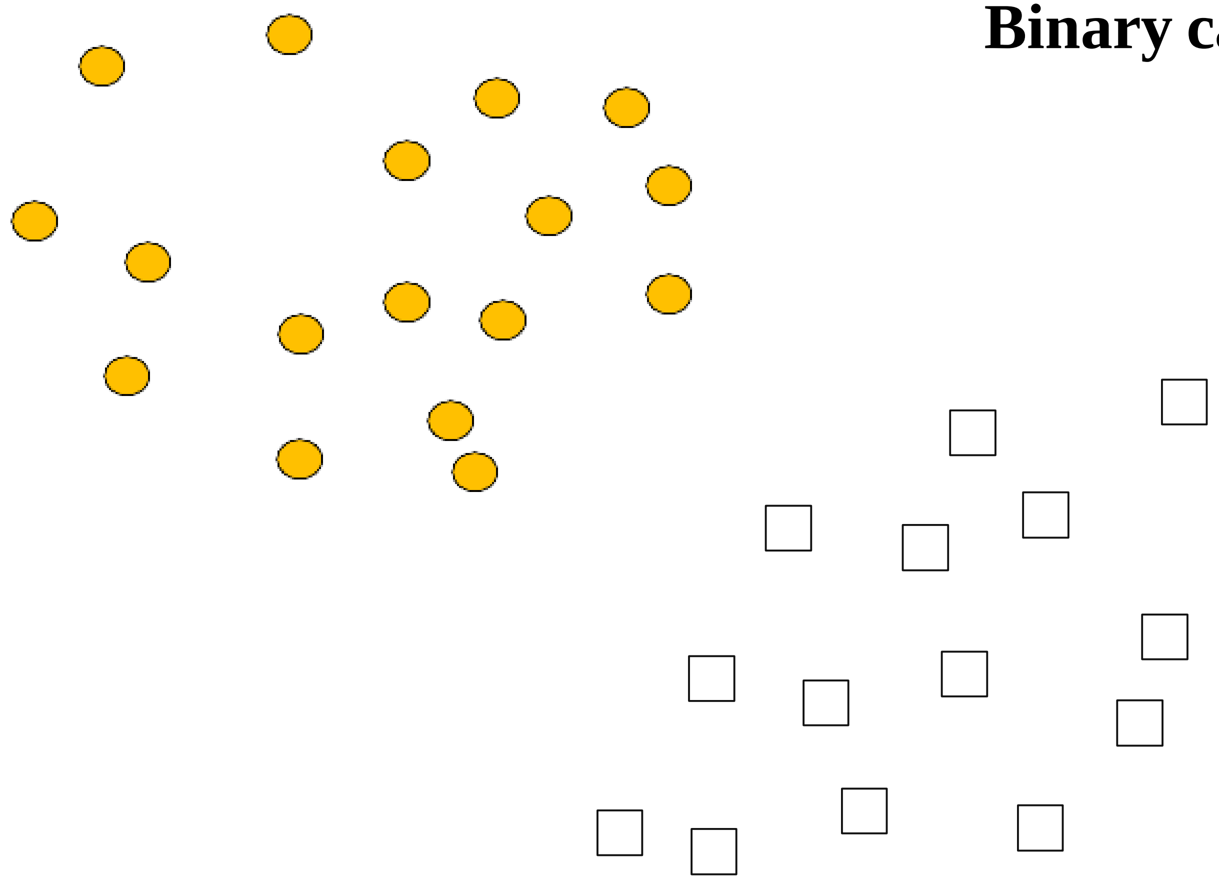
Support Vector Machines

Definition

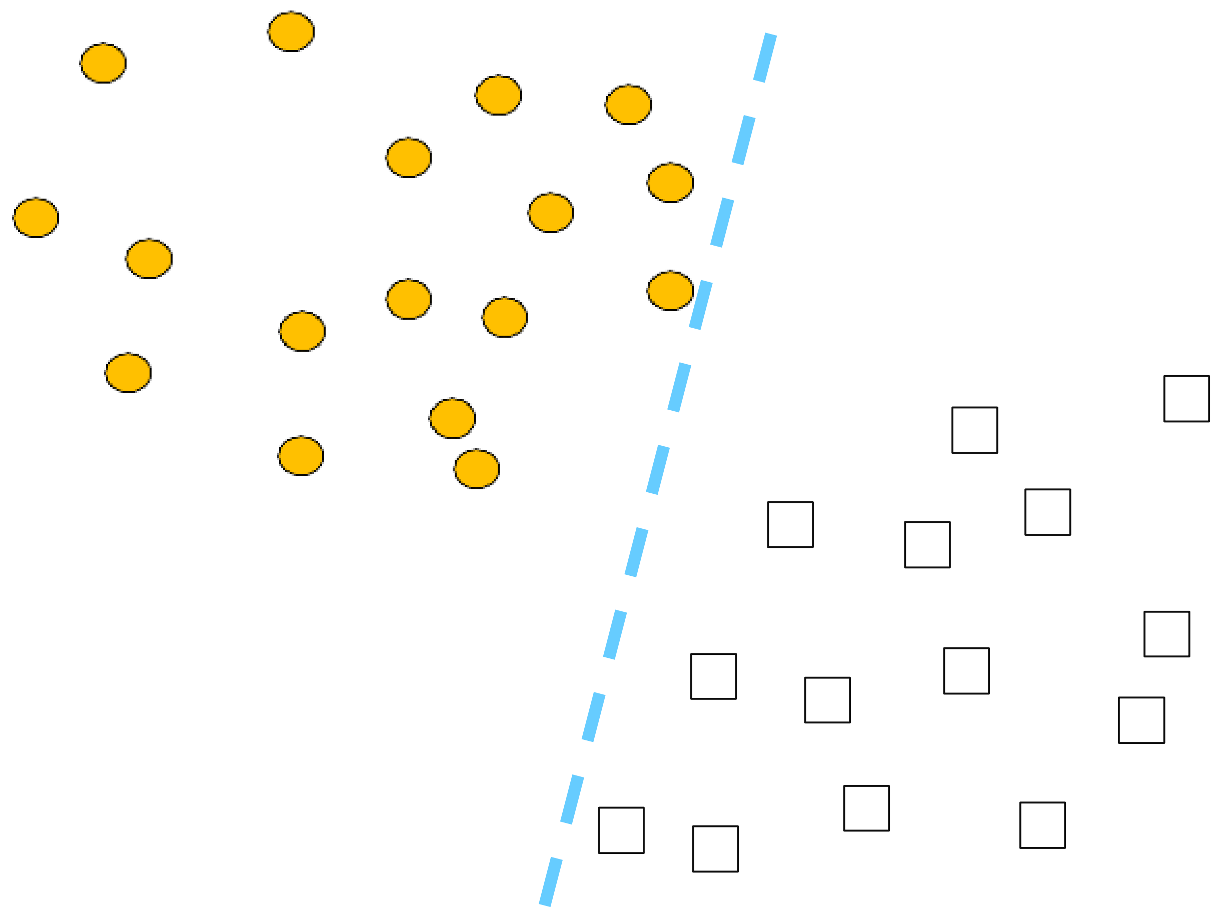
- ‘Support Vector Machine is a system for efficiently training linear learning machines in kernel-induced feature spaces, while respecting the insights of generalisation theory and exploiting optimisation theory.’
- Support Vector Machines is a comparatively new learning method that is used in general for the purpose of classification.
- Since the example data is often not linearly separable, SVM has spouted forth the notion of a “Kernel induced feature space”.
- Primary key used in SVMs is that the higher-dimensional space doesn’t need to be directly dealt with.
- SVMs are intuitive, are well-founded according to theory, and have proved to be practically successful

Which of the linear separators is optimal?

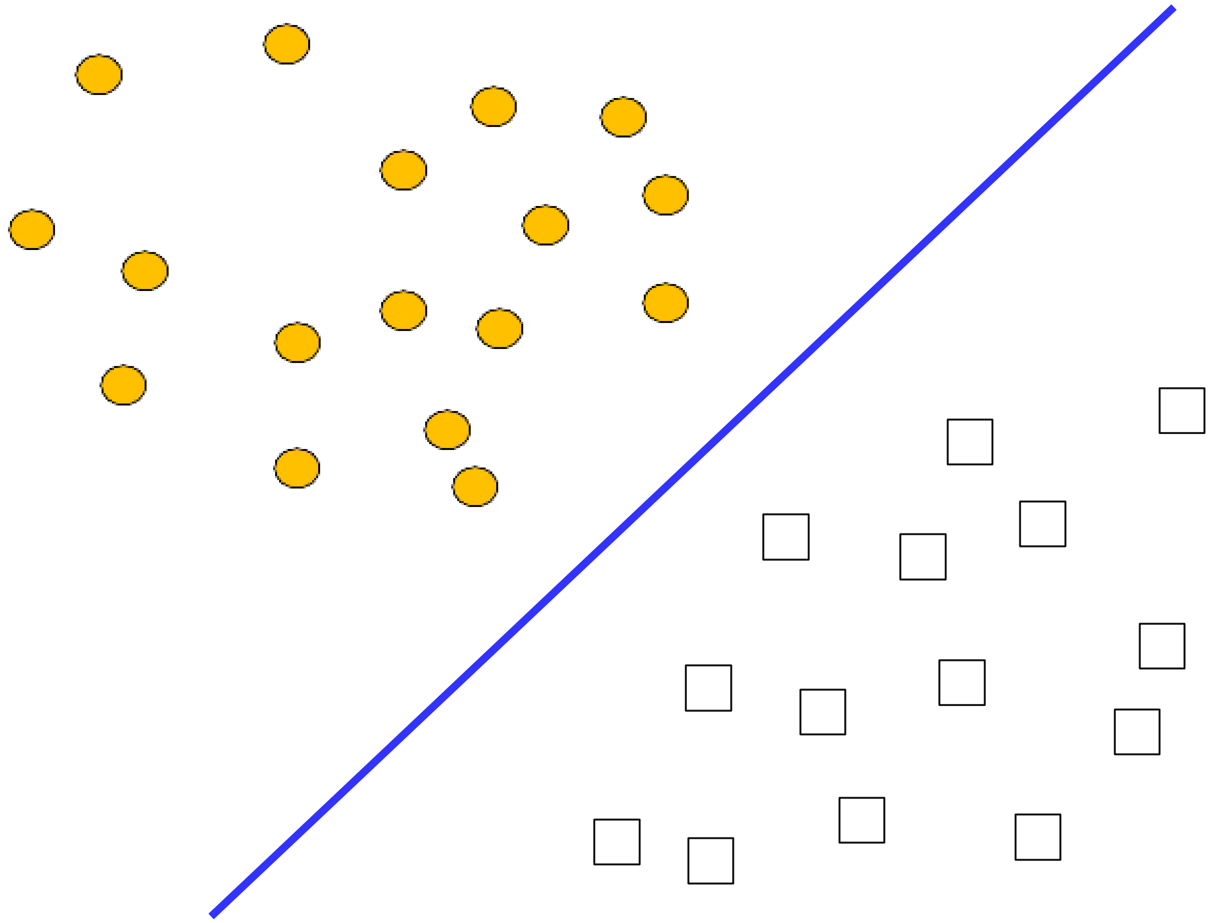
Binary case



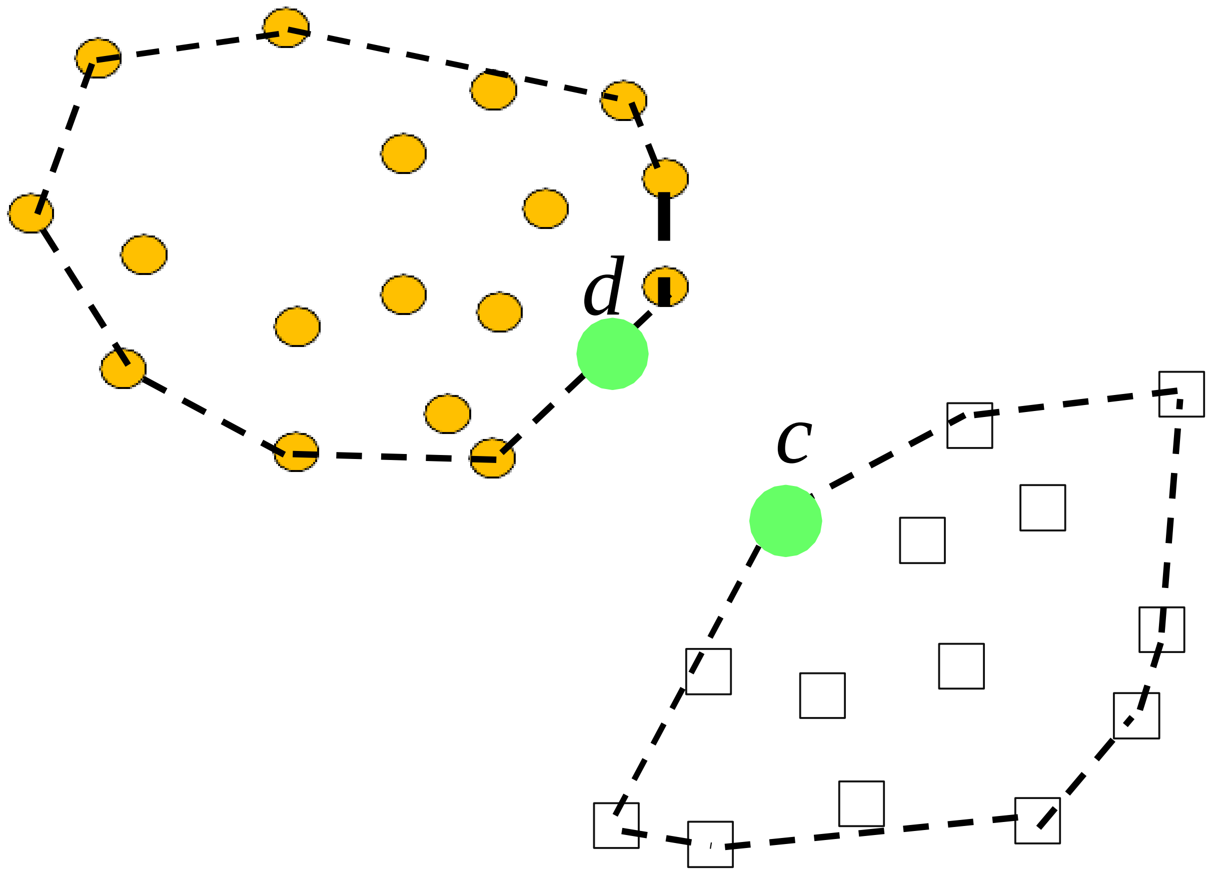
Best Linear Separator?



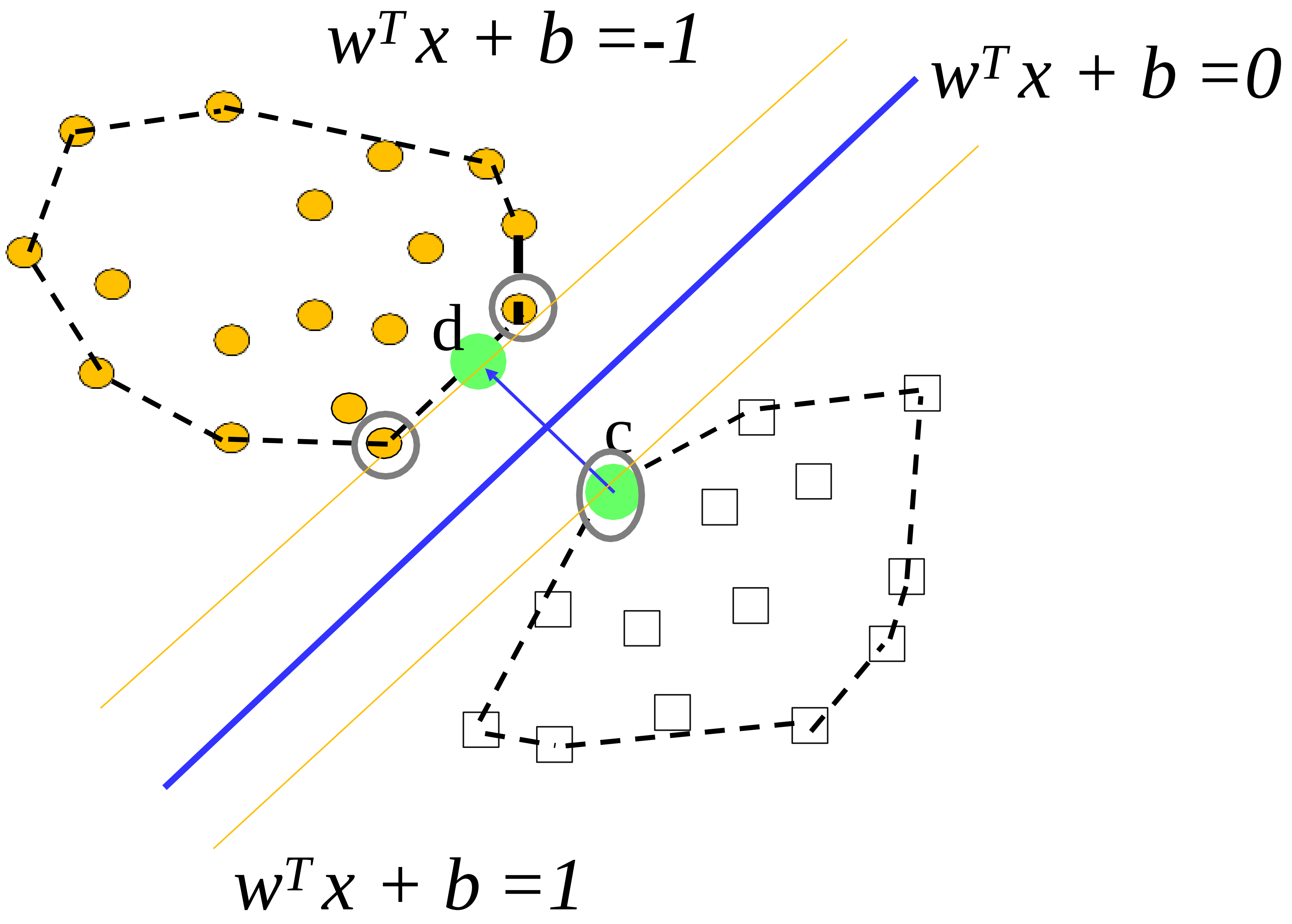
Best Linear Separator?



Find Closest Points in Convex Hulls

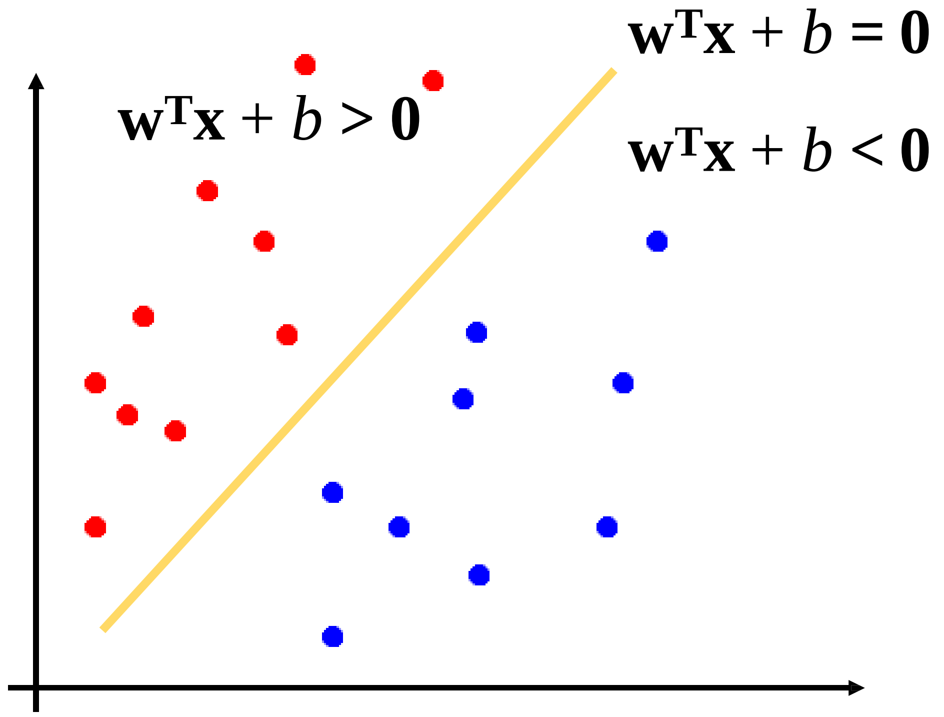


Plane Bisect Closest Points



Linear Discriminant Function

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



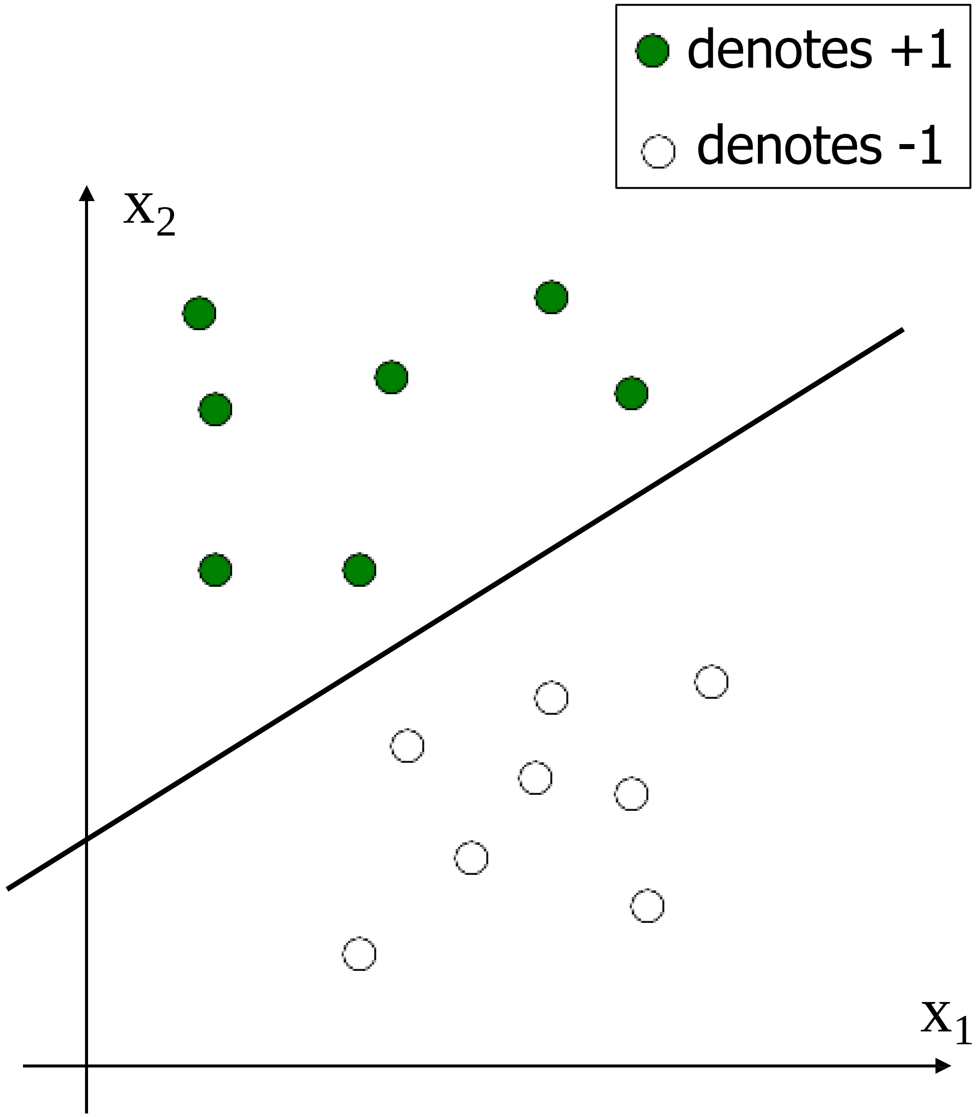
$\mathbf{w}^T\mathbf{x} + b = \mathbf{0}$
$\mathbf{w}^T\mathbf{x} + b > \mathbf{0}$
$\mathbf{w}^T\mathbf{x} + b < \mathbf{0}$

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

Linear Discriminant Function

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

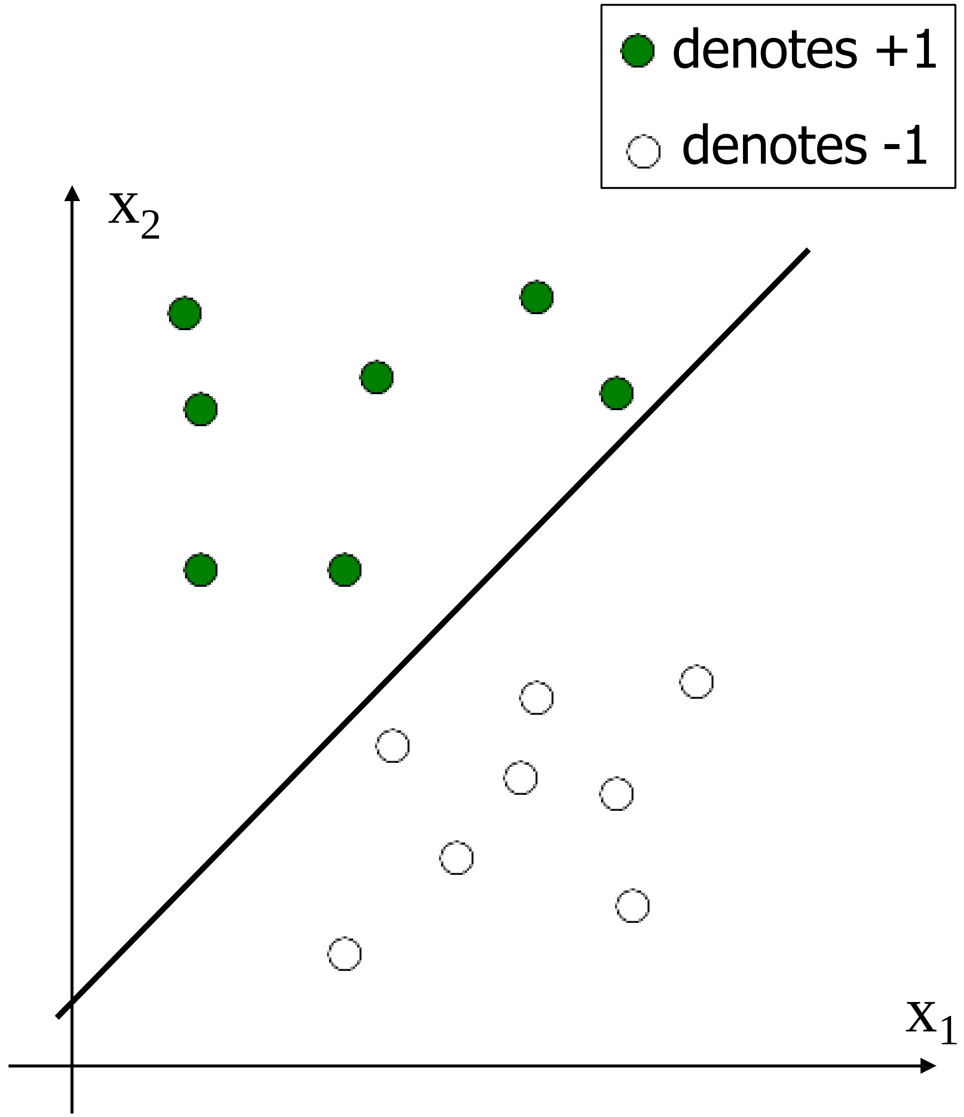
- Infinite number of answers!



Linear Discriminant Function

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

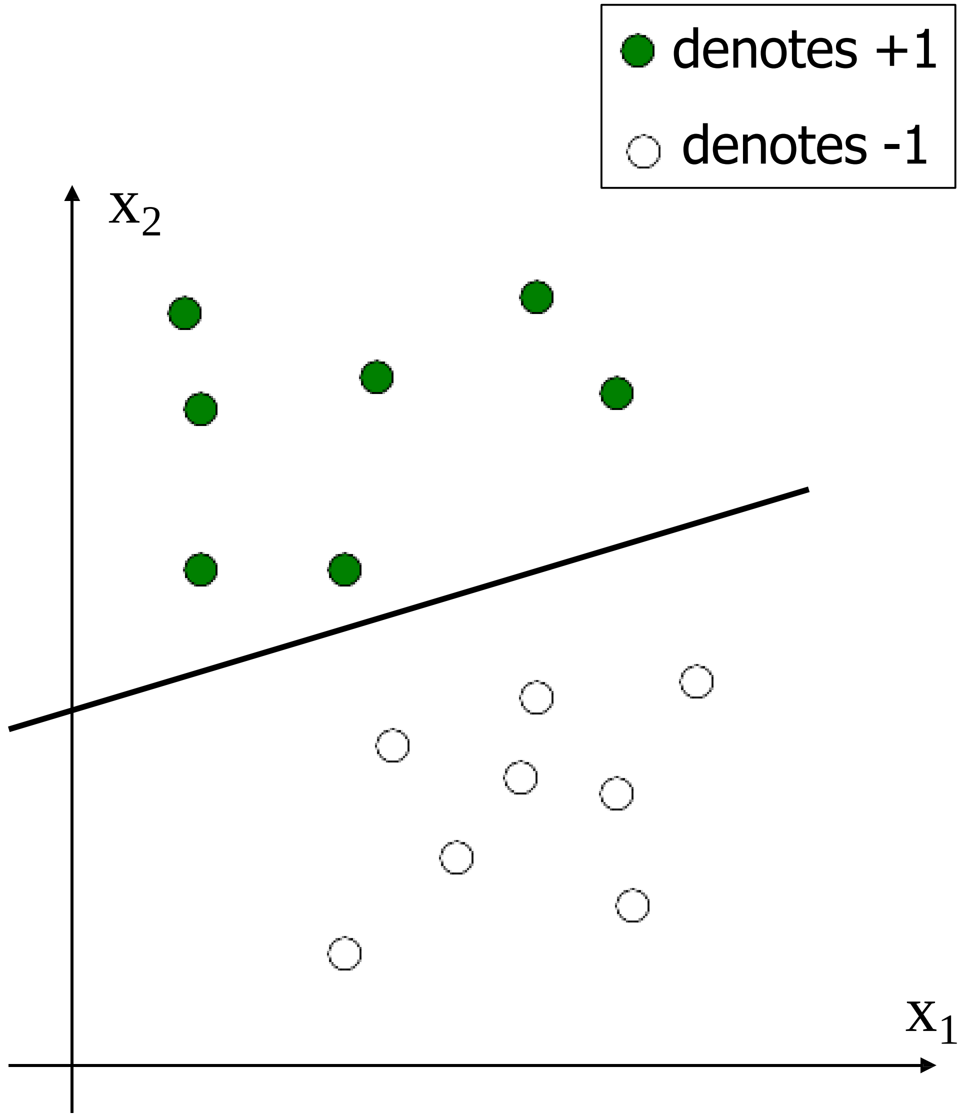
- Infinite number of answers!



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!

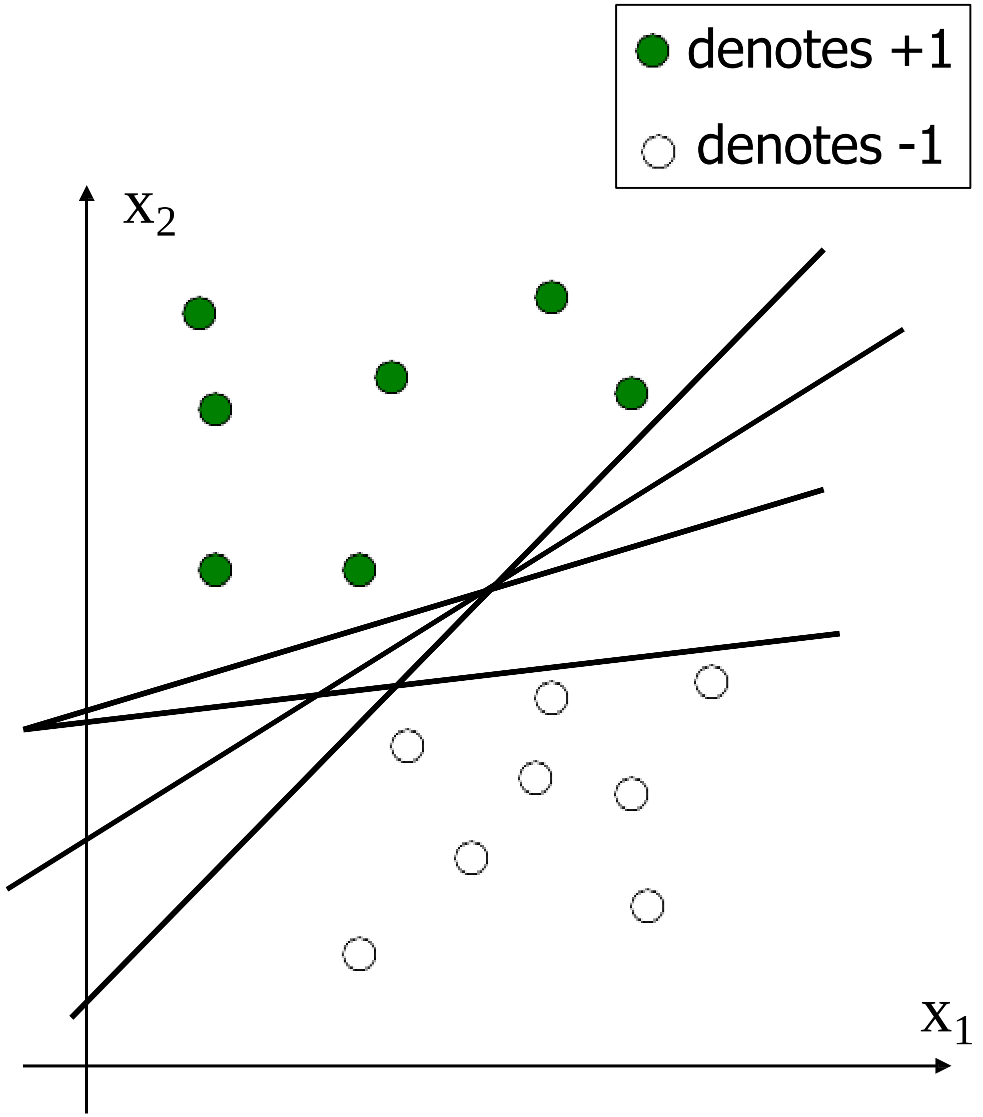


Linear Discriminant Function

- 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?

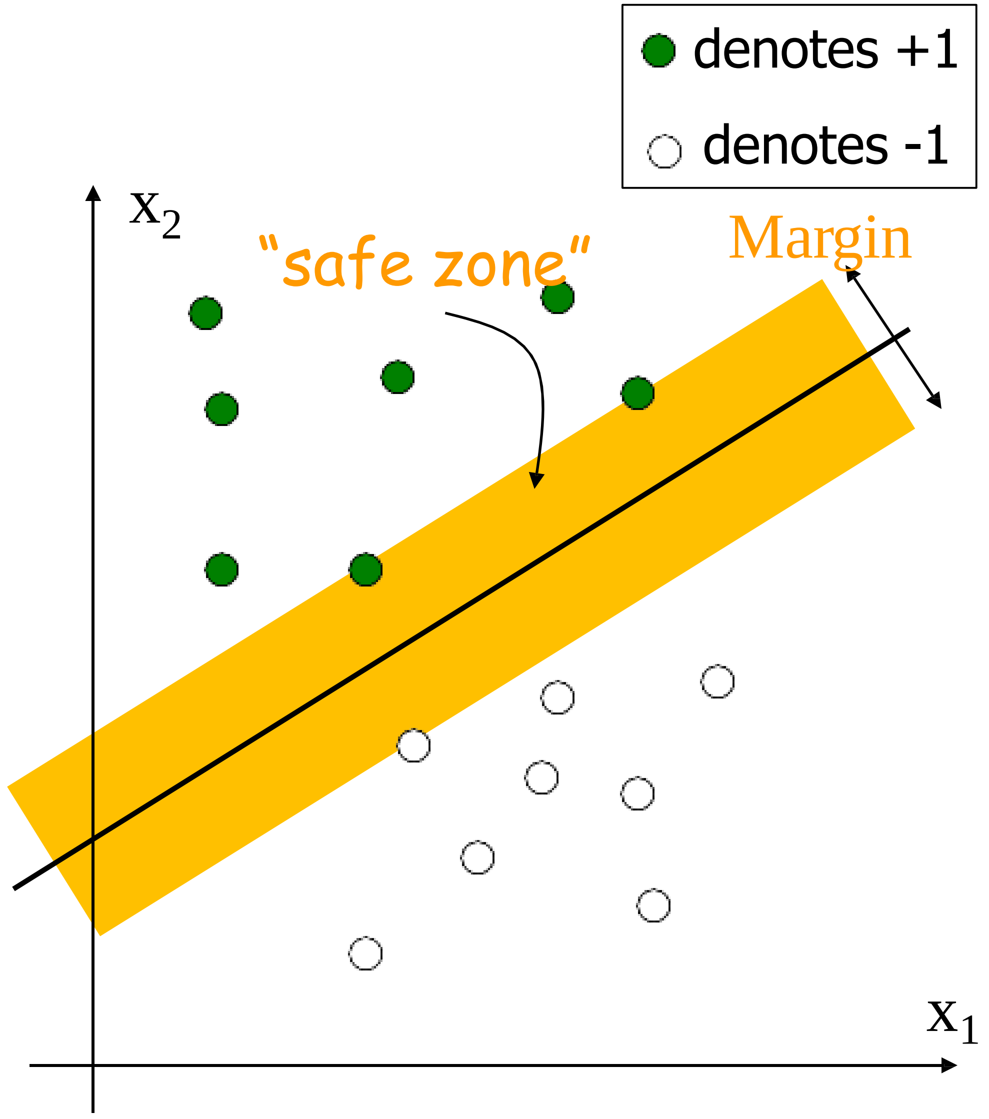


Large Margin Linear Classifier

- 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?
  - Robust to outliners and thus strong generalization ability



Large Margin Linear Classifier

- Given a set of data points:

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

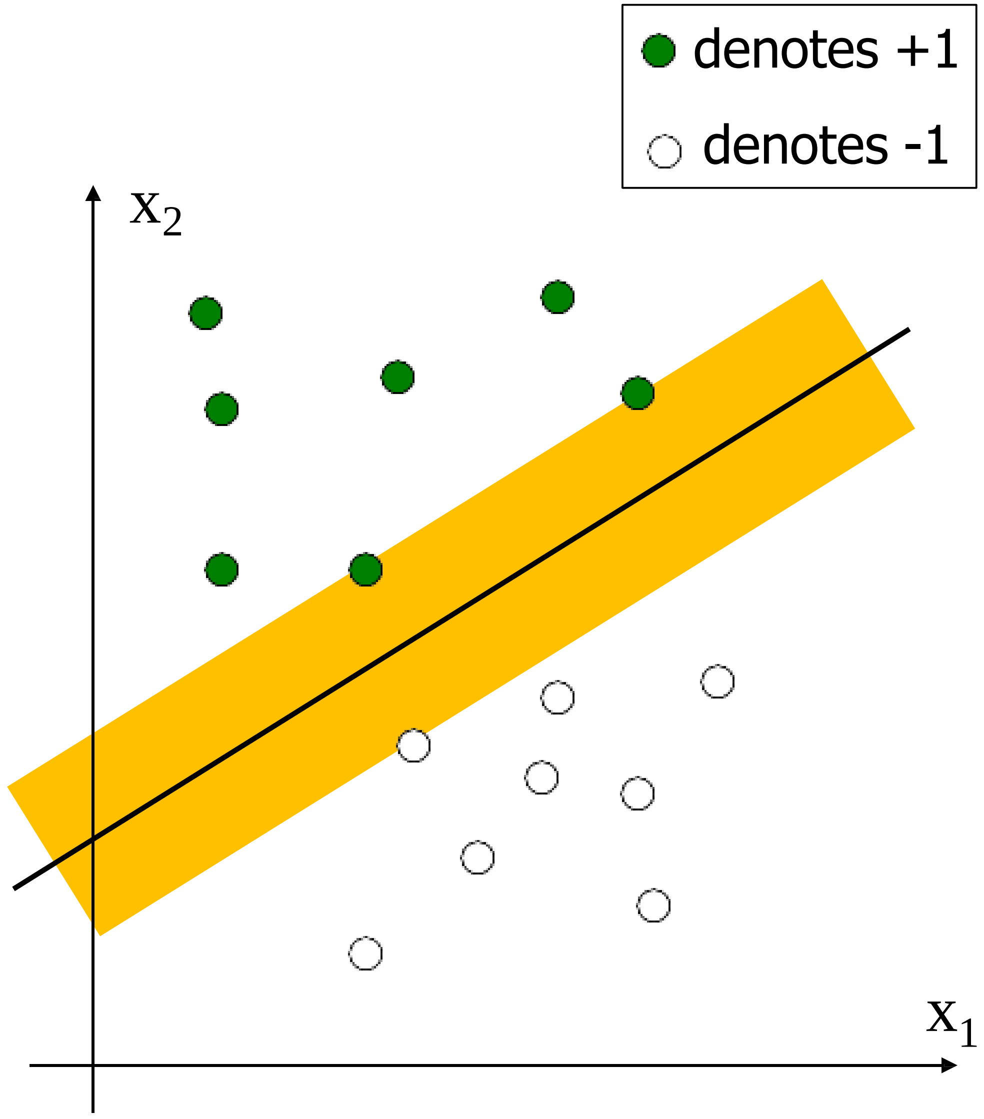
$$\text{For } y_i = +1,\quad \mathbf{w}^T\mathbf{x}_i + b > 0$$

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

- With a scale transformation on both and , 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

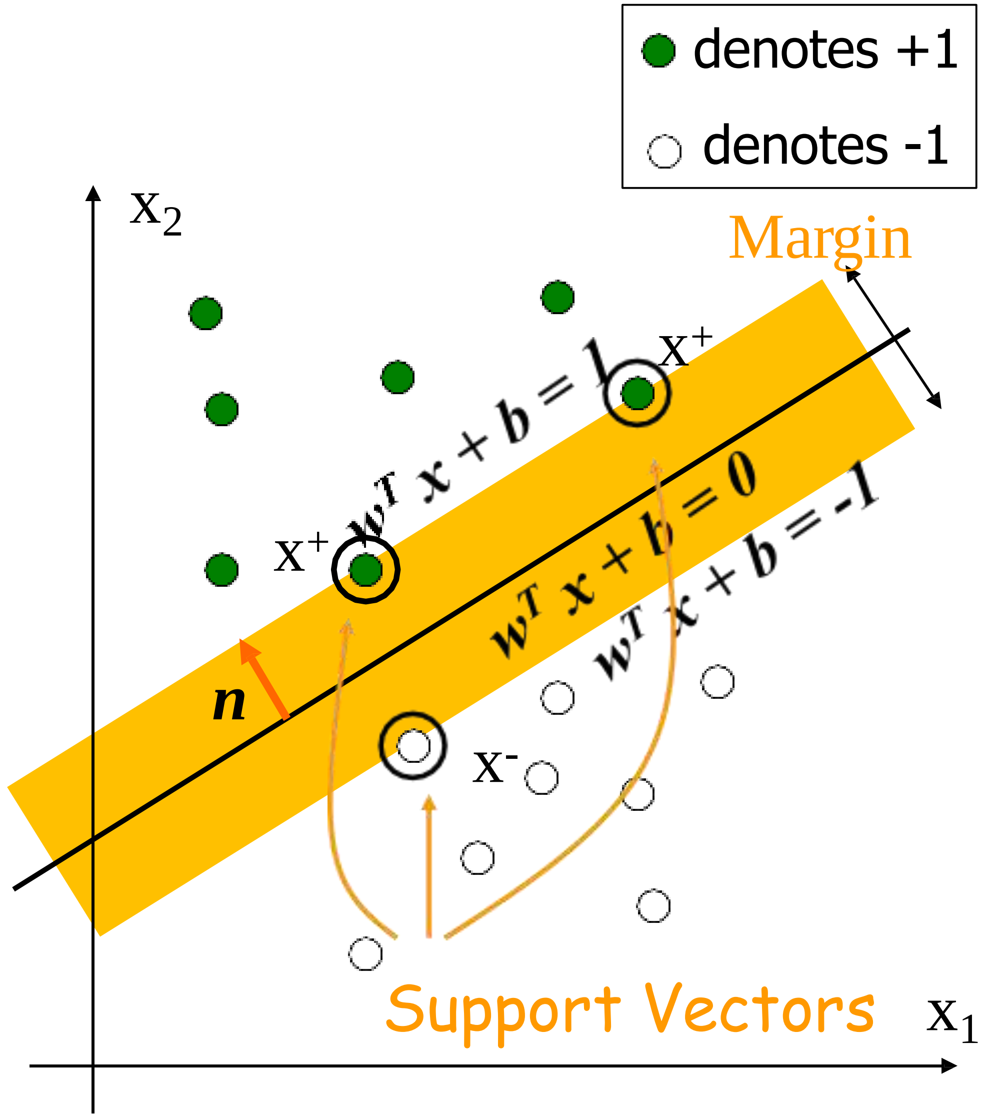
proschool An IMS Initiative

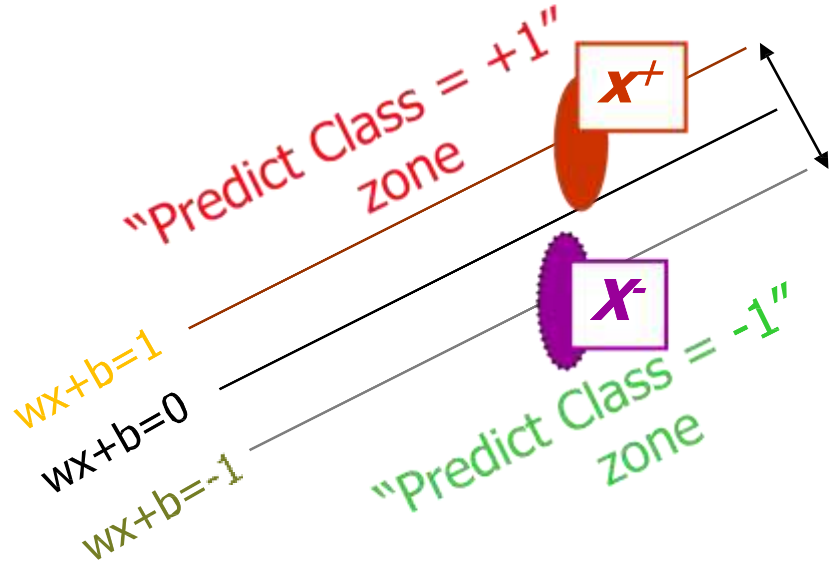
- We know that

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

- The **margin** width is:

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





M=Margin Width

What we know:

- $\boldsymbol{w} \,.\, \boldsymbol{x}^+ + b = +1$
- $\boldsymbol{w} \,.\, \boldsymbol{x}^- + b = -1$
- $\boldsymbol{w} \,.\, (\boldsymbol{x}^+ - \boldsymbol{x}^-) = 2$

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

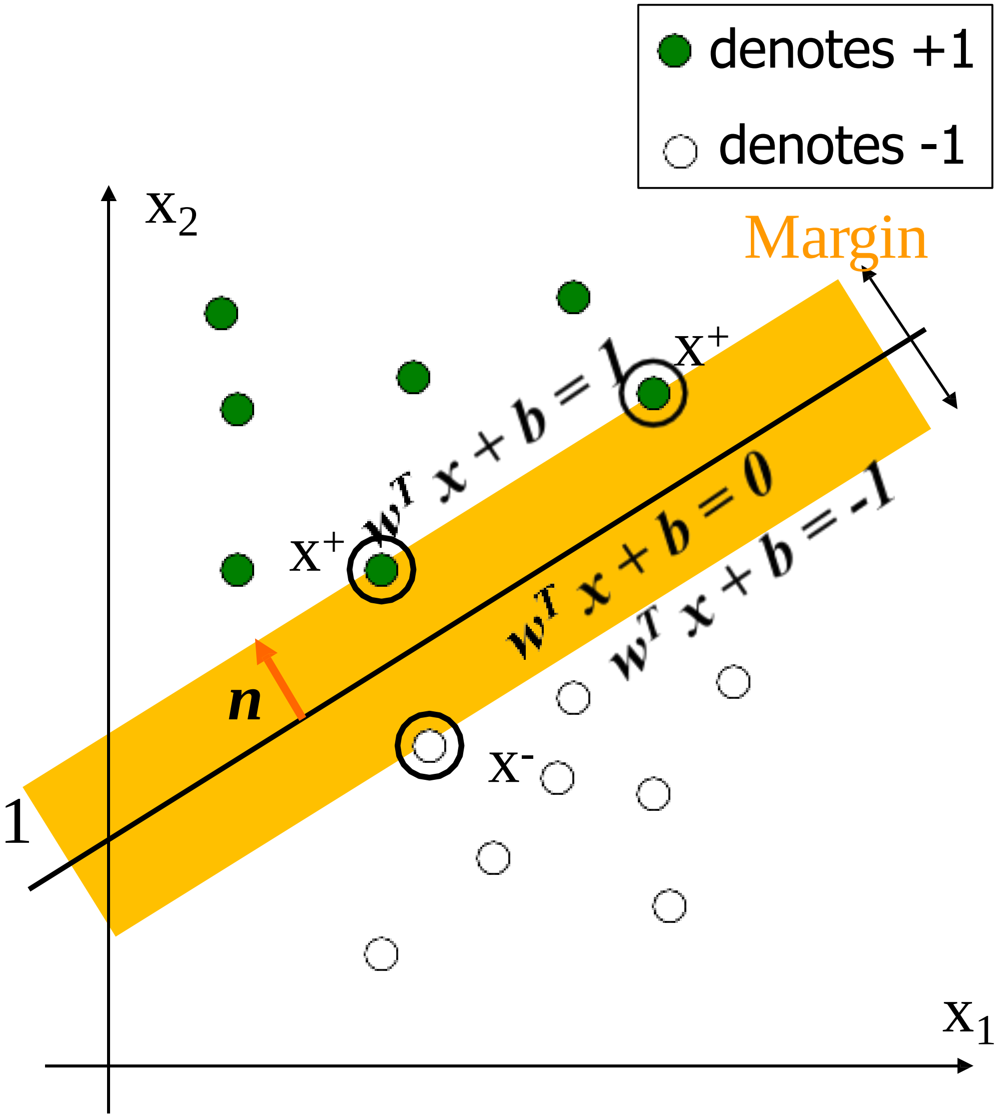
Large Margin Linear Classifier

- Formulation:

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

such that

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



proschool
An IMS Initiative

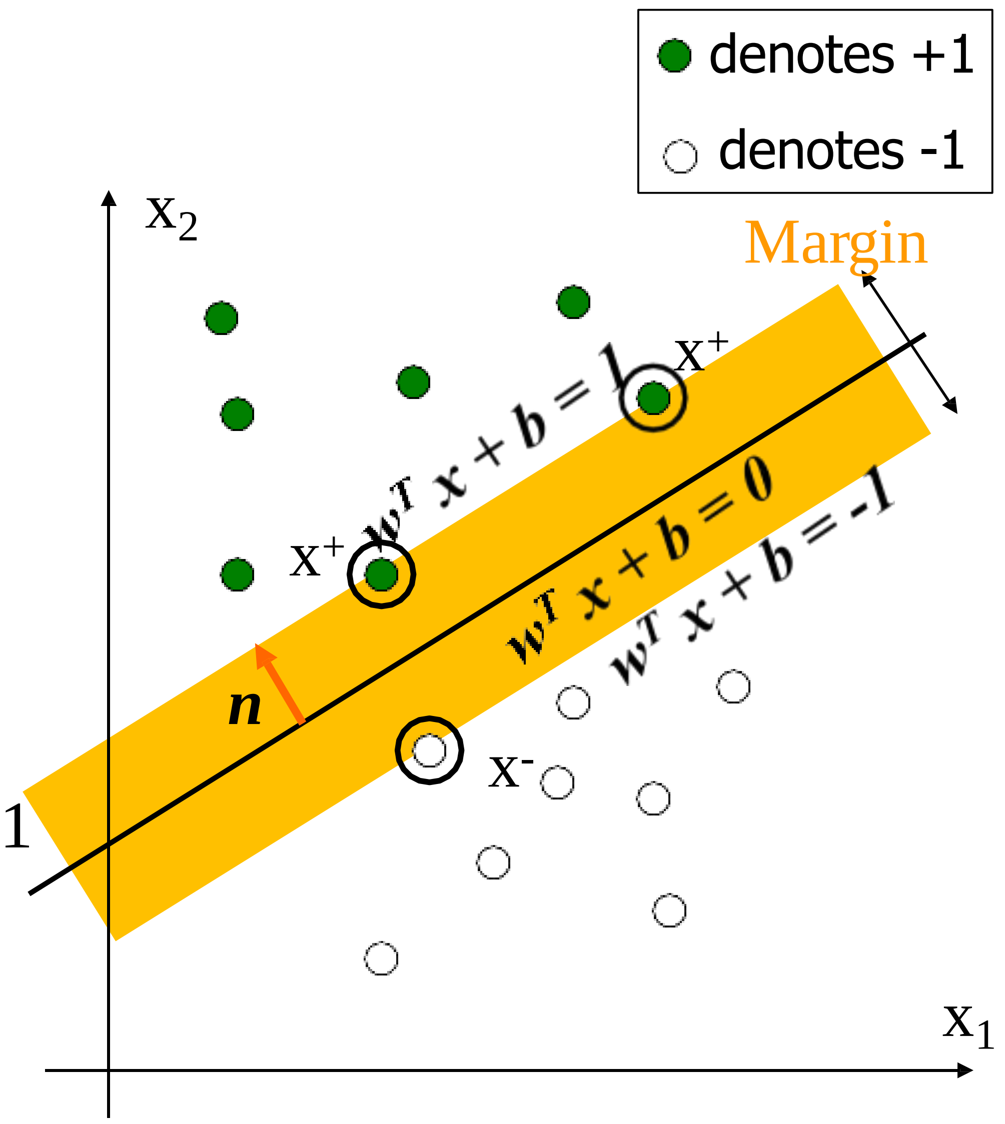
Large Margin Linear Classifier

- 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
Margin
x_2
x_1
x^+
x^+
x^-
$\boldsymbol{n}$
$\boldsymbol{w}^T\boldsymbol{x} + b = 1$
$\boldsymbol{w}^T\boldsymbol{x} + b = 0$
$\boldsymbol{w}^T\boldsymbol{x} + b = -1$

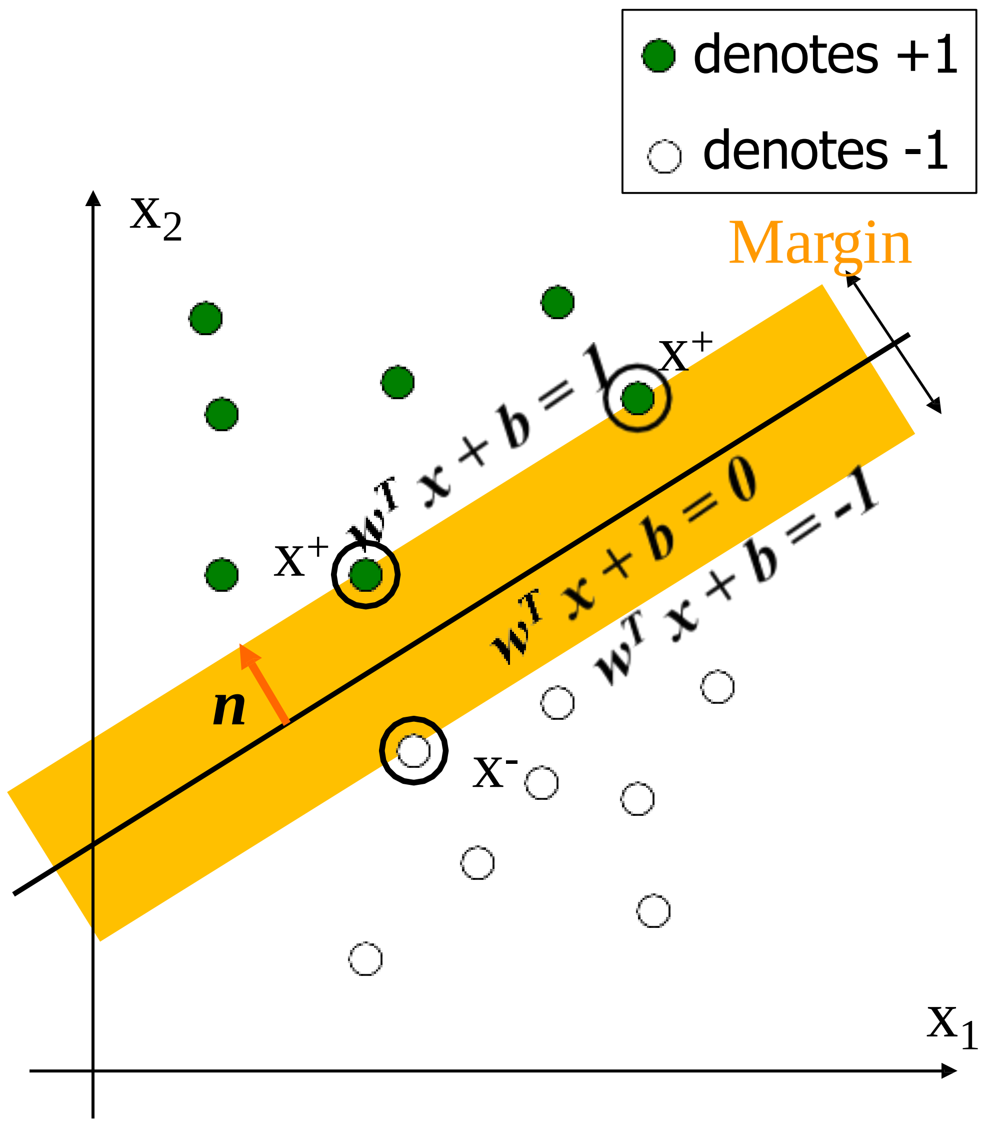
Large Margin Linear Classifier

- Formulation:

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

such that

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



Solving the Optimization Problem

Quadratic programming with linear constraints

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

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

$$\Updownarrow$$

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

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

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

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

Solving the Optimization Problem

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

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

Lagrangian Dual Problem

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

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

proschool
An IMS Initiative

Solving the Optimization Problem

- From KKT condition, we know:

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

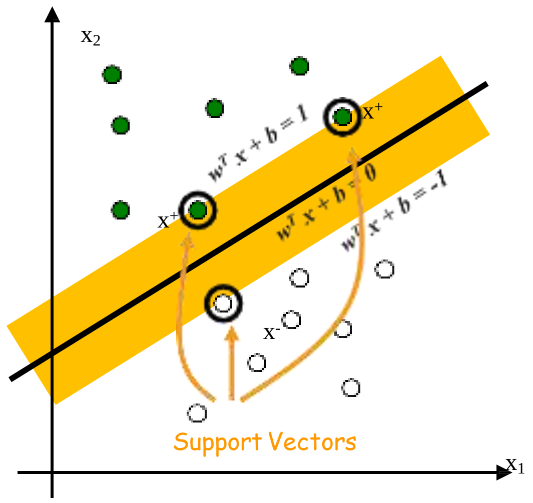
- Thus, only support vectors have $\alpha_i \neq 0$

- The solution has the form:

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

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 and the support vectors

- Also keep in mind that solving the optimization problem involved computing the dot products between all pairs of training points

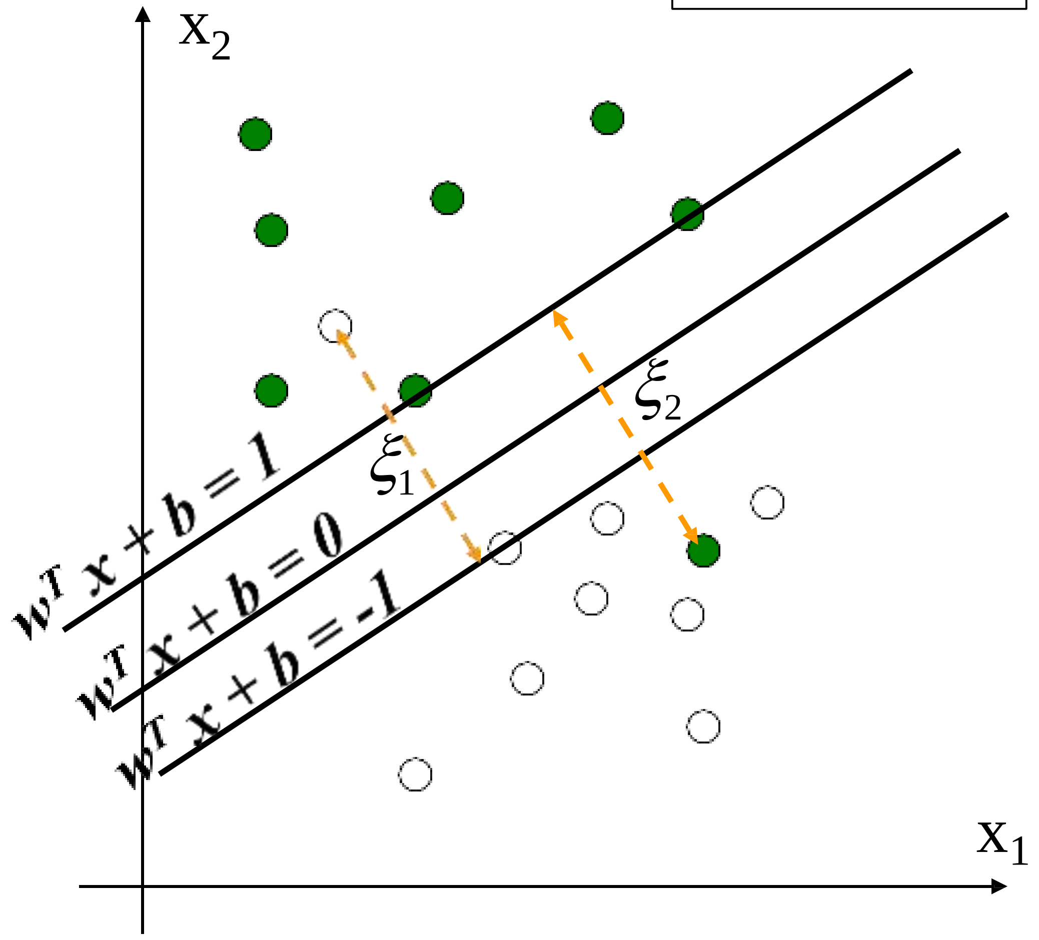
Large Margin Linear Classifier

- 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



Introducing slack variables

- Slack variables are constrained to be non-negative. When they are greater than zero they allow us to cheat by putting the plane closer to the datapoint than the margin. So we need to minimize the amount of cheating. This means we have to pick a value for lamba (this sounds familiar!)

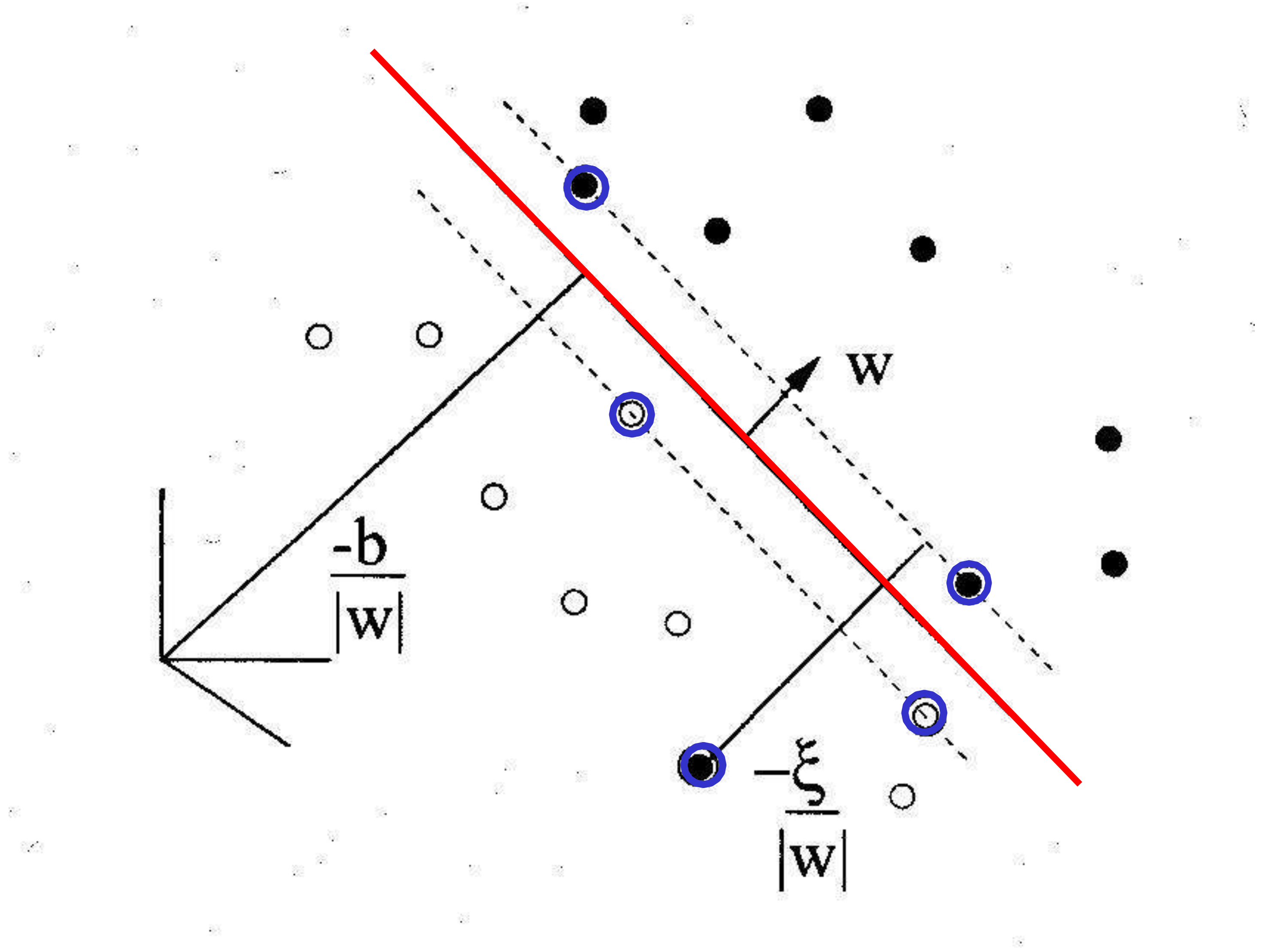
$$\mathbf{w}.\mathbf{x}^c + b \geq +1 - \xi^c \quad \textit{for positive cases}$$

$$\mathbf{w}.\mathbf{x}^c + b \leq -1 + \xi^c \quad \textit{for negativecases}$$

$$\textit{with} \quad \xi^c \geq 0 \quad \textit{for all}\, c$$

$$\textit{and} \quad \frac{\|\mathbf{w}\|^2}{2} + \lambda \sum_c \xi^c \quad \textit{as small as possible}$$

A picture of the best plane with a slack variable



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.

proschool
An ims Initiative

Large Margin Linear Classifier

- Formulation: (Lagrangian Dual Problem)

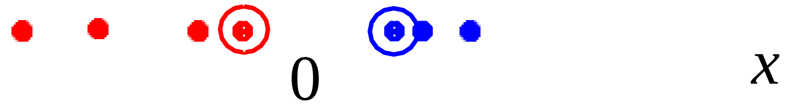
$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

such that

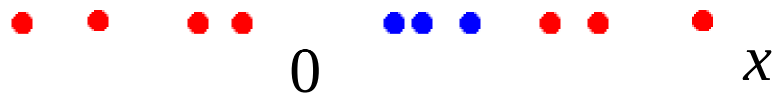
$$0 \le \alpha_i \le C$$

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

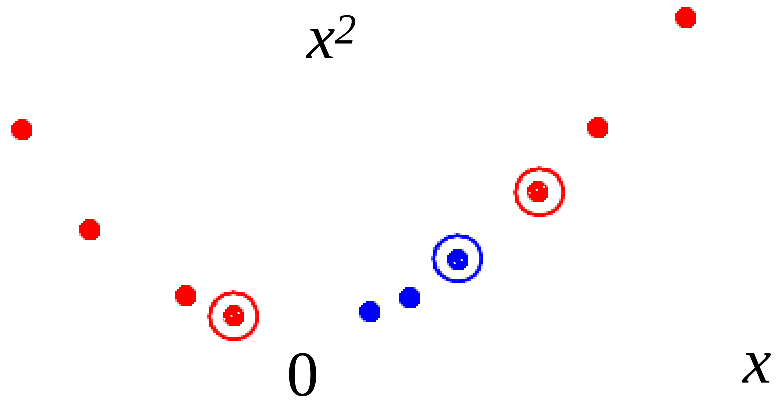
- Datasets that are linearly separable with noise work out great:



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

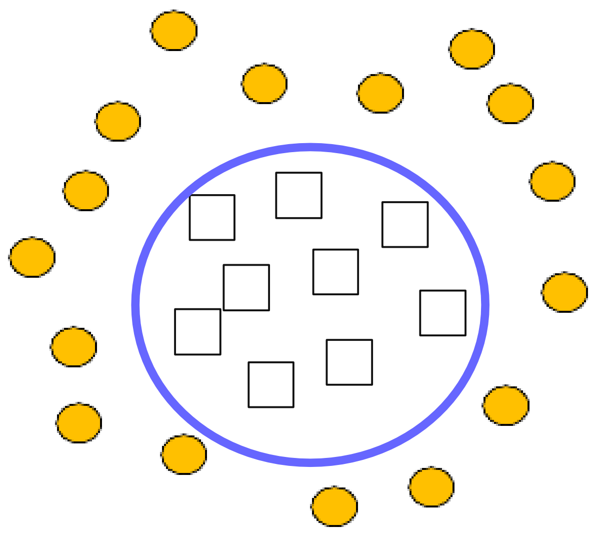


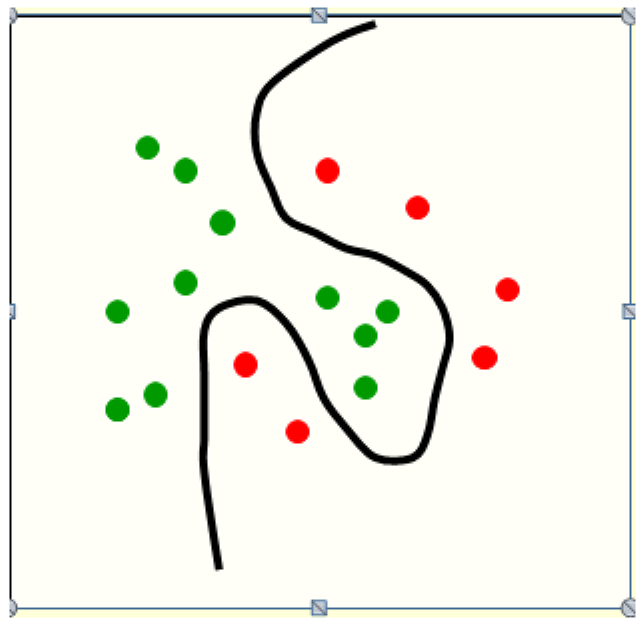
- How about… mapping data to a higher-dimensional space:



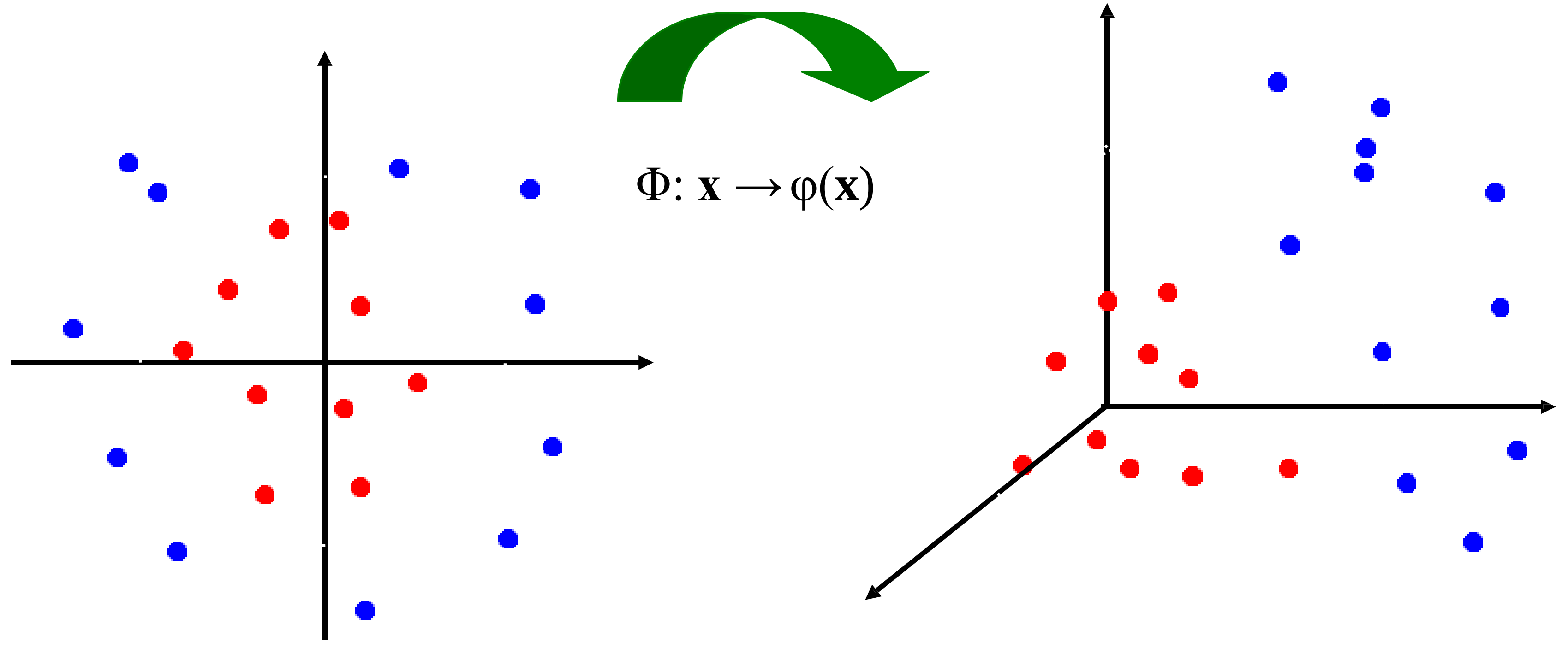
This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Nonlinear Classification



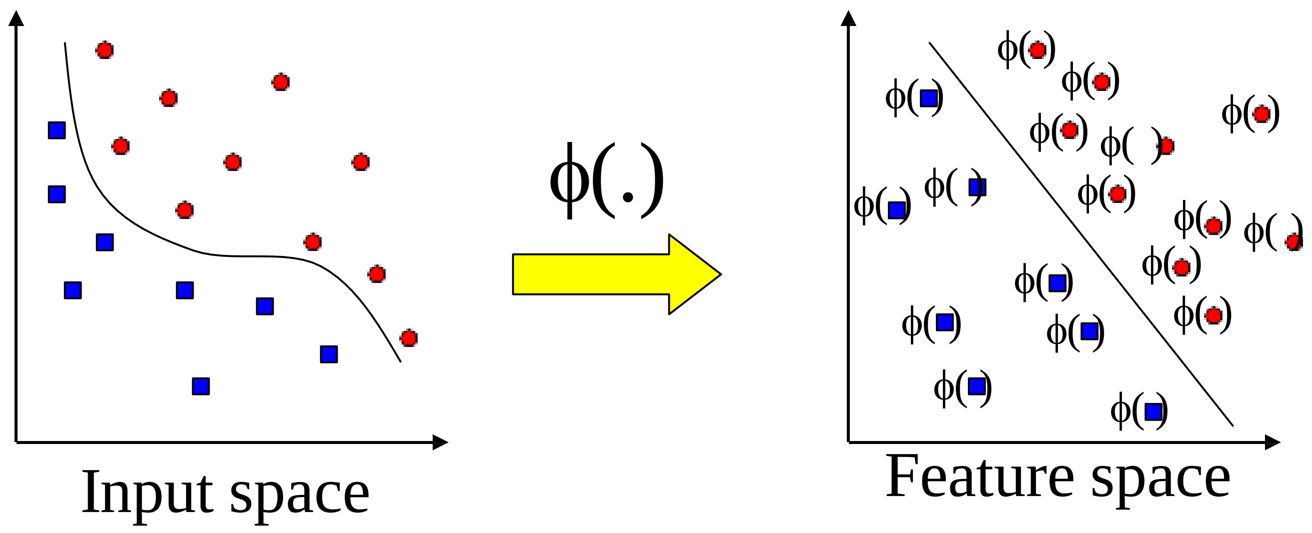


- General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is separable:



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

Transforming the Data



- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

- With this mapping, our discriminant function is now:

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

- 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:

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

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

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

- 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.

- 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

proschool
An IMS Initiative

- 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

Strengths and Weaknesses of SVM

- Strengths
  - Training is relatively easy
    - No local optimal, unlike in neural networks
  - It scales relatively well to high dimensional data
  - Tradeoff between classifier complexity and error can be controlled explicitly
  - Non-traditional data like strings and trees can be used as input to SVM, instead of feature vectors
  - By performing logistic regression (Sigmoid) on the SVM output of a set of data can map SVM output to probabilities.

- Weaknesses
  - Need to choose a "good" kernel function.

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/

Packages used in Case Studies

- **datetime** - In Python, date, time and datetime classes provides a number of function to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps. Whenever you manipulate dates or time, you need to import datetime function.

- **Sklearn -** scikit-learn is a collection of Python modules relevant to machine/statistical learning and data mining.

- **Joblib -** Joblib is a set of tools to provide lightweight pipelining in Python. In particular, joblib offers: transparent disk-caching of the output values and lazy re-evaluation (memoize pattern).

- **Pandas_profiling** - Generates profile reports from a pandas DataFrame. The pandas df.describe() function is great but a little basic for serious exploratory data analysis. pandas_profiling extends the pandas DataFrame with df.profile_report() for quick data analysis.

- **Seaborn** - Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

- **Warnings** - Warning are typically issued in situations where it is useful to alert the user of some condition in a program, where that condition (normally) doesn’t warrant raising an exception and terminating the program.

proschool
An IMS Initiative

Thank You.