Support Vector Machines

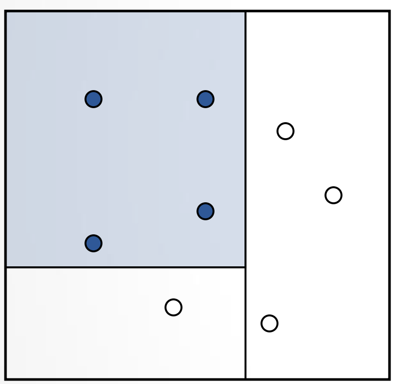
Vibhav Gogate
The University of Texas at dallas

What We have Learned So Far?

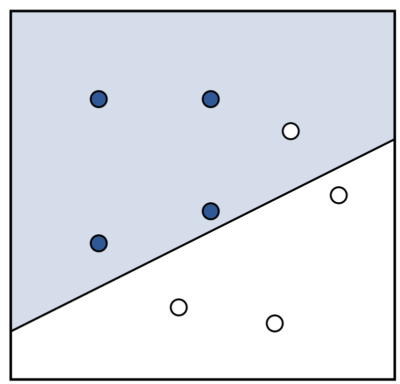
1. Decision Trees
2. Naïve Bayes
3. Linear Regression
4. Logistic Regression
5. Perceptron
6. Neural networks
7. K-Nearest Neighbors

- **Which of the above are linear and which are not?**
- **(1) (6) and (7) are non-linear**
    - **(2) is linear under certain restrictions**

Decision Surfaces

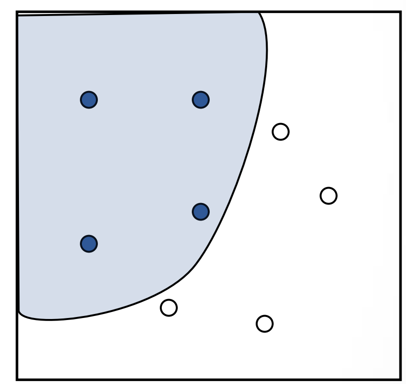


Decision Tree



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

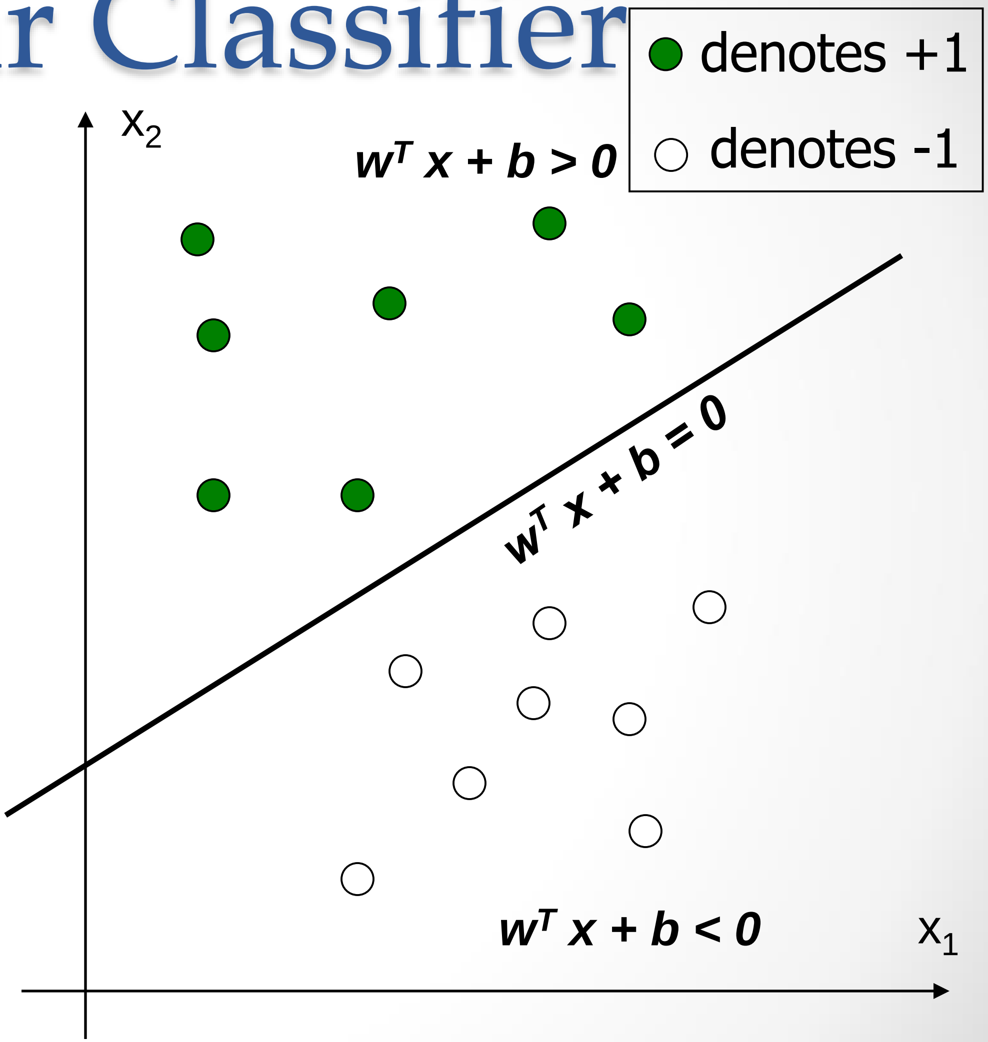
- Linear Discriminant Function
- Large Margin Linear Classifier
- Nonlinear SVM: The Kernel Trick
- SVM for Regression
- SMO Algorithm

Linear Discriminant Function or a Linear Classifier

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

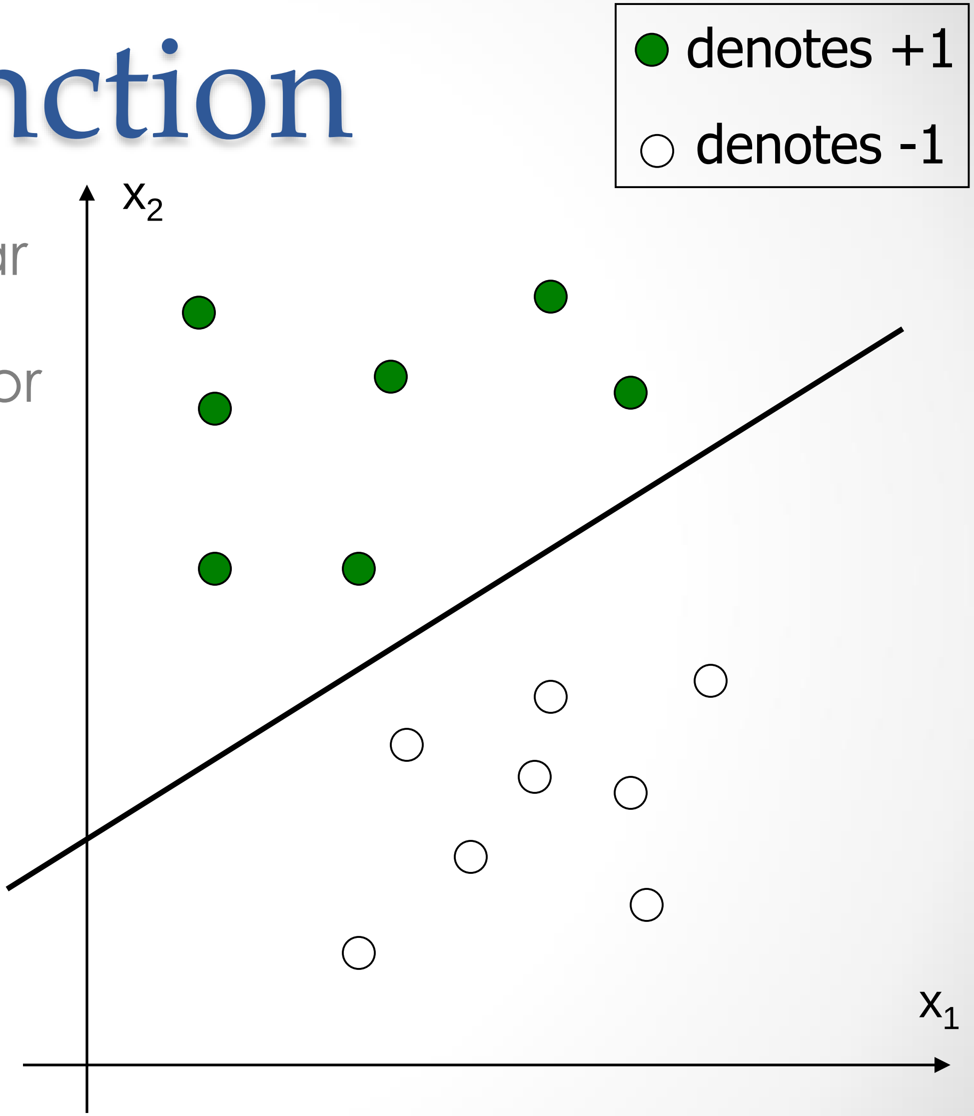
$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

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



Linear Discriminant Function

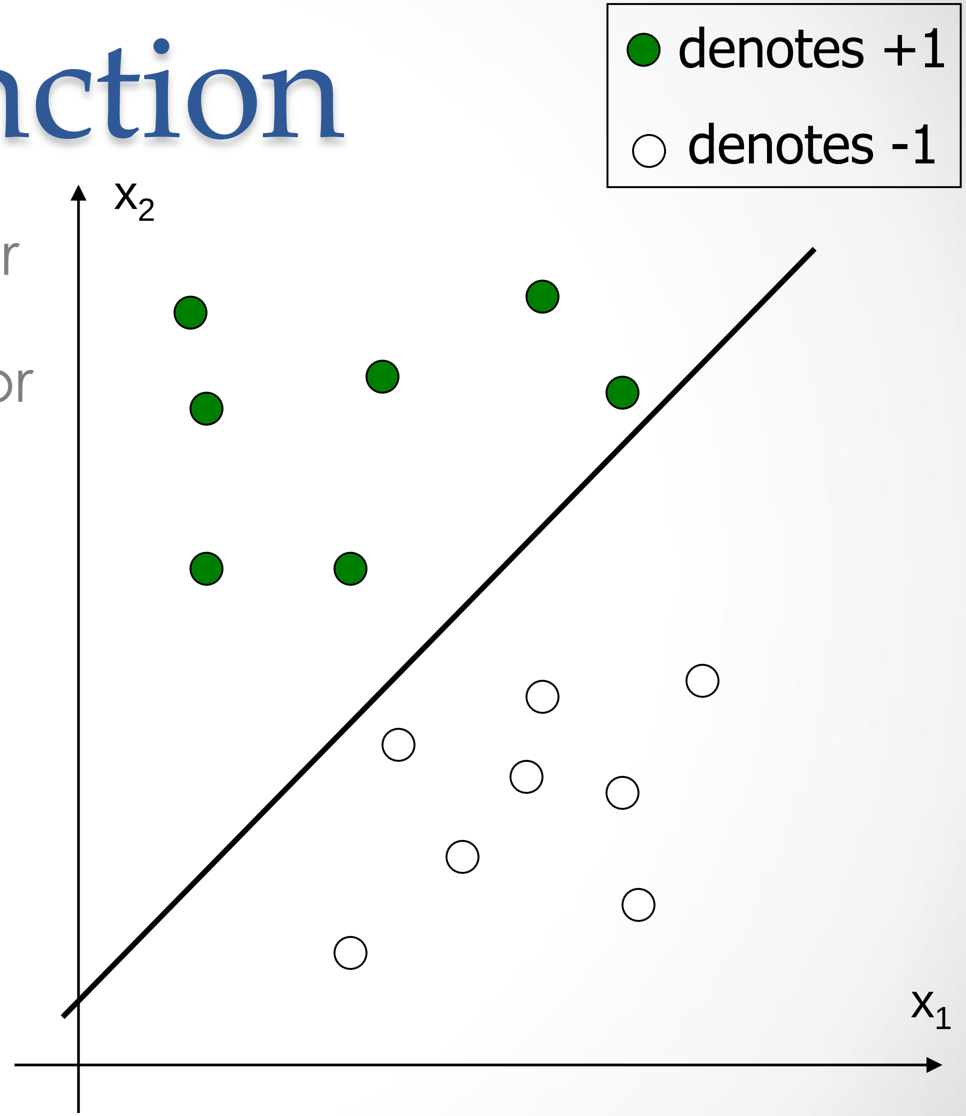
- How would you classify these points using a linear discriminant function in order to minimize the error rate?



- 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?

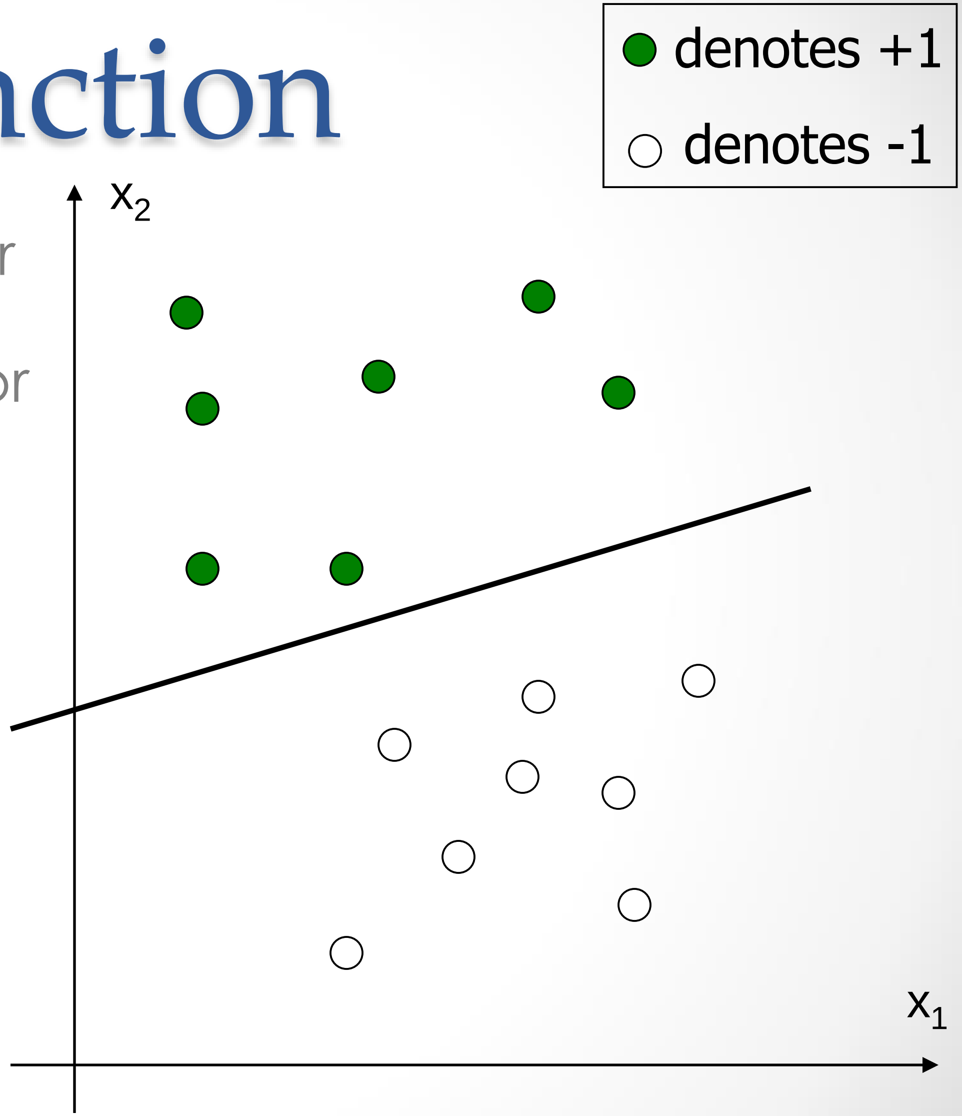


- Infinite number of answers!

Linear Discriminant Function

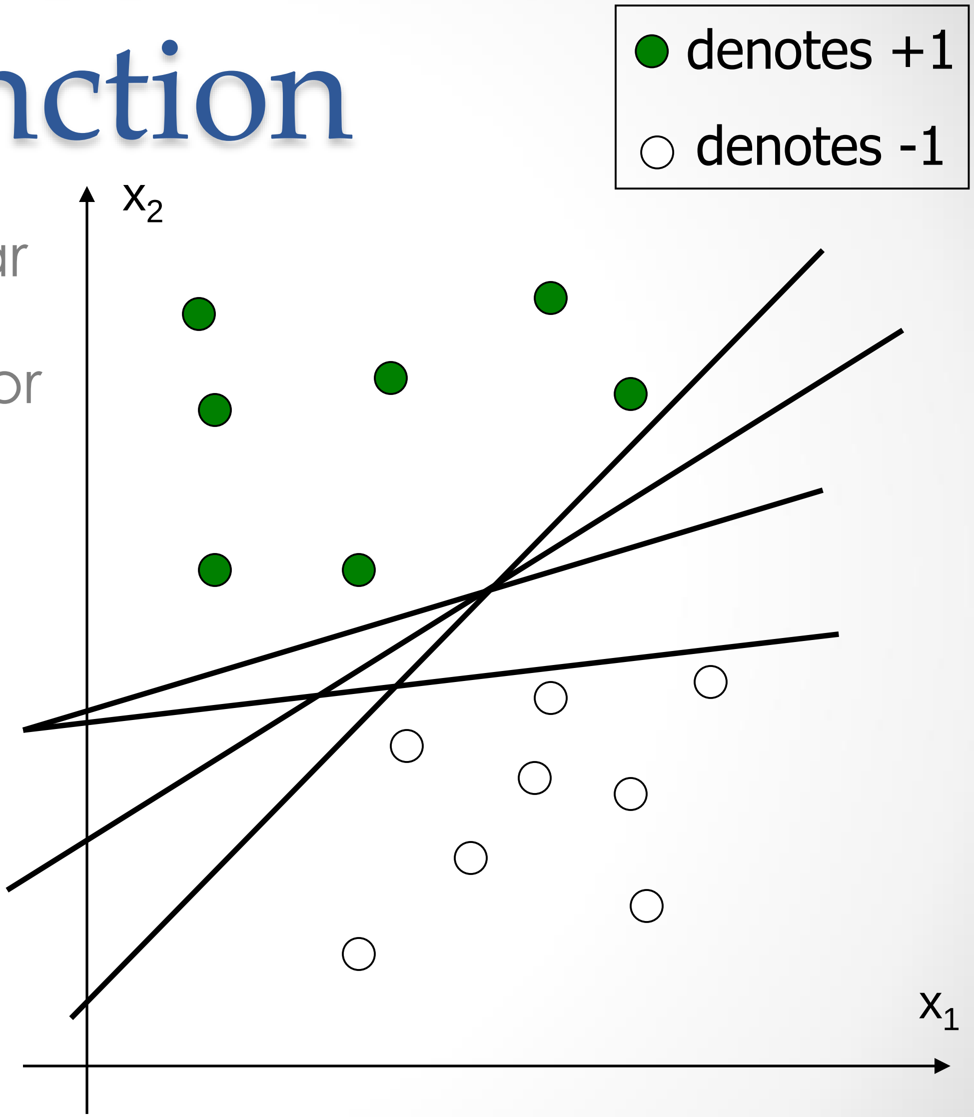
- How would you classify these points using a linear discriminant function in order to minimize the error rate?

- Infinite number of answers!



Linear Discriminant Function

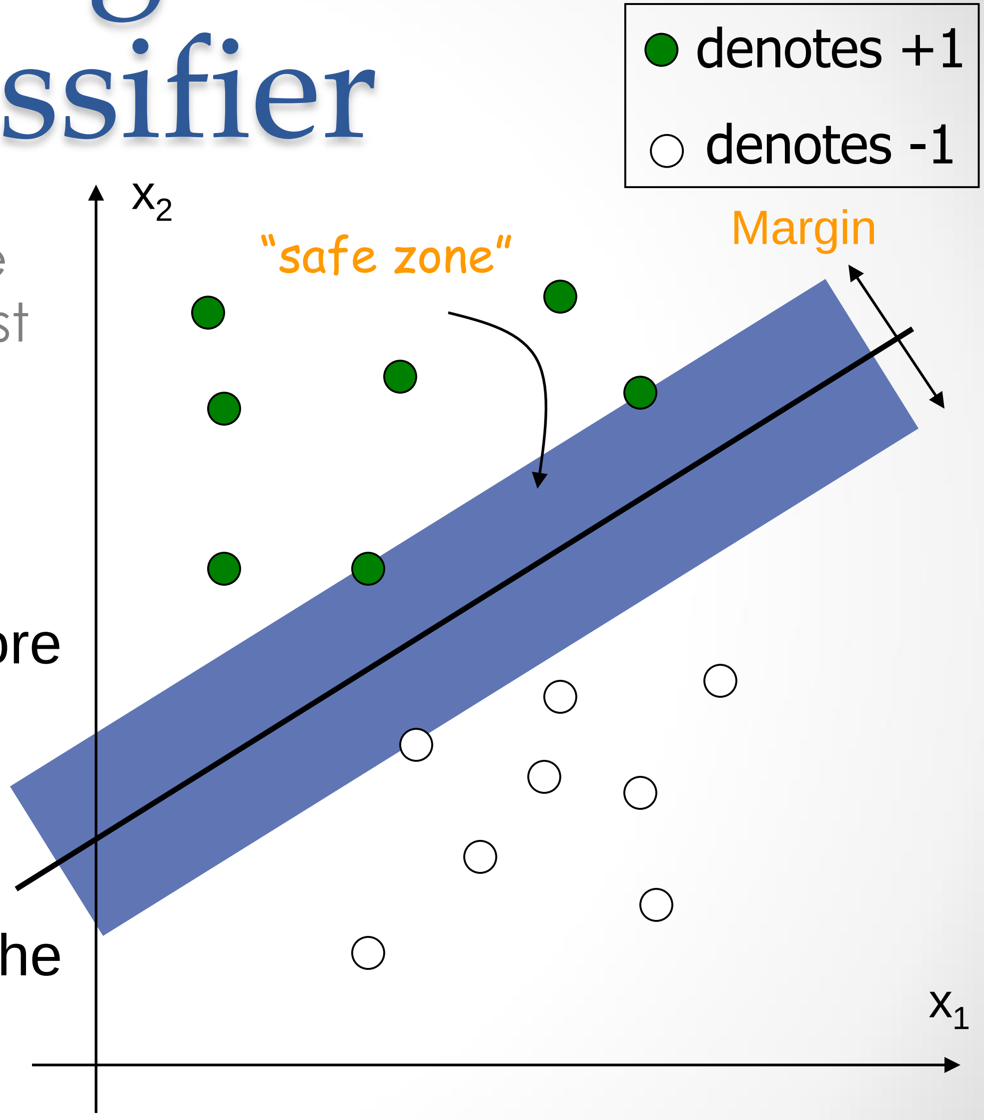
- How would you classify these points using a linear discriminant function in order to minimize the error rate?



- 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

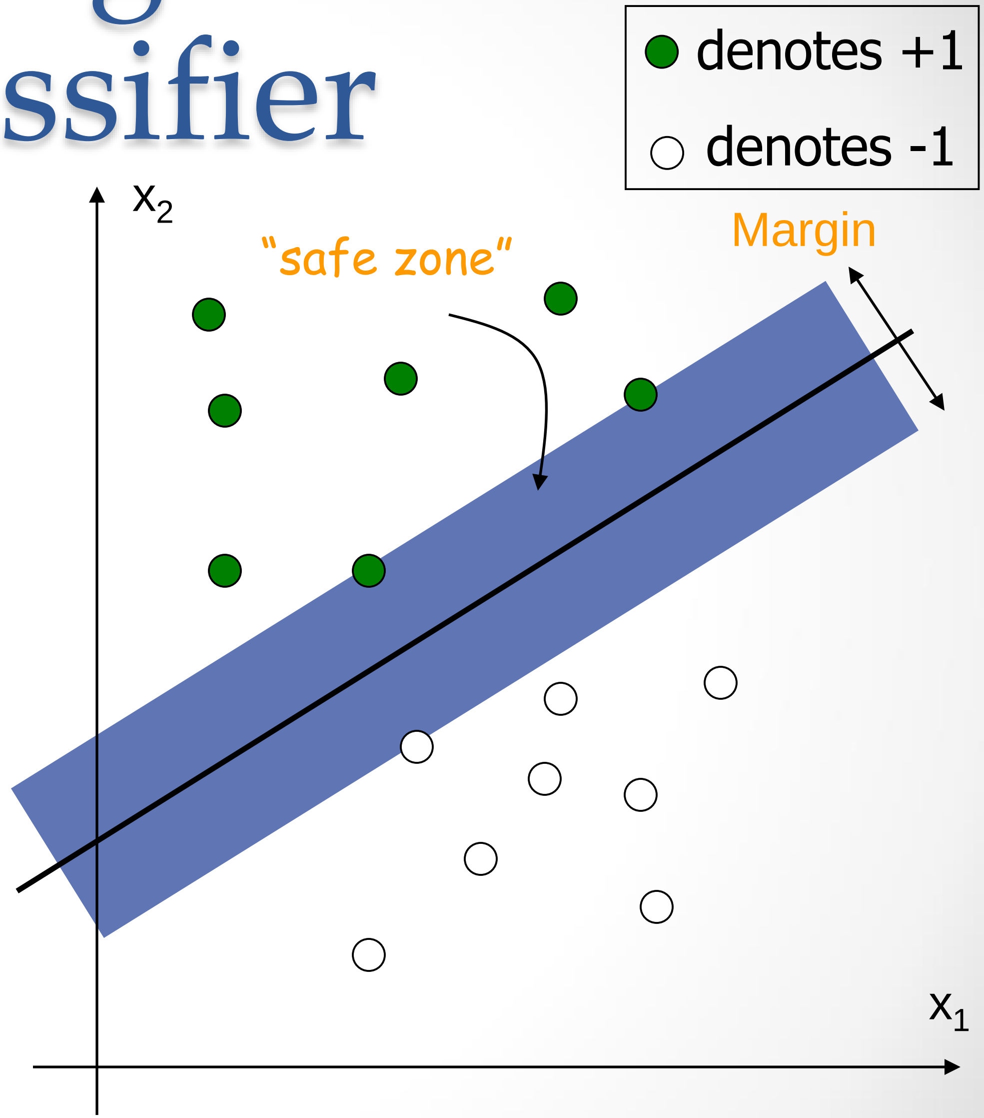
Large Margin Linear Classifier

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

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

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

- 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

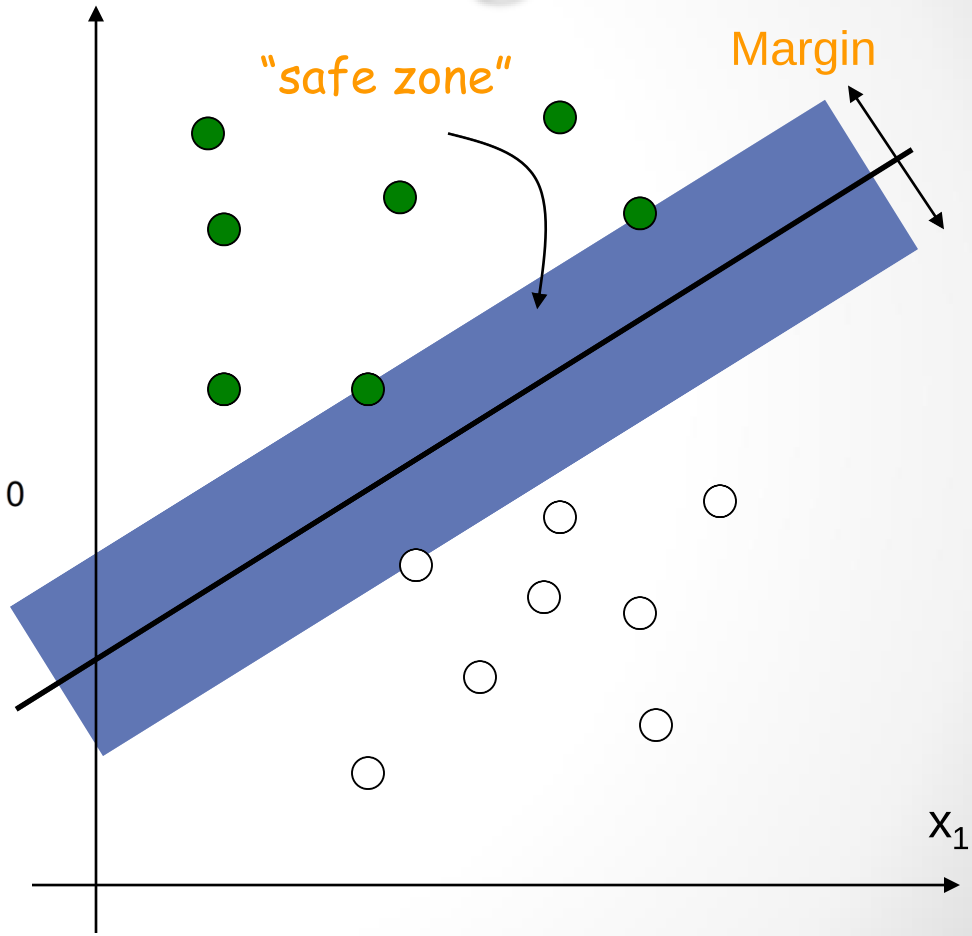
$$ax + by + c_2 = 0$$

the distance between them is given by:

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

Our lines in 2-D are:
$w_1 x_1 + w_2 x_2 + b - 1 = 0$ and $w_1 x_1 + w_2 x_2 + b + 1 = 0$

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



Large Margin Linear Classifier

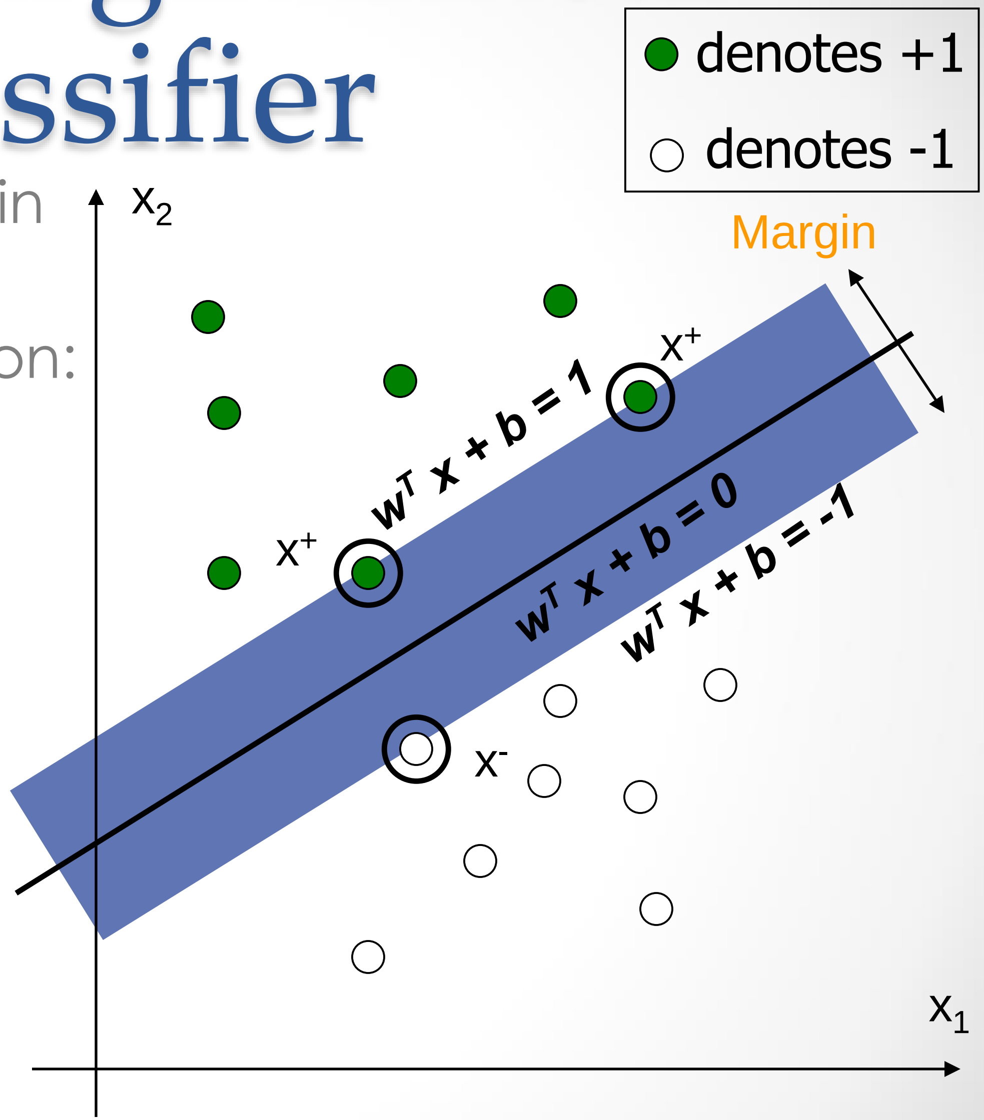
- Aim: Learn a large margin classifier
- Mathematical Formulation:

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

such that

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

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



Common theme in machine learning:
LEARNING IS OPTIMIZATION

Large Margin Linear Classifier

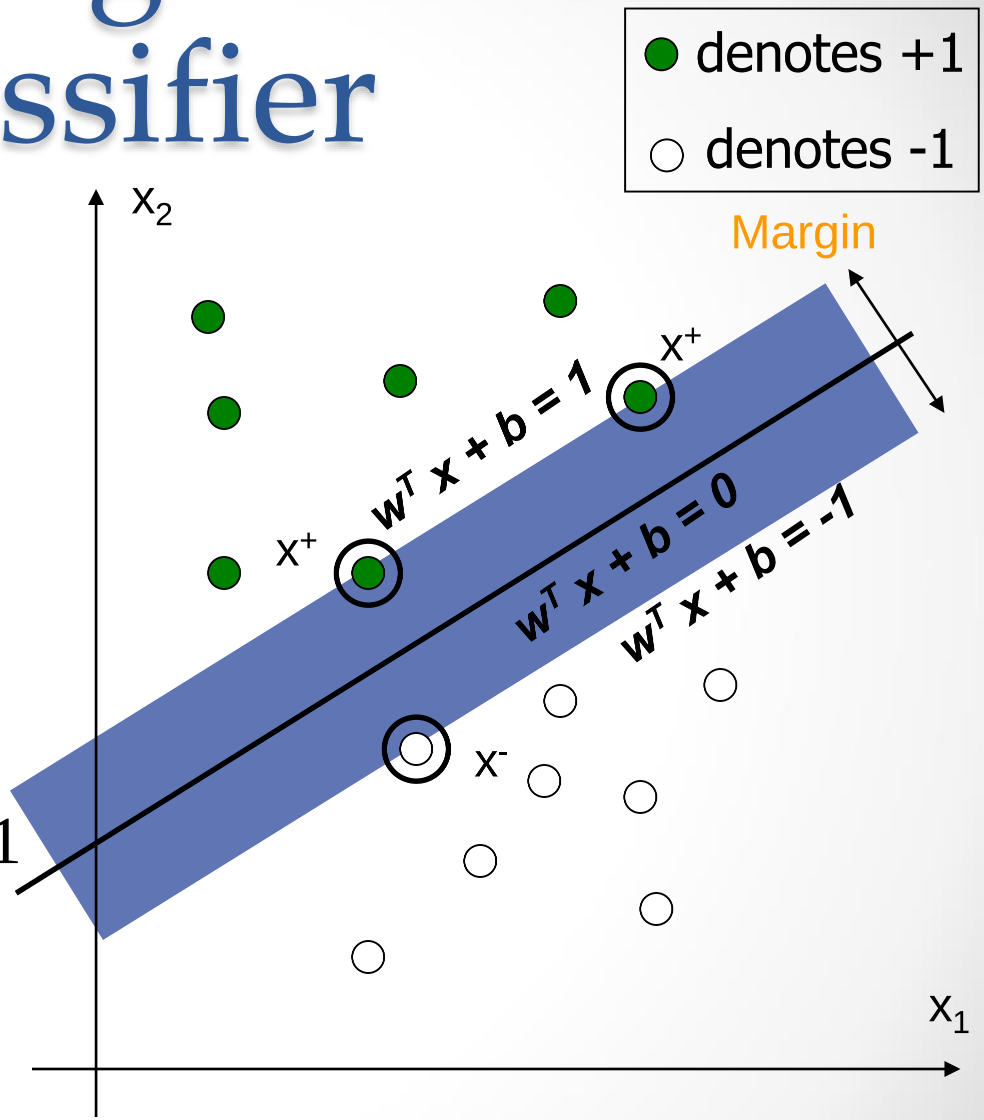
- Formulation:

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

such that

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

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



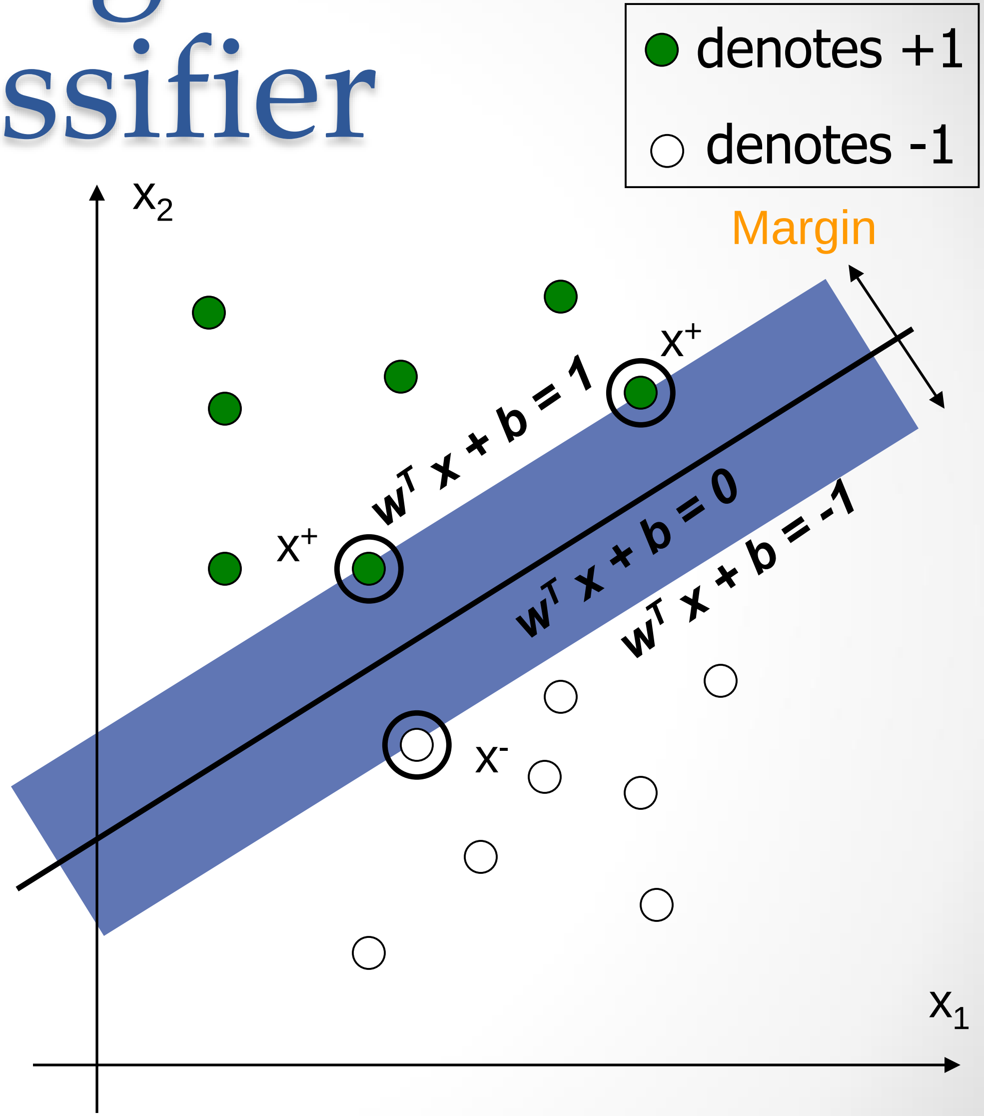
Large Margin Linear Classifier

- Formulation:

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

such that

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



Large Margin Linear Classifier

- Formulation:

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

such that

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

- This is a Quadratic programming problem with linear constraints
  - Off-the-shelf Software
- However, we will convert it to Lagrangian dual in order **to use the kernel trick!**

Lagrangian Duality

- The Lagrangian method is a general method for converting constrained optimization problems to unconstrained ones.
  - unconstrained optimization problems are easier
- When we have equality constraints, the transformation looks as follows

$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & h_i(w) = 0, \quad i = 1, \ldots, l \end{aligned} \quad \Rightarrow \quad \mathcal{L}(w, \beta) = f(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

set $\mathcal{L}$'s partial derivatives to zero: $\dfrac{\partial \mathcal{L}}{\partial w_i} = 0; \quad \dfrac{\partial \mathcal{L}}{\partial \beta_i} = 0,$

solve for w and β.

Lagrangian Duality

$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & g_i(w) \le 0, \quad i = 1, \ldots, k \\ & h_i(w) = 0, \quad i = 1, \ldots, l. \end{aligned}$$

Lagrangian function

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

Same as solving the primal problem

Not the same as solving the following dual problem

$$\min_w \max_{\alpha, \beta \,:\, \alpha_i \ge 0} \mathcal{L}(w, \alpha, \beta)$$

Let the optimal solution be

$$p^* = \min_w \max_{\alpha, \beta \,:\, \alpha_i \ge 0} \mathcal{L}(w, \alpha, \beta)$$

$$\max_{\alpha, \beta \,:\, \alpha_i \ge 0} \min_w \mathcal{L}(w, \alpha, \beta)$$

Let the optimal solution be

$$d^* = \max_{\alpha, \beta \,:\, \alpha_i \ge 0} \min_w \mathcal{L}(w, \alpha, \beta)$$

$$p^* \ge d^*$$

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 Function

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

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

Solving the Optimization Problem

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

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

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

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

Solving the Optimization Problem

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

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

Lagrangian Dual Problem

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

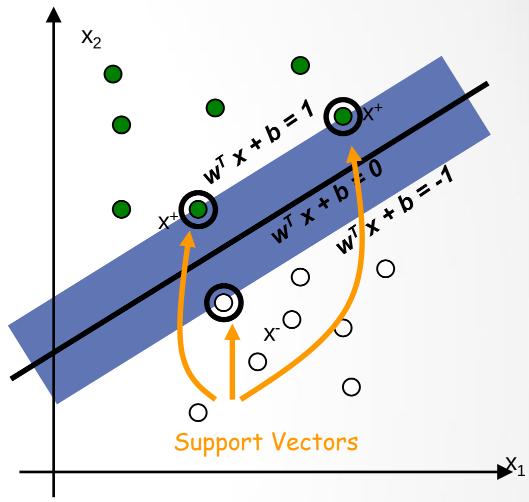
$$\text{s.t.} \quad \alpha_i \geq 0 \text{ , and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

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:

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

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:

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

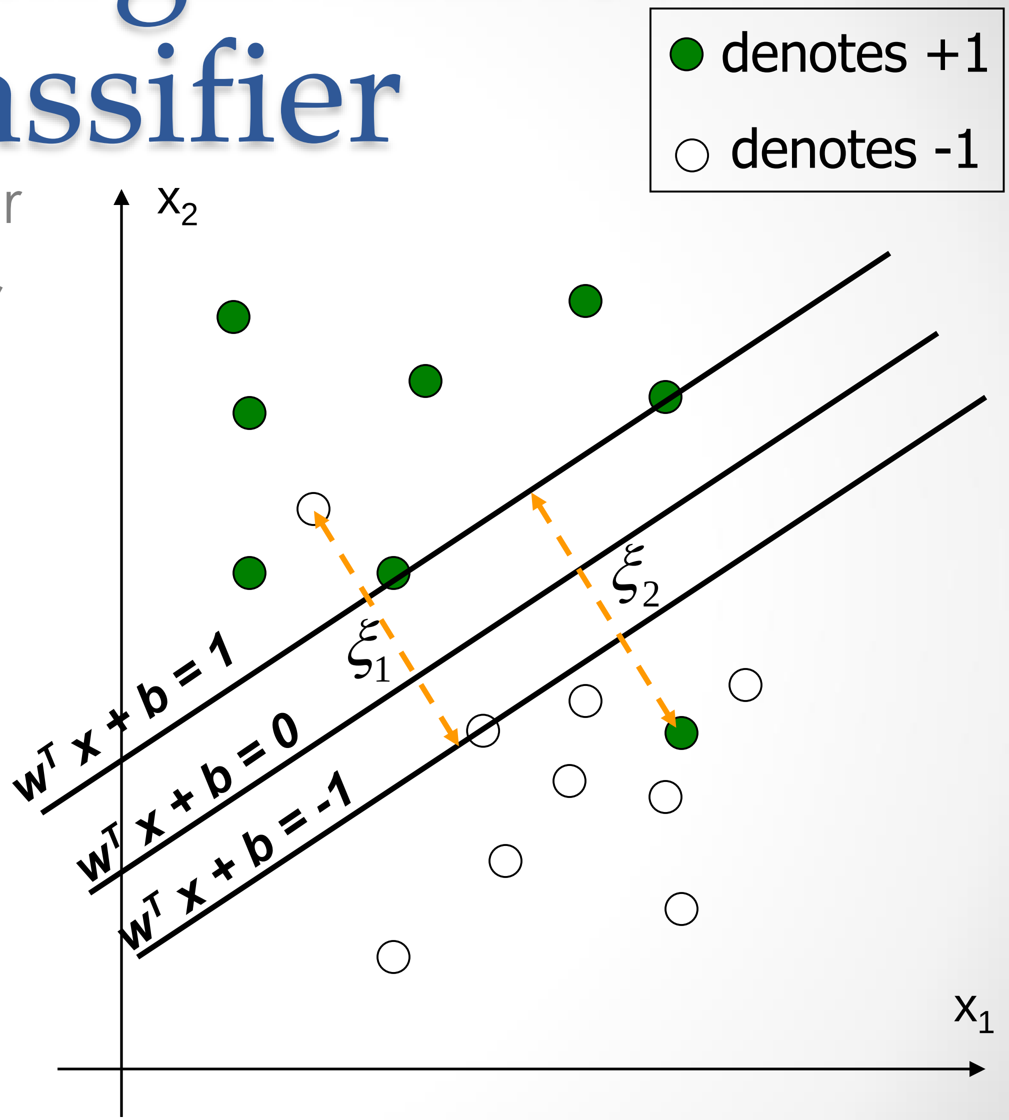
- Notice it relies on a *dot product* between the test point ***x*** and the support vectors $\boldsymbol{x}_i$

- 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

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

$$\xi_i \geq 0$$

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

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

Without slack variables

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

Large Margin Linear Classifier

- Formulation: (Lagrangian Dual Problem)

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

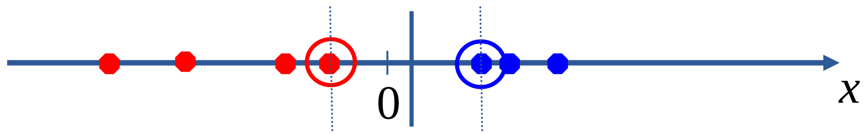
such that

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

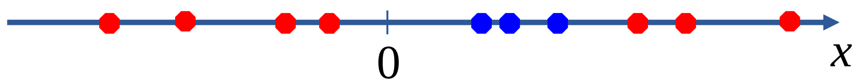
$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

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?



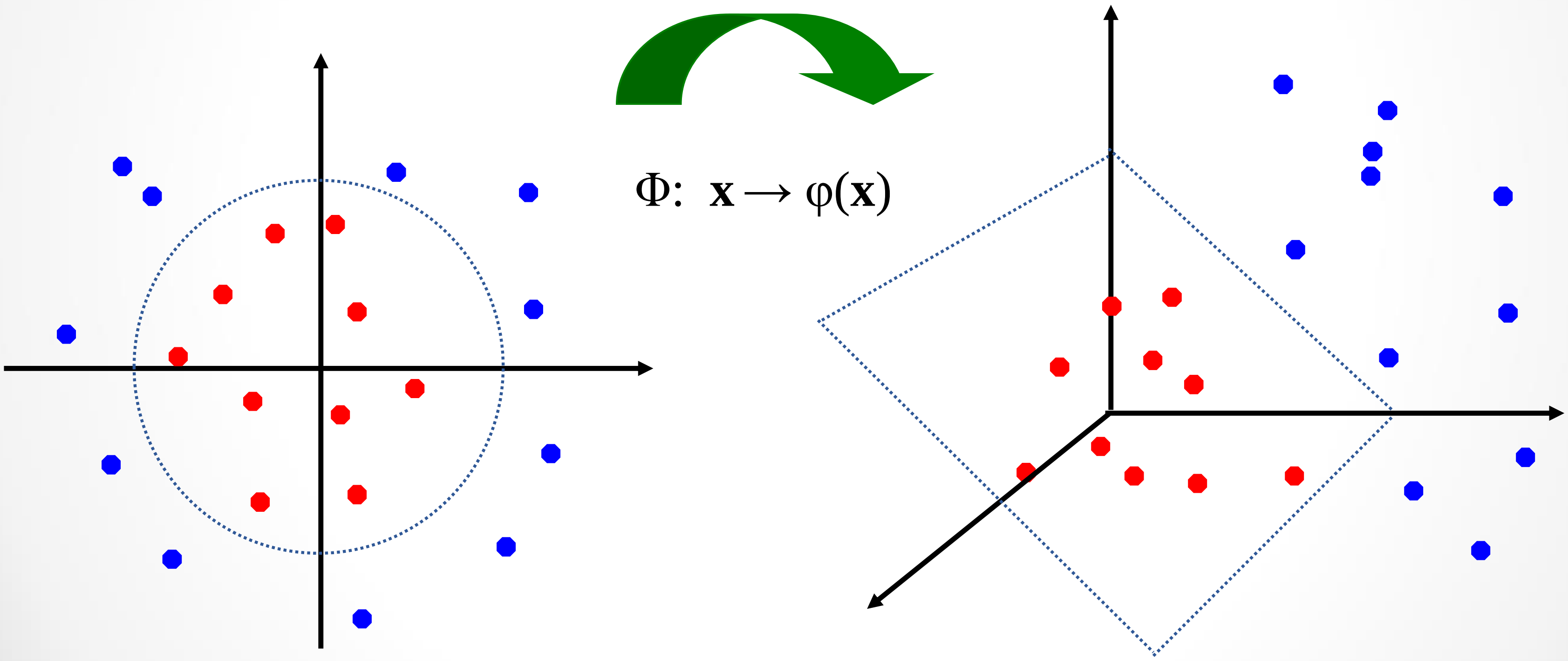
- Kernel Trick!!!
  - SVM = Linear SVM + Kernel Trick

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:



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: Kernel matrix should be positive semidefinite.

Nonlinear SVM: Optimization

- Formulation: (Lagrangian Dual Problem)

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

$$\text{such that} \quad 0 \le \alpha_i \le C$$

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

- 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.

Additional Resource

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