condition, we know:

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

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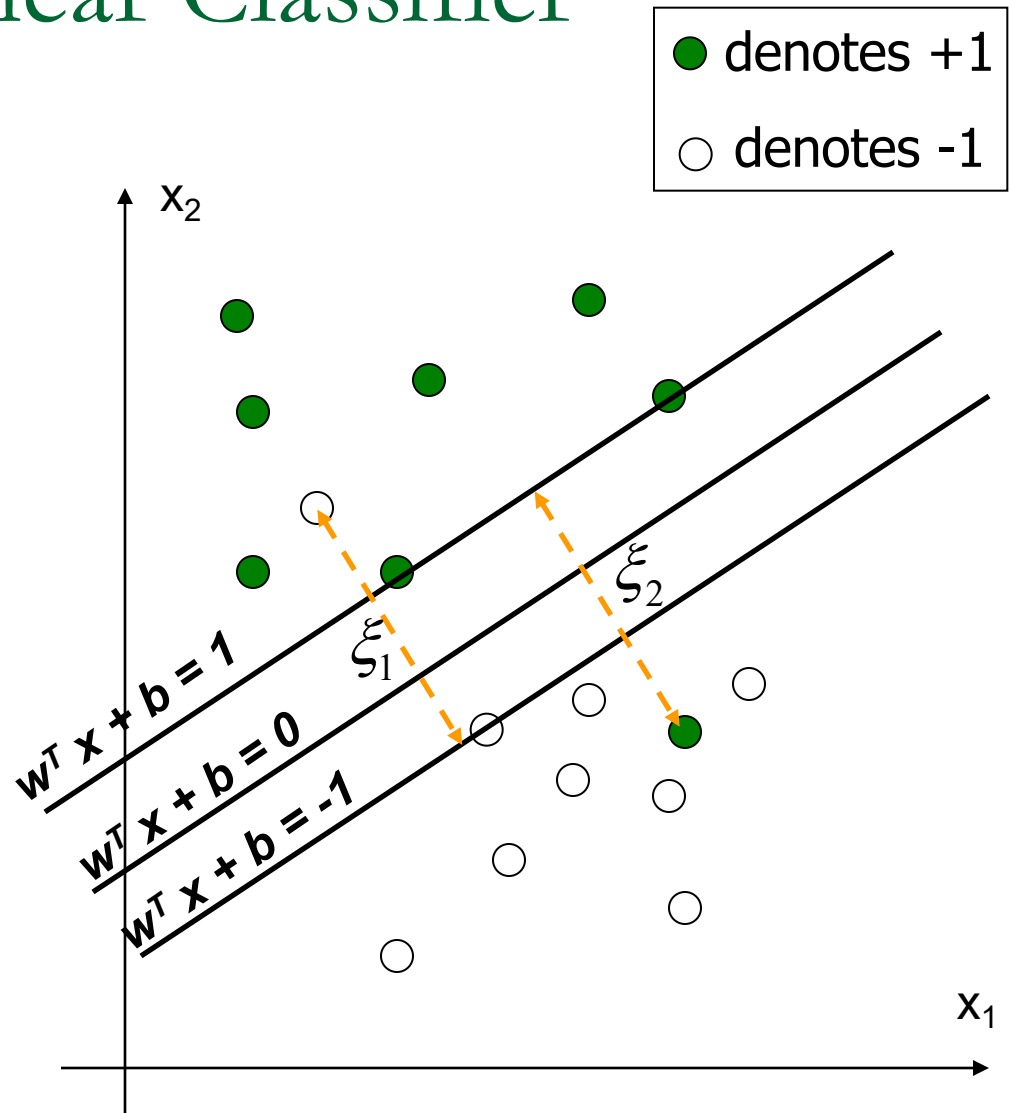
$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \sum_{i \in \text{SV}} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

- Notice it relies on a *dot product* between the test point **x** and the support vectors **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 ξ_i can be added to allow mis-classification of difficult or noisy data points

denotes +1
denotes -1

x_2

$\mathbf{w}^T \mathbf{x} + b = 1$

$\mathbf{w}^T \mathbf{x} + b = 0$

$\mathbf{w}^T \mathbf{x} + b = -1$

ξ_1

ξ_2

x_1

Large Margin Linear Classifier

- Formulation:

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

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)

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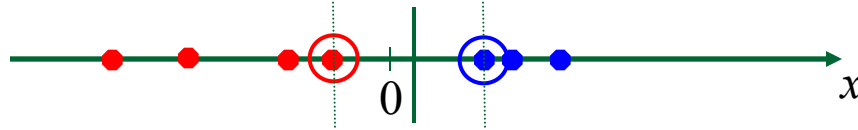
such that

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

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

Non-linear SVMs

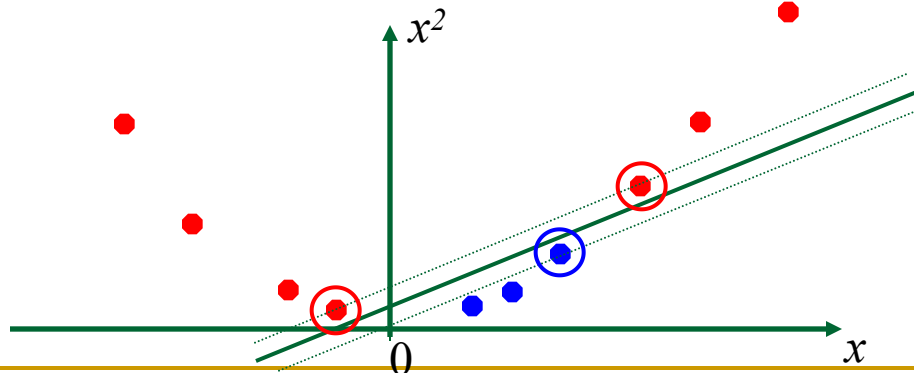- Datasets that are linearly separable with noise work out great:

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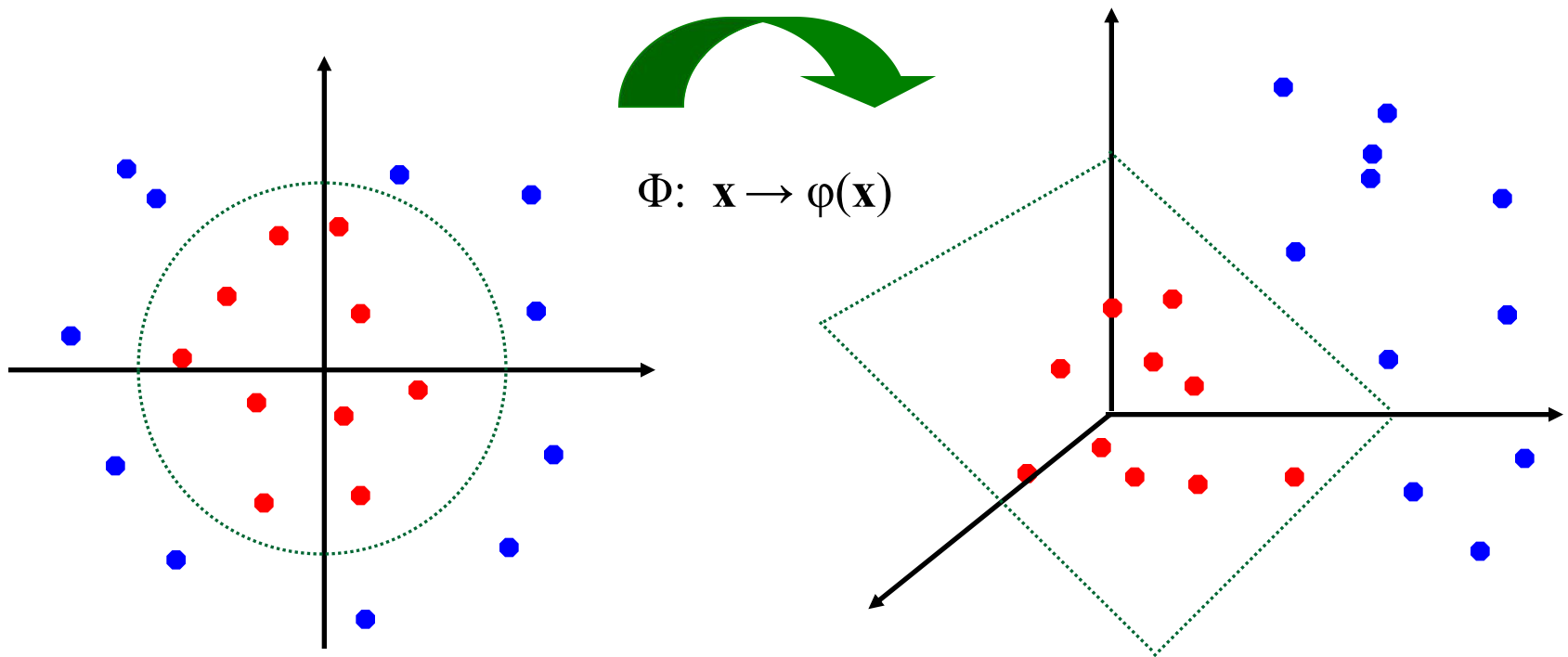

- How about… mapping data to a higher-dimensional space:

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:

$$\Phi: \ \mathbf{x} \to \varphi(\mathbf{x})$$

Nonlinear SVMs: The Kernel Trick

- With this mapping, our discriminant function is now:

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

- 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 $K(\mathbf{x_i},\mathbf{x_j}) = \varphi(\mathbf{x_i})^T\varphi(\mathbf{x_j})$:

$K(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^T\mathbf{x_j})^2$,

$\quad\quad = 1+ x_{i1}^2x_{j1}^2 + 2\,x_{i1}x_{j1}\,x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$

$\quad = [1 \;\; x_{i1}^2 \;\; \sqrt{2}\,x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^T \, [1 \;\; x_{j1}^2 \;\; \sqrt{2}\,x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}]$

$\quad = \varphi(\mathbf{x_i})^T\varphi(\mathbf{x_j}), \quad$ where $\varphi(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\,x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$

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:

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

  - 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)

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

such that

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

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

- The solution of the discriminant function is

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

- 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

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/

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

- http://www.support-vector.net/

- http://www.support-vector.net/icmltutorial.pdf

- http://www.kernelmachines.org/papers/tutorial-nips.ps.gz

- http://www.clopinet.com/isabelle/Projects/SVM/applist.html