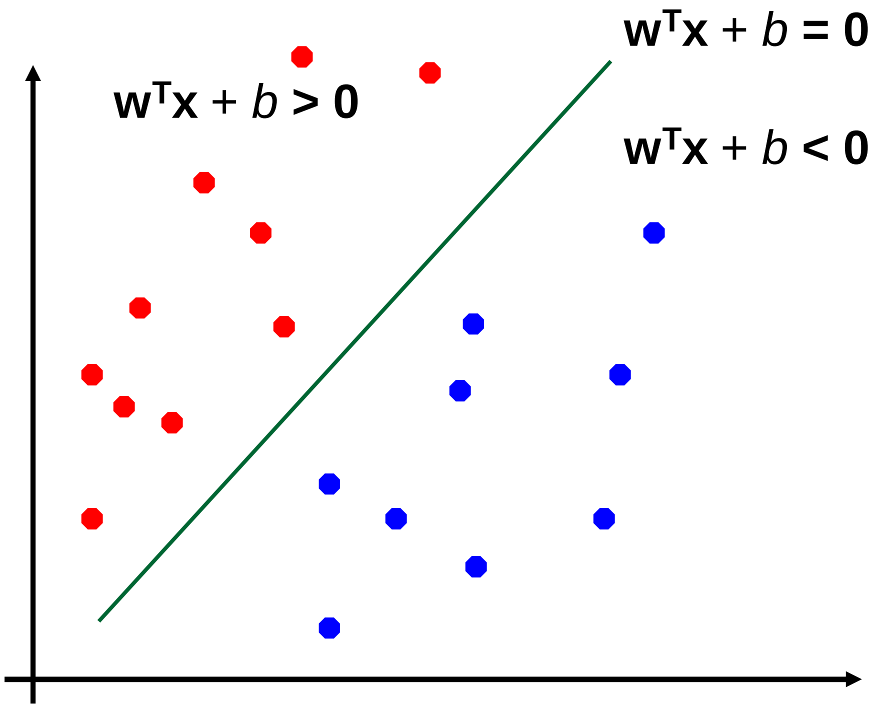
Support Vector Machine

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.

Perceptron Revisited: Linear Separators

- Binary classification can be viewed as the task of separating classes in feature space:


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

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

Support Vector Machines

- SVM are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis
- SVM Linear classifier
  - Is a classification algorithm
  - Steps:
    - Given a set of training examples, each marked/labeled as one of two categories
    - SVM training algorithm builds a model
    - The model assigns new unseen data into one of the two categories

What's so Special?

Hyperplane

- SVM constructs a hyperplane or a set of hyperplanes in a high-dimensional space that separates the two classes.

- What is a hyperplane?
  - Line equation is y=ax+b
  - Hyperplane equation is $w^T.x$

$$y - ax - b = 0$$

Given two vectors w$\begin{pmatrix} -b \\ -a \\ 1 \end{pmatrix}$ $and\ x \begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$

We have

$$w^T x = -b \times 1 + (-a) \times x + 1 \times y$$
$$w^T x = y - ax - b$$

Both are two different ways of representing the same thing.

Then why Hyperplane equation? – easier to work with in many dimensions

Assumptions – For now..

There are only two classes in the given data. Every training example belongs to one of the two classes (say positive class and negative class)

Given data is linearly separable

Discriminant Function

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

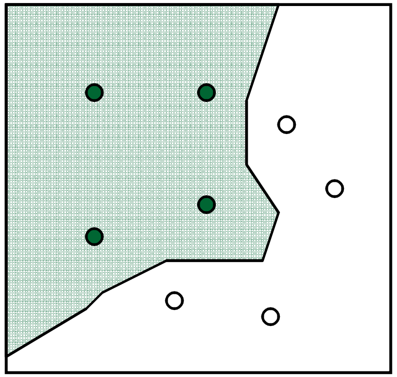
$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \qquad \text{for all} \quad j \neq i$$

- For two-category case, $g(\mathbf{x}) \equiv g_1(\mathbf{x}) - g_2(\mathbf{x})$

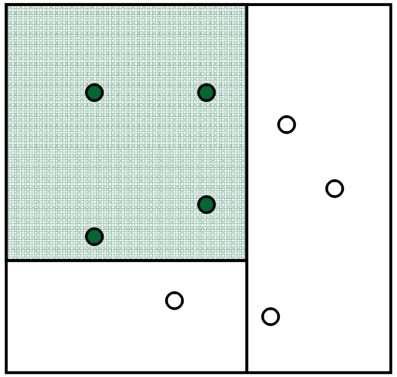
$$\text{Decide } \omega_1 \text{ if } g(\mathbf{x}) > 0; \text{ otherwise decide } \omega_2$$

Discriminant Function

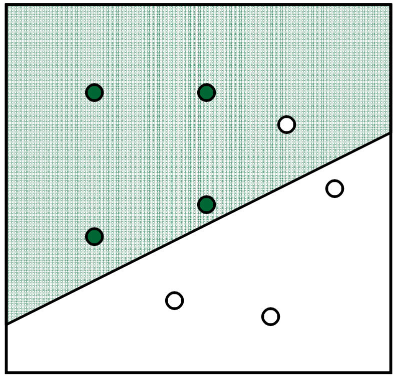
- It can be arbitrary functions of *x*, such as:



Nearest Neighbor

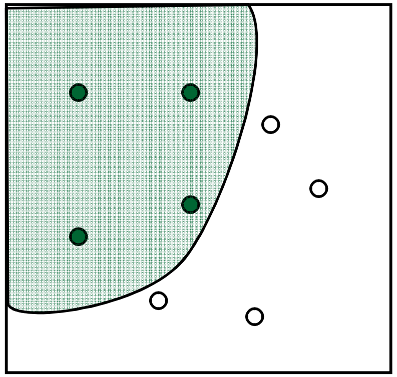


Decision Tree



Linear Functions

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



Nonlinear Functions

Linear Discriminant Function

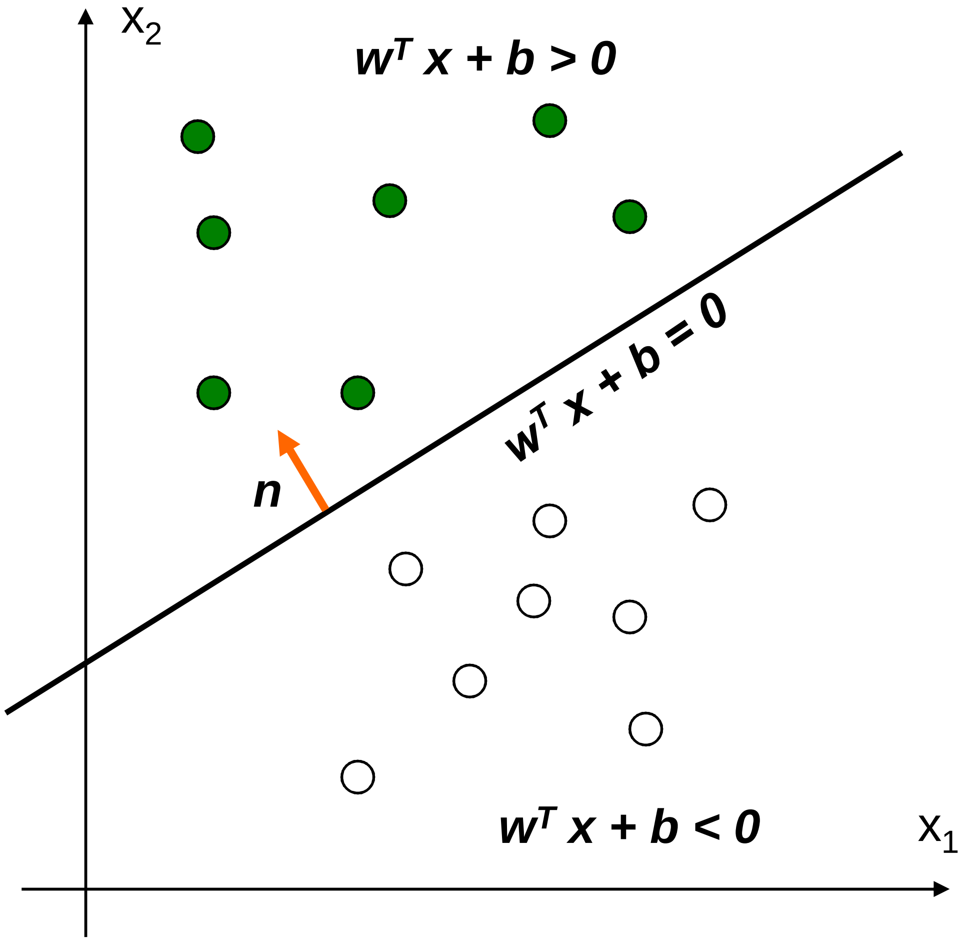
- g(x) is a linear function:

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

- A hyper-plane in the feature space

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

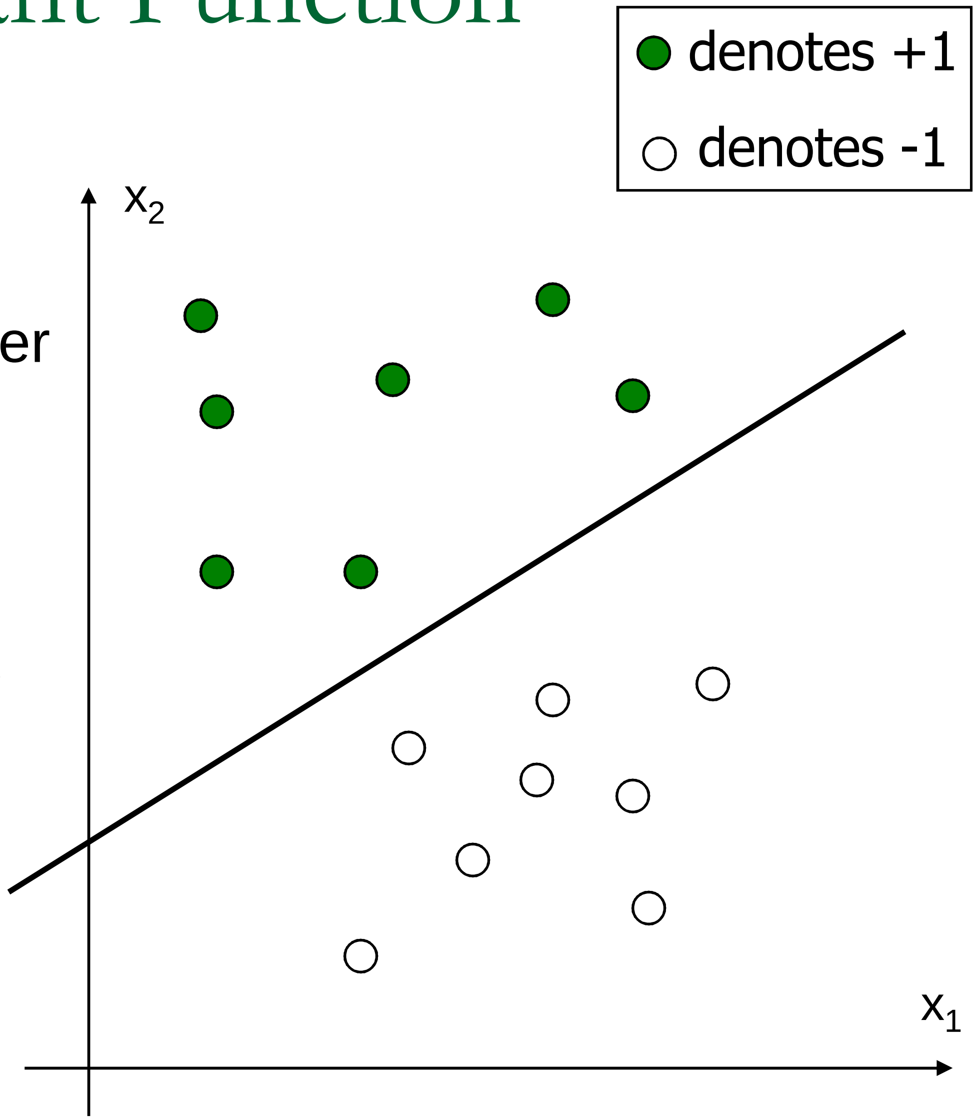
$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$



Linear Discriminant Function

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

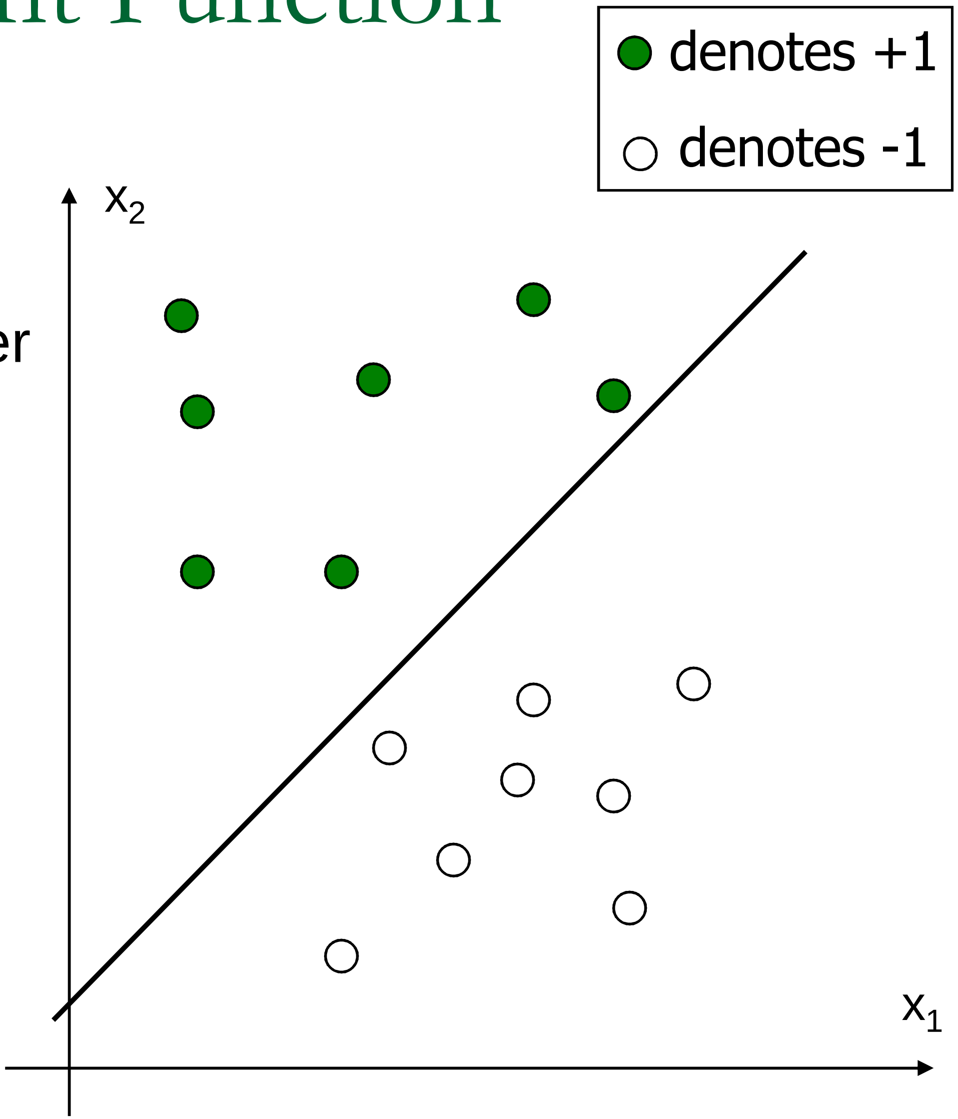
- Infinite number of answers!



Linear Discriminant Function

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

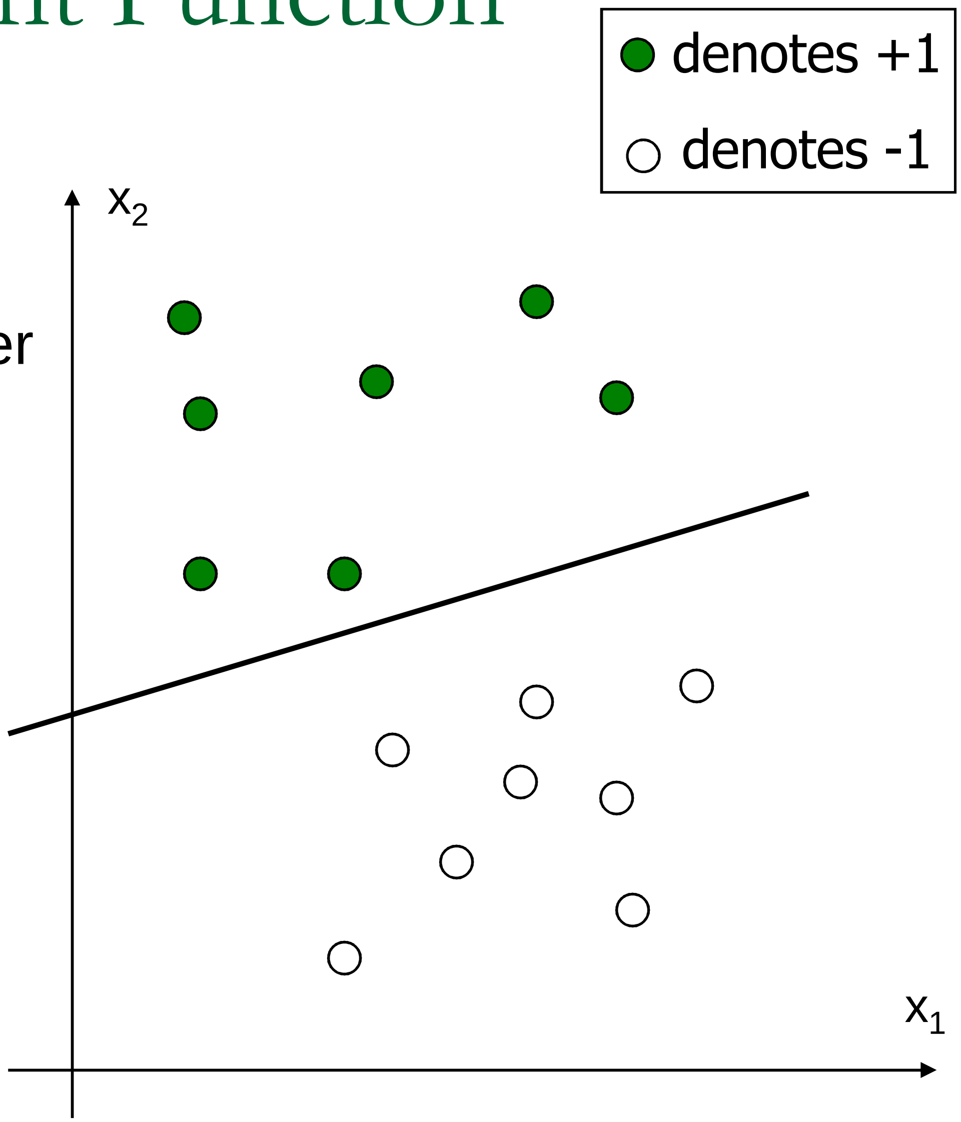
- Infinite number of answers!



Linear Discriminant Function

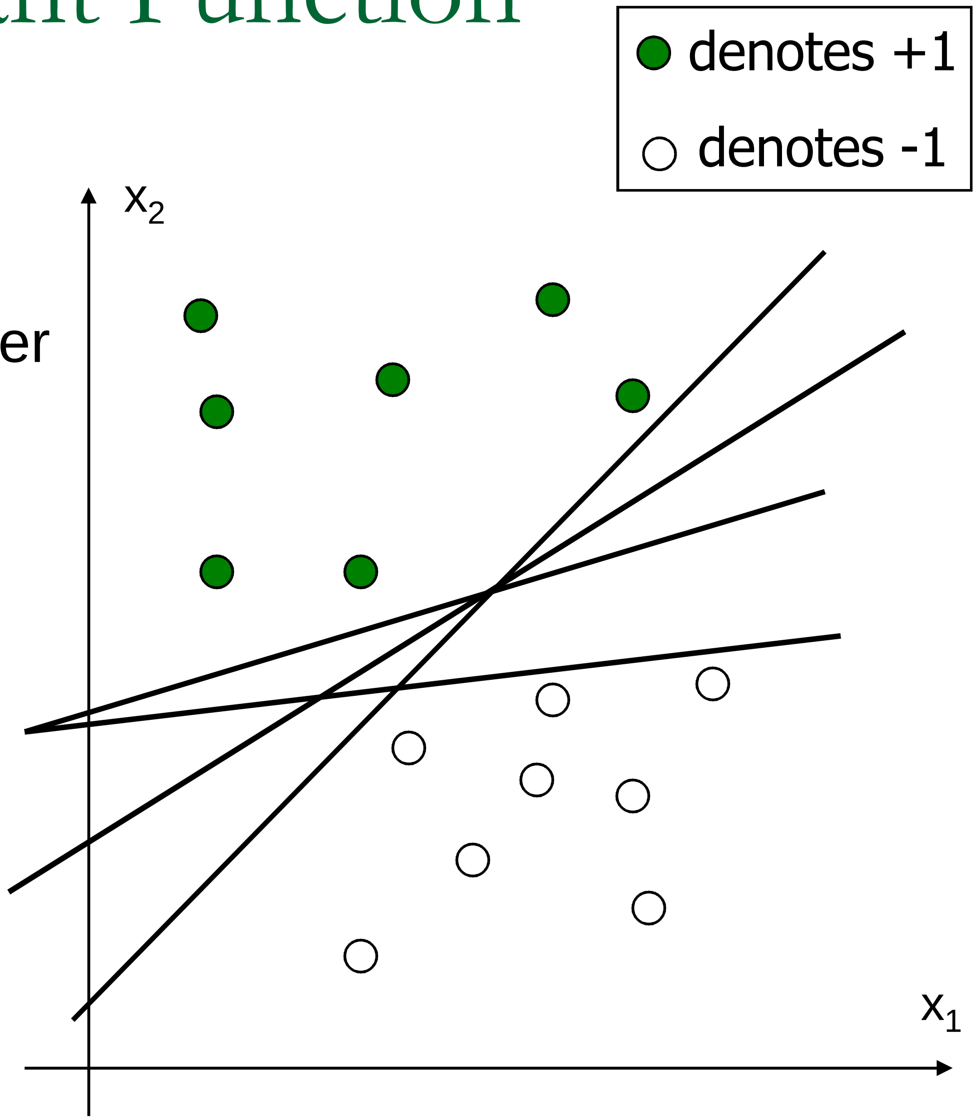
- How would you classify these points using a linear discriminant function in order to minimize the error rate?

- 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?

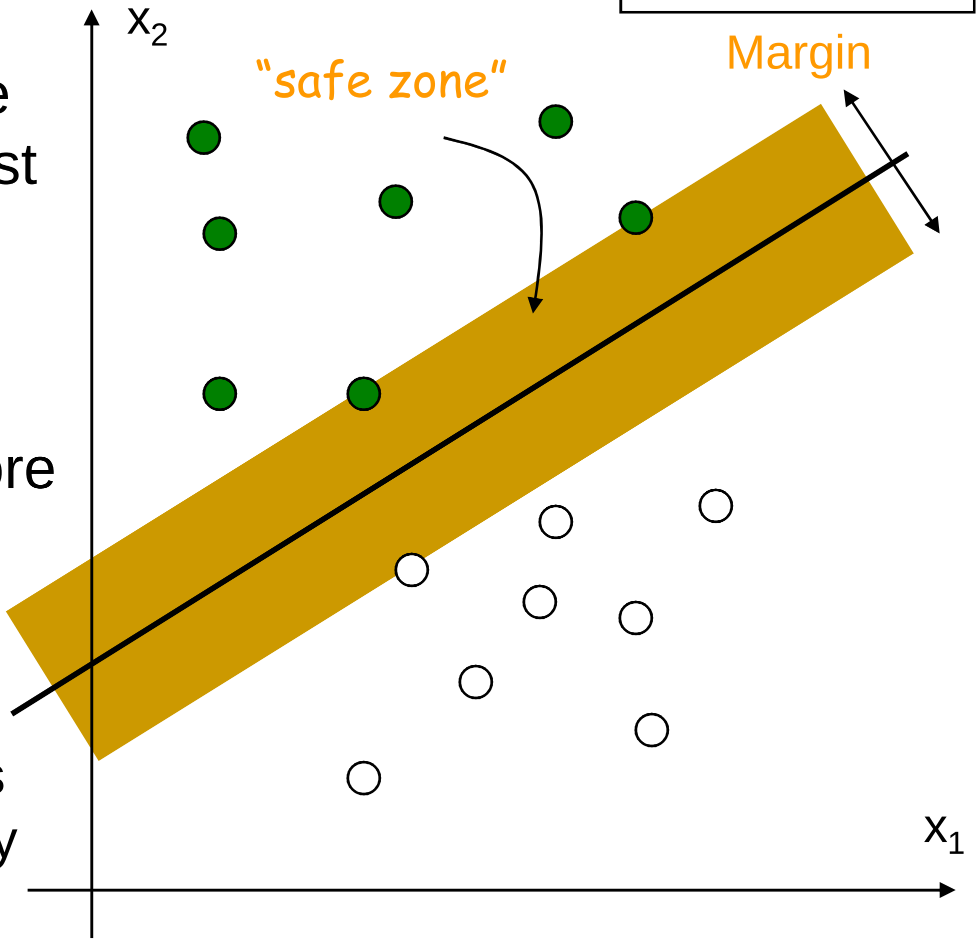


Large Margin Linear Classifier

● 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 outliers and thus strong generalization ability



Large Margin Linear Classifier

- Given a set of data points:

$\{(\mathbf{x}_i, y_i)\},\ i = 1, 2, \cdots, n,$ where

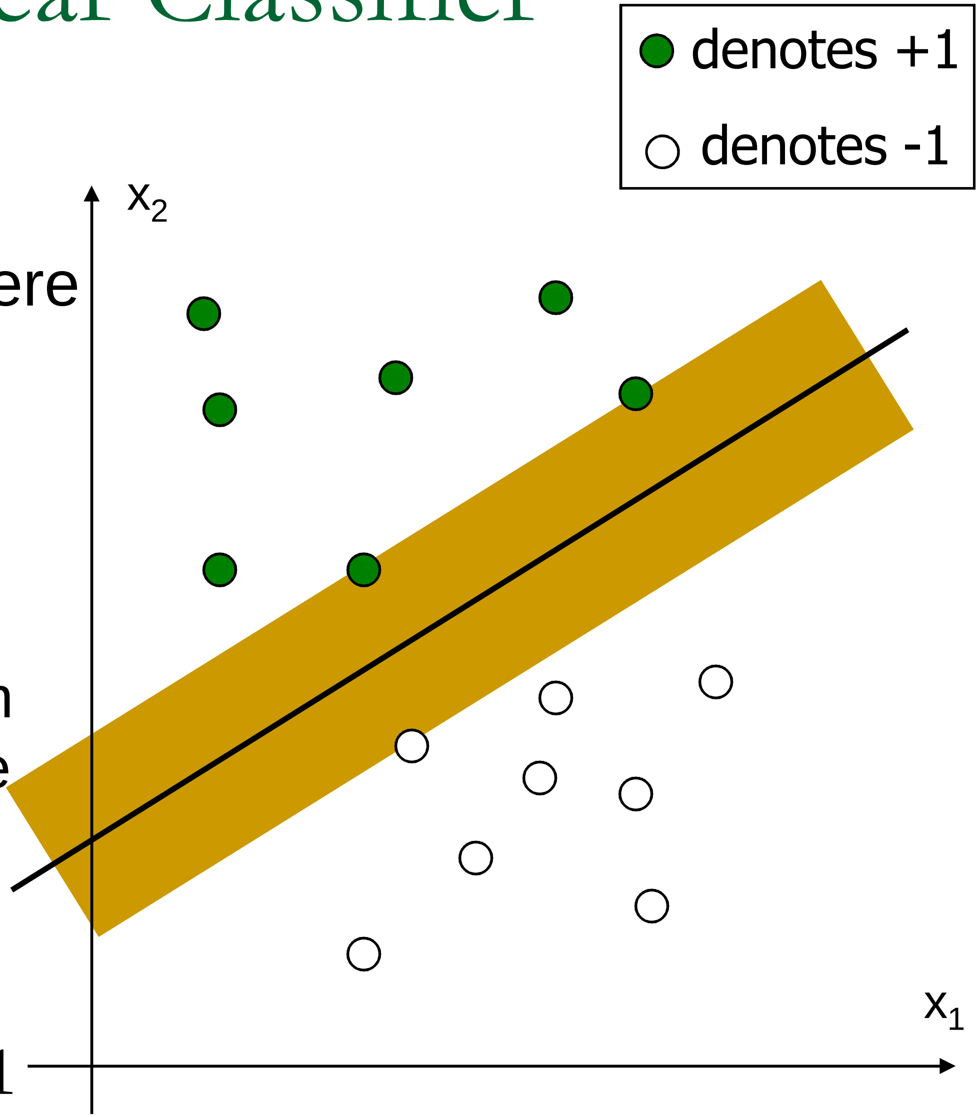
$$\text{For } y_i = +1,\quad \mathbf{w}^T\mathbf{x}_i + b > 0$$

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

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

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

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



Large Margin Linear Classifier

- We know that

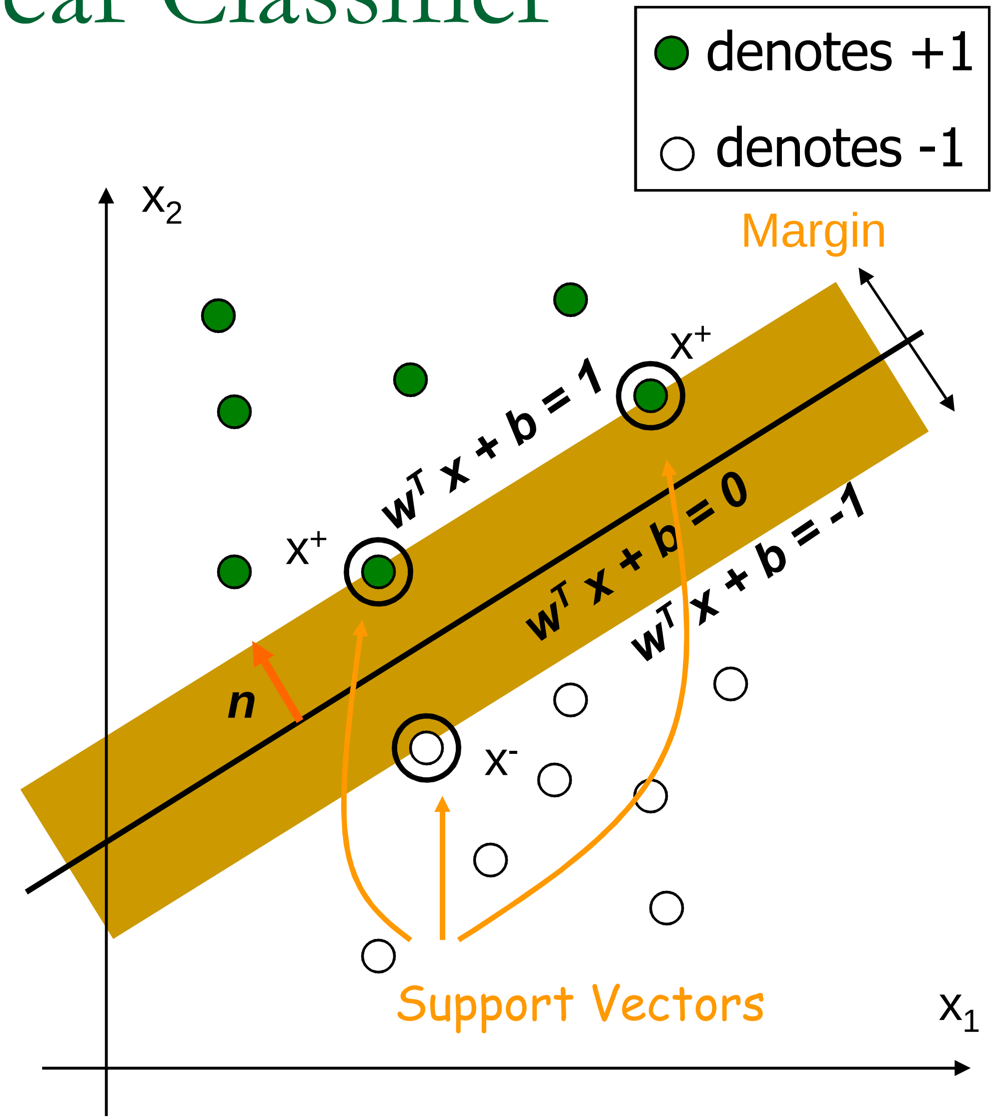
$$\mathbf{w}^T\mathbf{x}^+ + b = 1$$

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

- The margin width is:

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

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



Large Margin Linear Classifier

● denotes +1

○ denotes -1

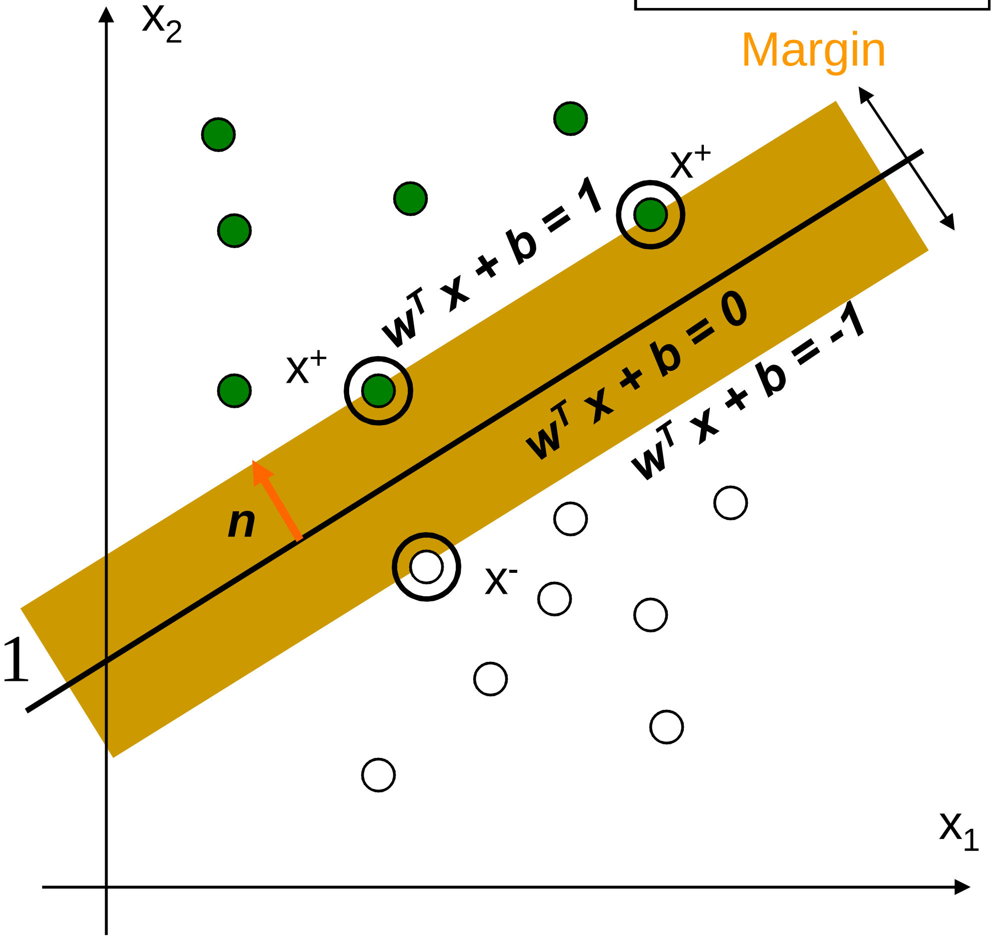
- Formulation:

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

such that

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

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



Large Margin Linear Classifier

- Formulation:

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

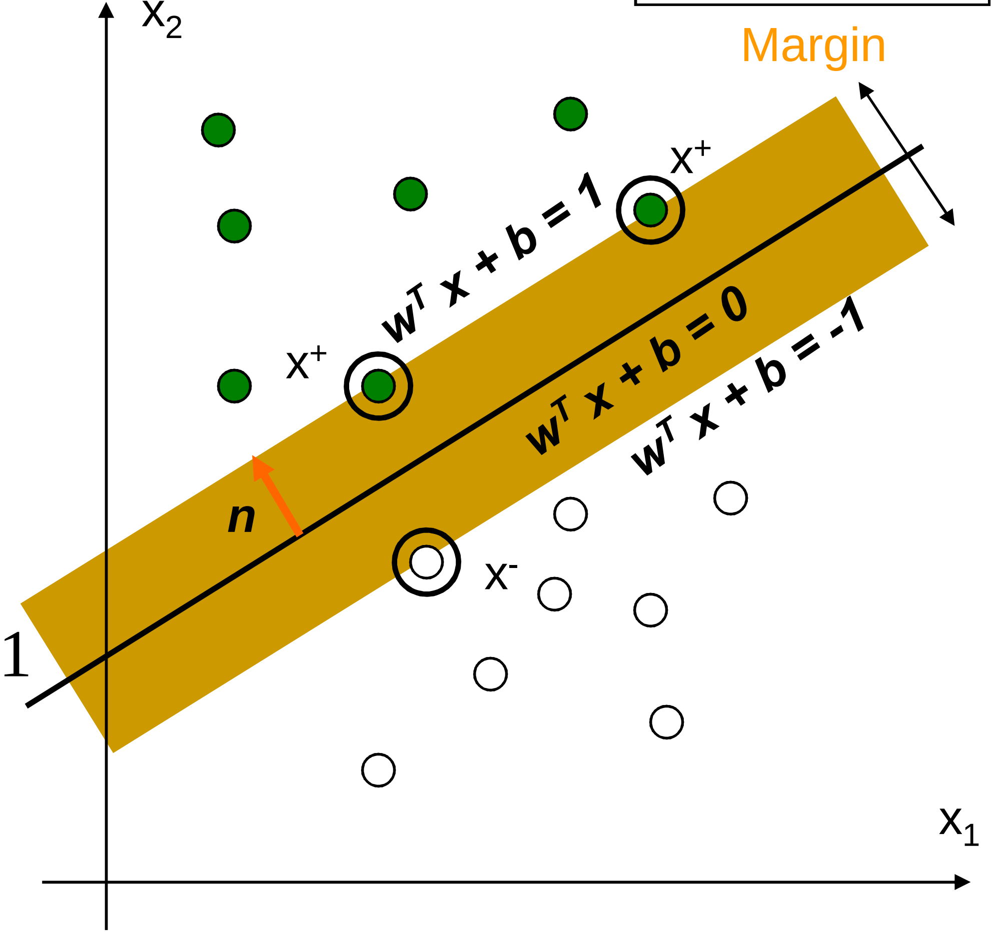
such that

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

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

● denotes +1

○ denotes -1



x₂
Margin
x⁺
x⁺
$\mathbf{w}^T \mathbf{x} + b = 1$
$\mathbf{w}^T \mathbf{x} + b = 0$
$\mathbf{w}^T \mathbf{x} + b = -1$
n
x⁻
x₁

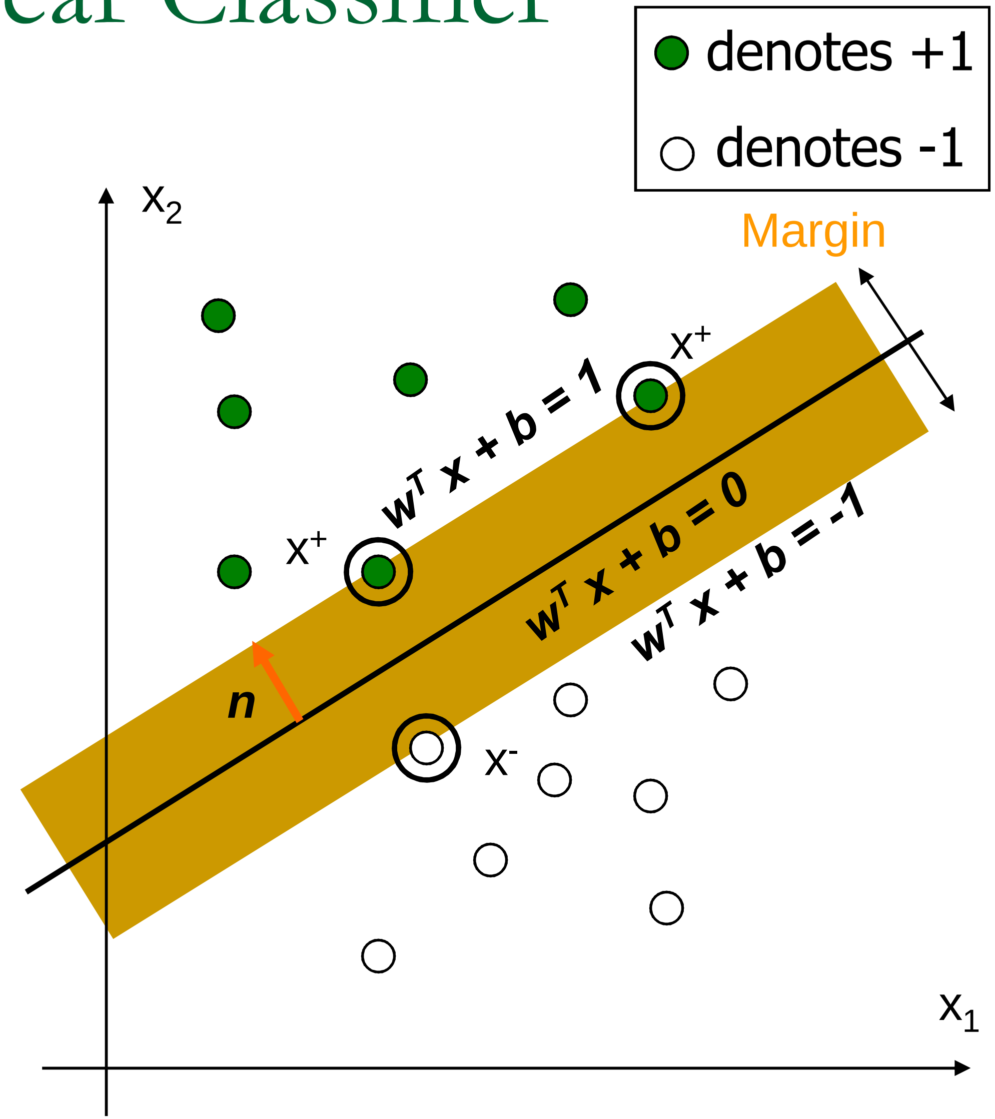
Large Margin Linear Classifier

- Formulation:

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

such that

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



SVM Primal Form

Objective function

$$\text{Minimize: } Q(w) = (w^T w) / 2 = ||w||^2 / 2$$

Constraints

$$y_i (w^T x_i + b) \geq +1, \text{ where } i = 1, 2, \ldots, N$$

When the feature space dimension and number of examples are low the primal problem can be solved without much problem. But, as will be discussed later, because we map the input space into a high-dimensional feature space, in some cases, with infinite dimensions, we convert primal form into the equivalent dual problem whose number of variables is the number of training data.

SVM Dual Form Formulation

Langragian primal form

Objective $$\max_{\alpha}\left(\min_{w,b}\left(L(w,b,\alpha)\right)\right)=\frac{1}{2}w^T\cdot w-\sum_{i=1}^{N}\alpha_i\left\{y_i\left(w^T\cdot x_i+b\right)-1\right\}$$

Constraints $$\alpha=\left[\alpha_1,\alpha_2,\cdots,\alpha_n\right], where \quad \alpha_i\geq 0$$

Langragian multipliers

The optimal solution is given by the saddle point, where is minimized with respect to w and *b* and maximized with respect to α_i (≥ 0), and it satisfies the following Karush-Kuhn-Tucker (KKT) conditions.

SVM Dual Form Formulation

Objective

$$\max_{\alpha}\left(\min_{w,b}\left(L(w,b,\alpha)\right)\right)=\frac{1}{2}w^{T}\cdot w-\sum_{i=1}^{N}\alpha_{i}\left\{y_{i}\left(w^{T}\cdot x_{i}+b\right)-\right.$$

Constraints

$$\alpha=\left[\alpha_{1},\alpha_{2},\cdots,\alpha_{n}\right], where \quad \alpha_{i}\geq 0$$

KKT Conditions

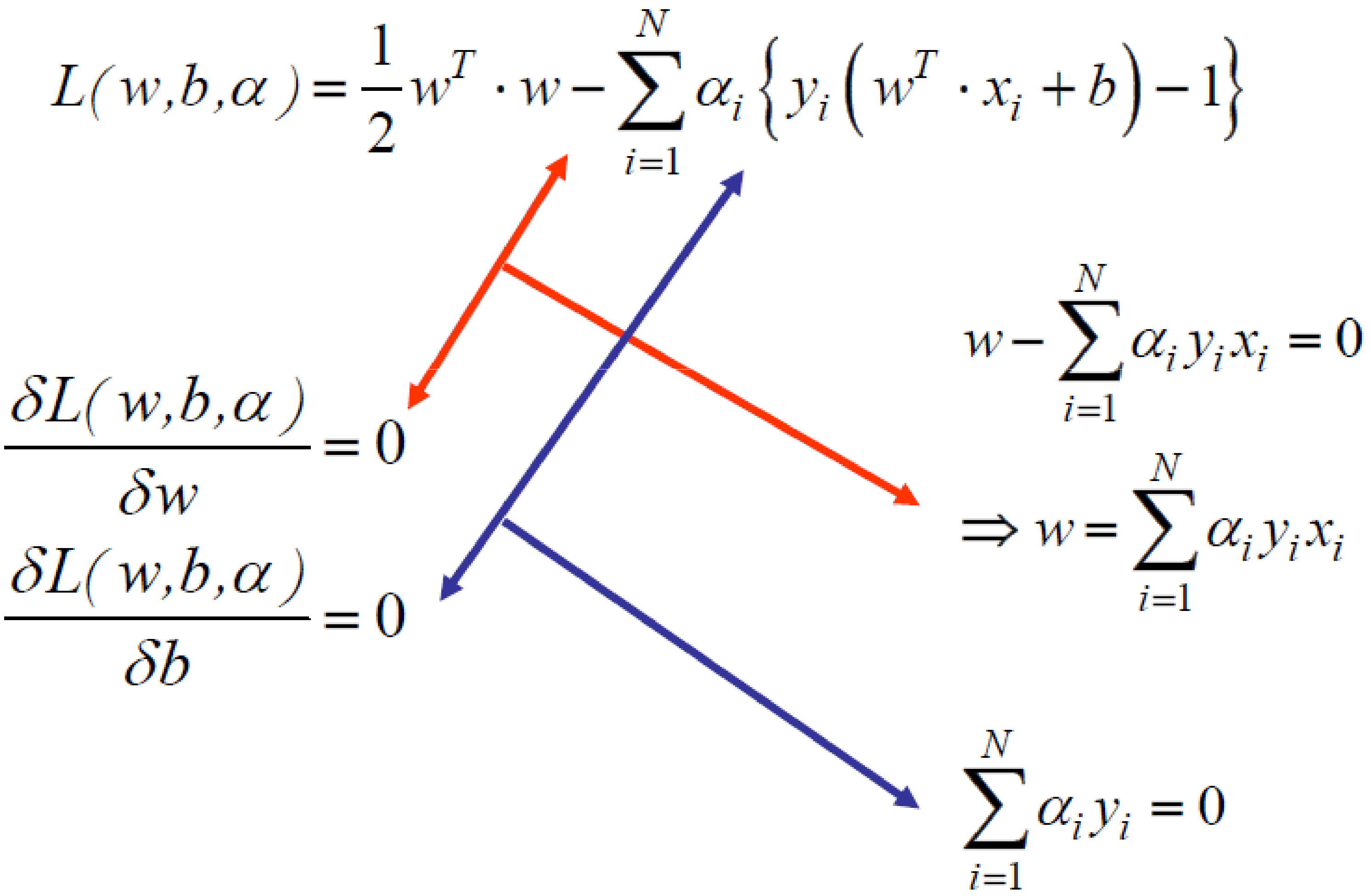
$$\frac{\delta L(w,b,\alpha)}{\delta w}=0$$

$$\frac{\delta L(w,b,\alpha)}{\delta b}=0$$

$$\alpha_{i}\left\{y_{i}\left(w^{T}\cdot x_{i}\right)-1\right\}=0, \qquad i=1,2,\ldots,N.$$

KKT Complementary conditions

SVM Dual Form Formulation



$$L(w,b,\alpha)=\frac{1}{2}w^T\cdot w-\sum_{i=1}^{N}\alpha_i\left\{y_i\left(w^T\cdot x_i+b\right)-1\right\}$$

$$\frac{\delta L(w,b,\alpha)}{\delta w}=0$$

$$\frac{\delta L(w,b,\alpha)}{\delta b}=0$$

$$w-\sum_{i=1}^{N}\alpha_i y_i x_i=0$$

$$\Rightarrow w=\sum_{i=1}^{N}\alpha_i y_i x_i$$

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

SVM Dual Form Formulation

$$L(w,b,\alpha) = \frac{1}{2} w^T \cdot w - \sum_{i=1}^{N} \alpha_i \left\{ y_i \left(w^T \cdot x_i + b \right) - 1 \right\}$$

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

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

$$L(w,b,\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T \cdot x_i$$

SVM Dual Form

Objective

$$\underset{\alpha}{Maximize}\left(L(w,b,\alpha)\right)=\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^T\cdot x_i$$

$$=\sum_{i\in SV}\alpha_i-\frac{1}{2}\sum_{i,j\in SV}\alpha_i\alpha_j y_i y_j x_i^T\cdot x_i$$

Constraints

$$\sum_{i=1}^{N}\alpha_i y_i=0,\qquad \alpha_i\geq 0$$

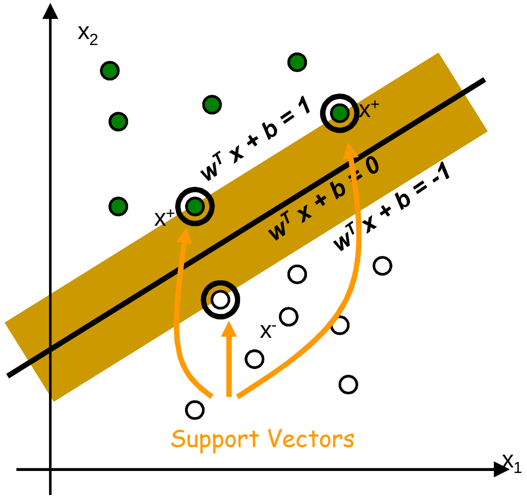
Hard Margin Support Vector Machine

We have assumed that the data are linearly separable

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:

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

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:

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

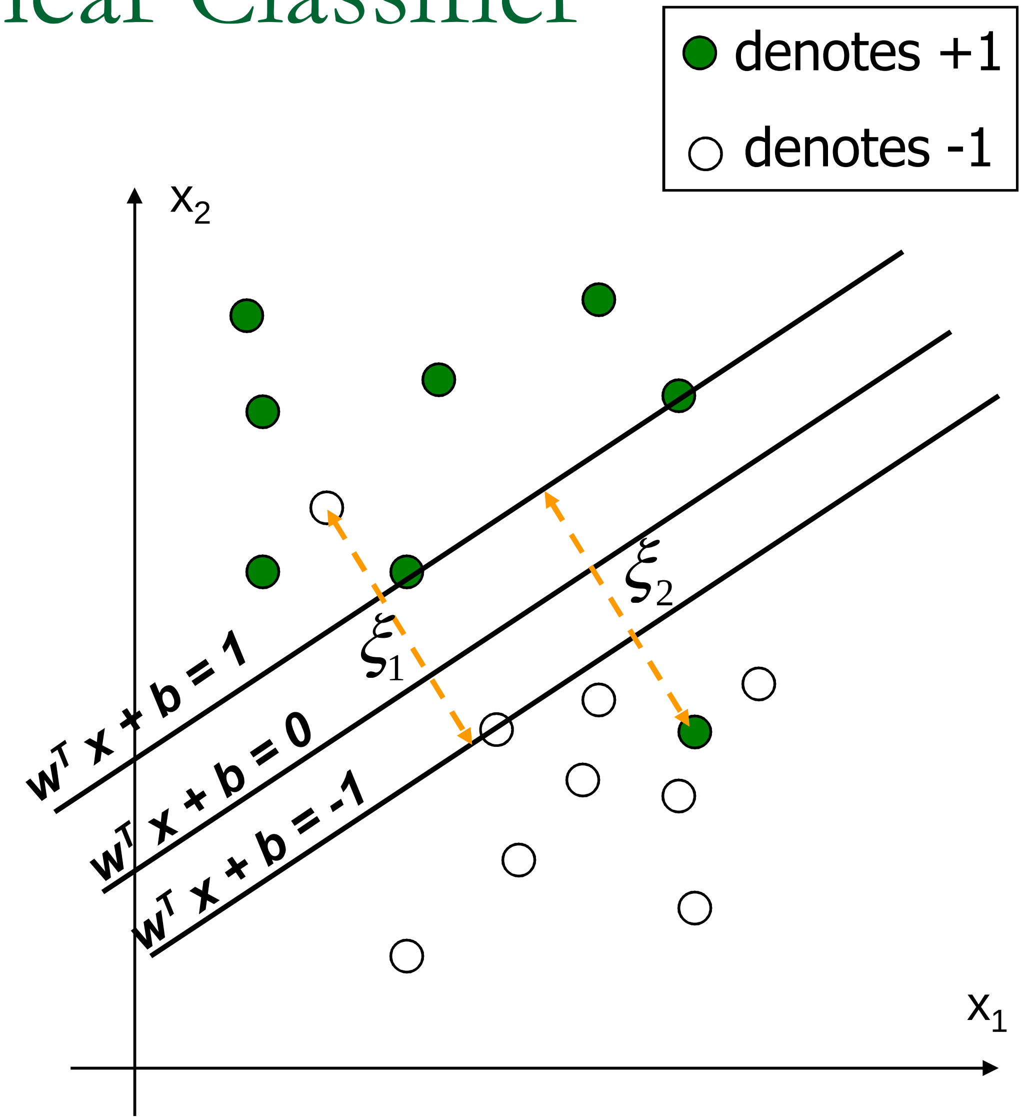
- Notice it relies on a *dot product* between the test point ***x*** and the support vectors $\boldsymbol{x}_i$

- Also keep in mind that solving the optimization problem involved computing the dot products $\boldsymbol{x}_i^T\boldsymbol{x}_j$ 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



Large Margin Linear Classifier

- Formulation:

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

such that

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

$$\xi_i \geq 0$$

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

Large Margin Linear Classifier

- Formulation: (Lagrangian Dual Problem)

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

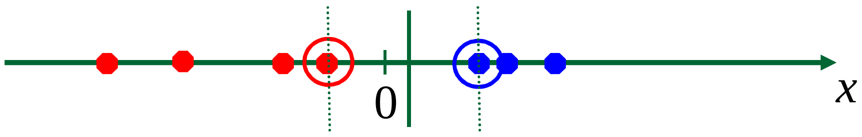
such that

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

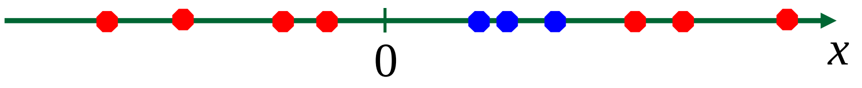
$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

Non-linear SVMs

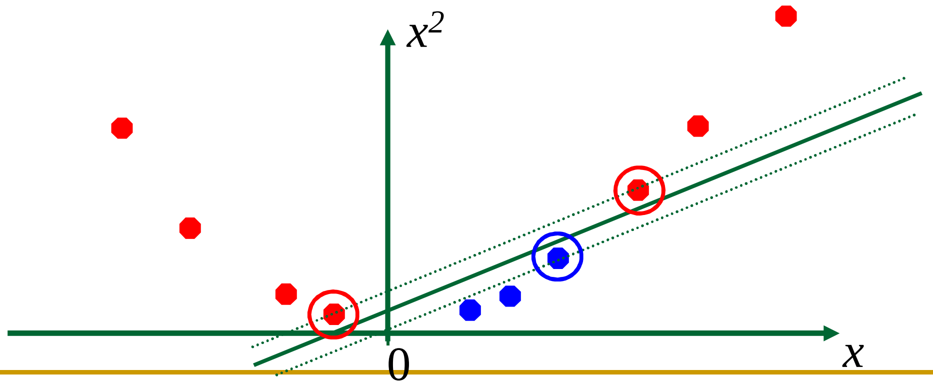
- Datasets that are linearly separable with noise work out great:



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



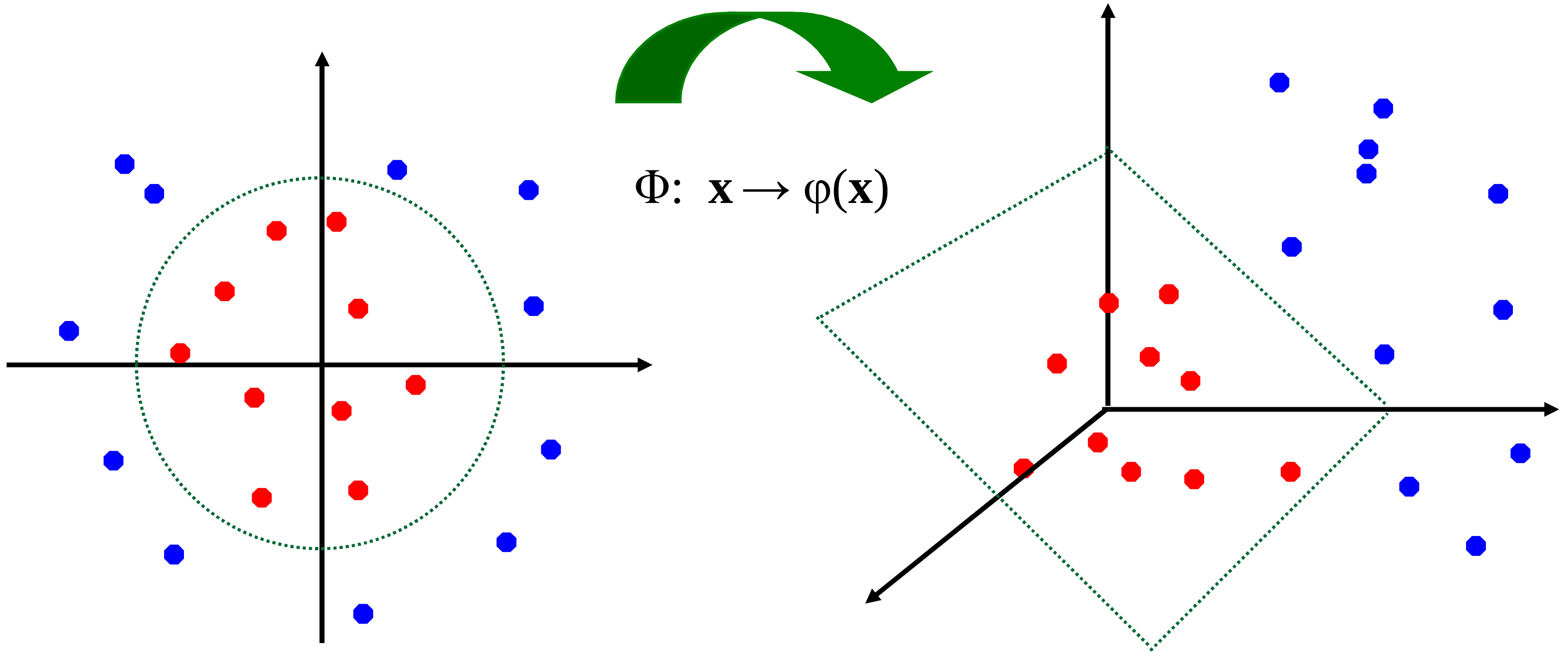
- How about… mapping data to a higher-dimensional space:



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:



$\Phi\colon\ \mathbf{x} \rightarrow \varphi(\mathbf{x})$

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

Nonlinear SVMs: The Kernel Trick

- With this mapping, our discriminant function is now:

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

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

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

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)

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

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***

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.