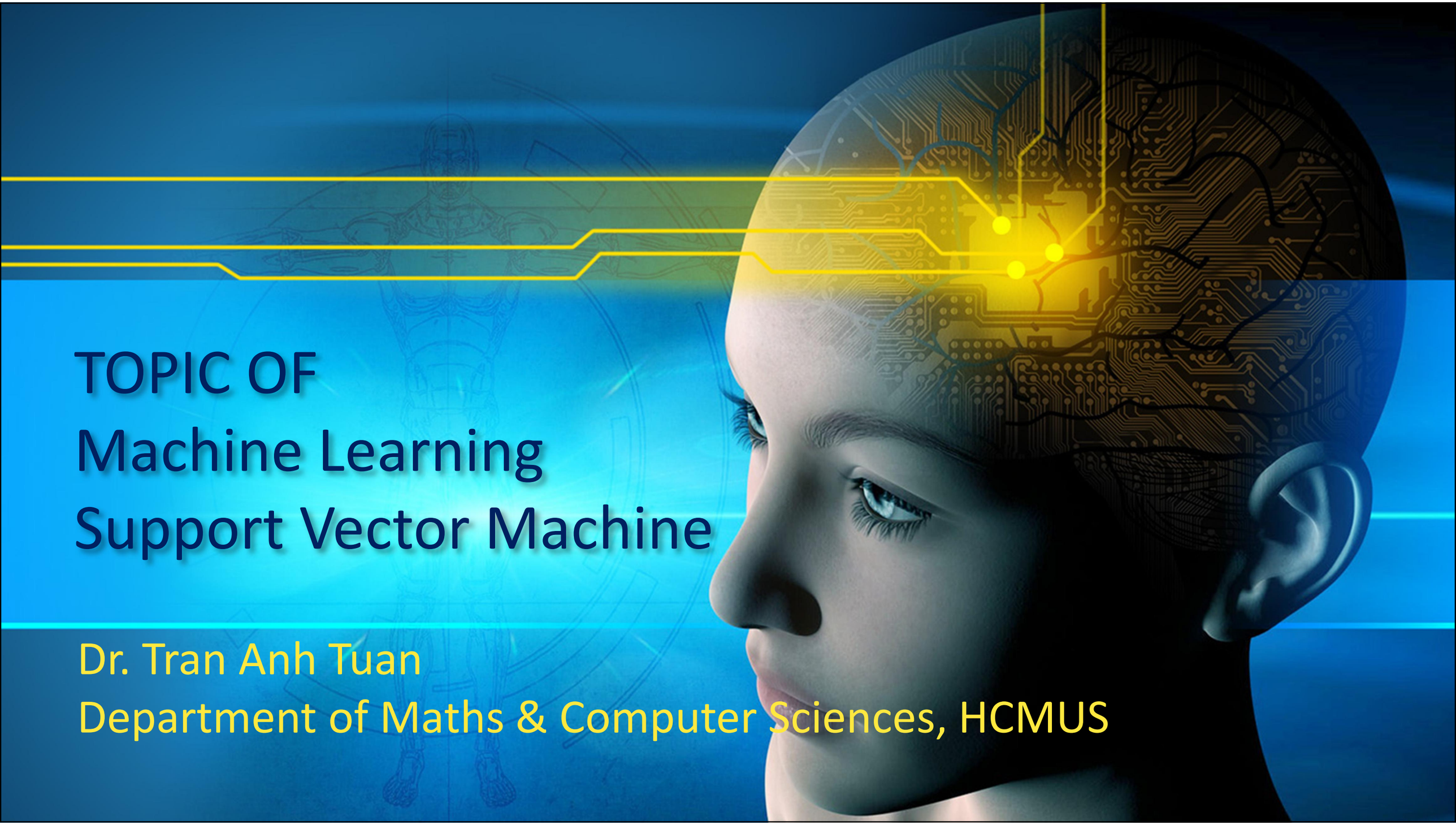
TOPIC OF
Machine Learning
Support Vector Machine



Dr. Tran Anh Tuan
Department of Maths & Computer Sciences, HCMUS

Content

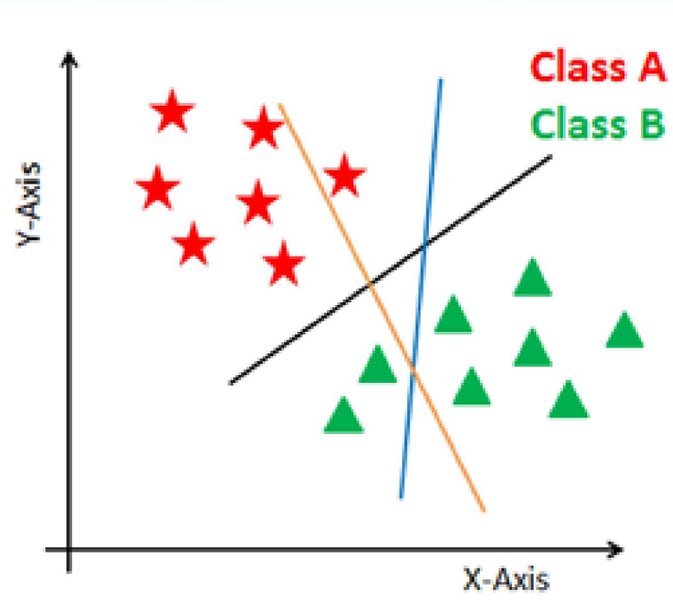
Support Vector Machine

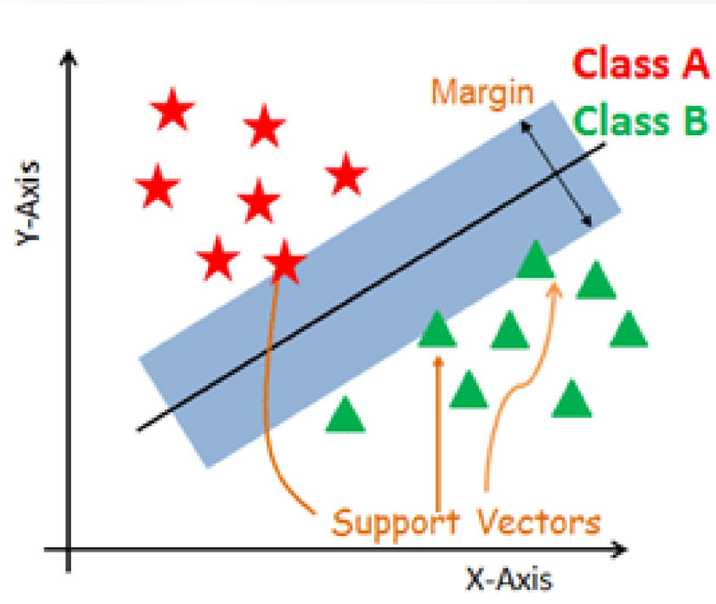
- Introduction
- Kuhn-Tucker Theorem

Practice

- Use SVM to to distinguish between different types of fruits
- Use SVM to detect an object in image
- Homework

Support Vector Machine ?

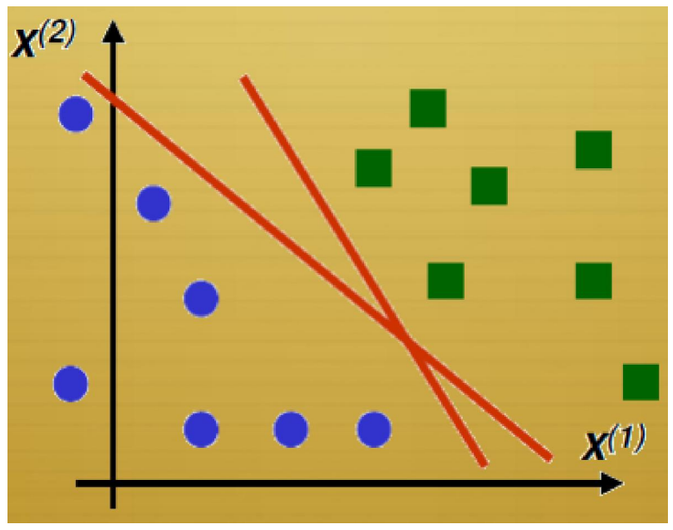




Introduction

- Said to start in 1979 with Vladimir Vapnik's paper
- Major developments throughout 1990's
- Elegant theory
  - Has good generalization properties
- Have been applied to diverse problems very successfully in the last 10-15 years
- One of the most important developments in pattern recognition in the last 10 years

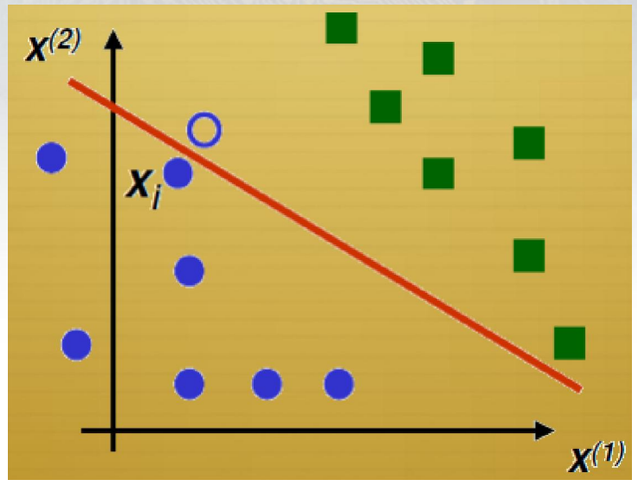
which separating hyperplane should we choose?



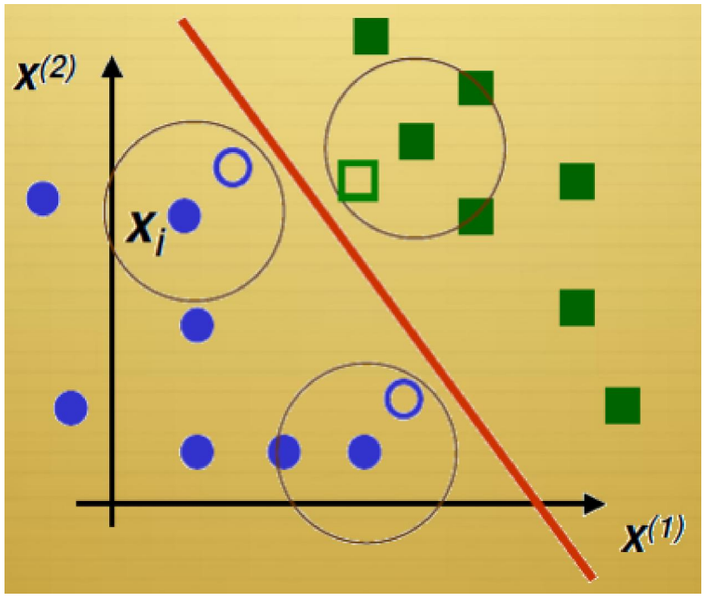
- Linear Discriminant Functions can be written as
- $g(x) = w^t x + w_0$
  - $g(x) > 0$ ➔ x is belong to class 1
  - $g(x) < 0$ ➔ x is belong to class 2

Introduction

- Training data is just a subset of all possible data
  - Suppose hyperplane is close to sample xi
  - If we see new sample close to sample i, it is likely to be on the wrong side of the hyperplane.
- Poor generalization (performance on unseen data)
- New samples close to the old samples will be classified correctly
  - Good generalization

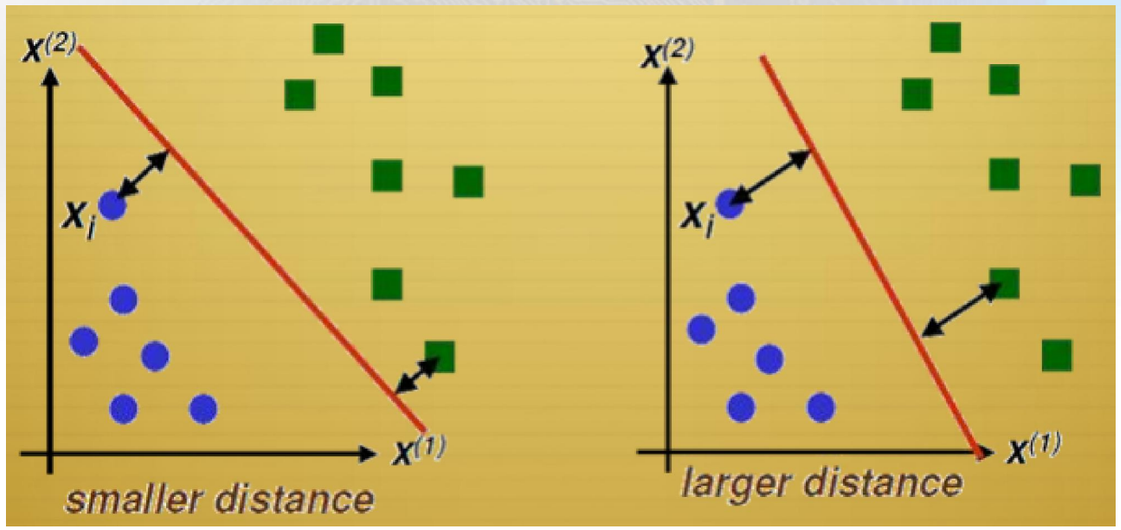


- Linear Discriminant Functions can be written as
- $g(x) = w^t x + w_0$
  - $g(x) > 0$ ➔ x is belong to class 1
  - $g(x) < 0$ ➔ x is belong to class 2

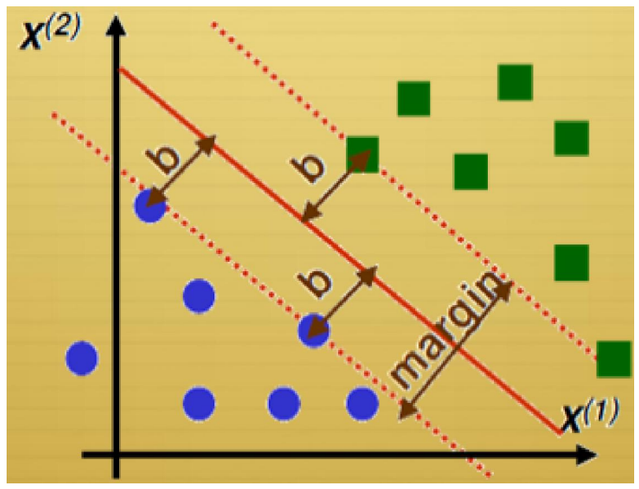


Introduction

- Idea #1: maximize distance to the closest example
- An optimal hyperplane
  - distance to the closest negative example = distance to the closest positive example



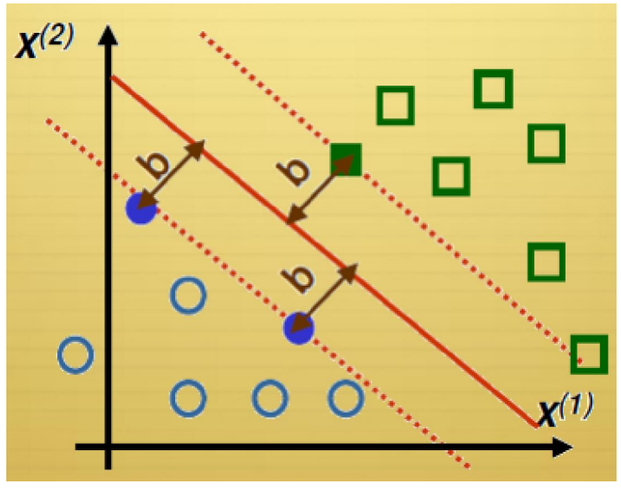
- Idea #2: maximize the margin.
- margin is twice the absolute value of distance b of the closest example to the separating hyperplane
  - Better generalization (performance on test data)
  - in both practice and theory



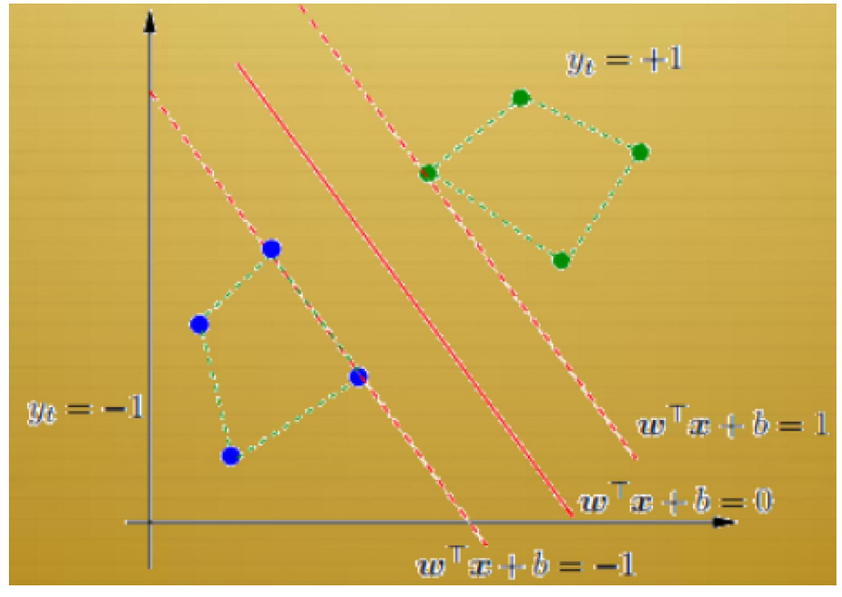
Introduction

- Support vectors are the samples closest to the separating hyperplane
  - they are the most difficult patterns to classify
  - Optimal hyperplane is completely defined by support vectors

➔we do not know which samples are support vectors without finding the optimal hyperplane.



- Canonical hyperplane: w and b are chosen such that for closest points $|w^t x + b| = 1$.
- For $\lambda \neq 0$, $(\lambda w, \lambda b)$ describes the same hyperplane as (w,b),
- i.e., $\{x| w^t x + b = 0\} = \{x|\lambda(w^t x + b) = 0\}$.



Introduction

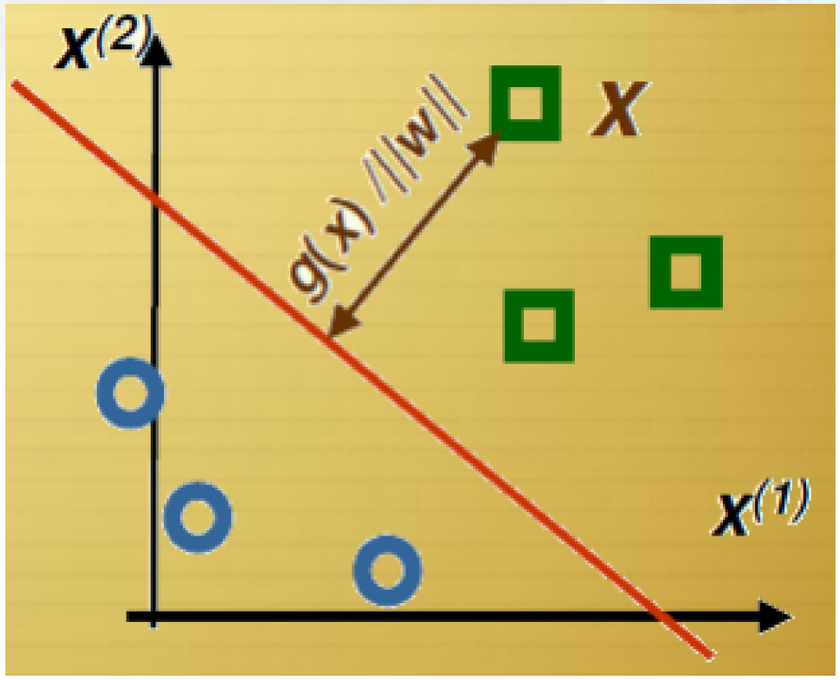
- Formula for the Margin
  - $g(x) = w^t x + w_0$
  - absolute distance between x and the boundary g(x) = 0

$$\frac{|w^t x + w_0|}{\|w\|}$$

  - distance is unchanged for hyperplane $g_1(x) = \alpha g(x)$

$$\frac{|\alpha w^t x + \alpha w_0|}{\|\alpha w\|} = \frac{|w^t x + w_0|}{\|w\|}$$

  - Let xi be an example closest to the canonical hyperplane. $|w^t x_i + w_0| = 1$
  - now the largest margin hyperplane is unique



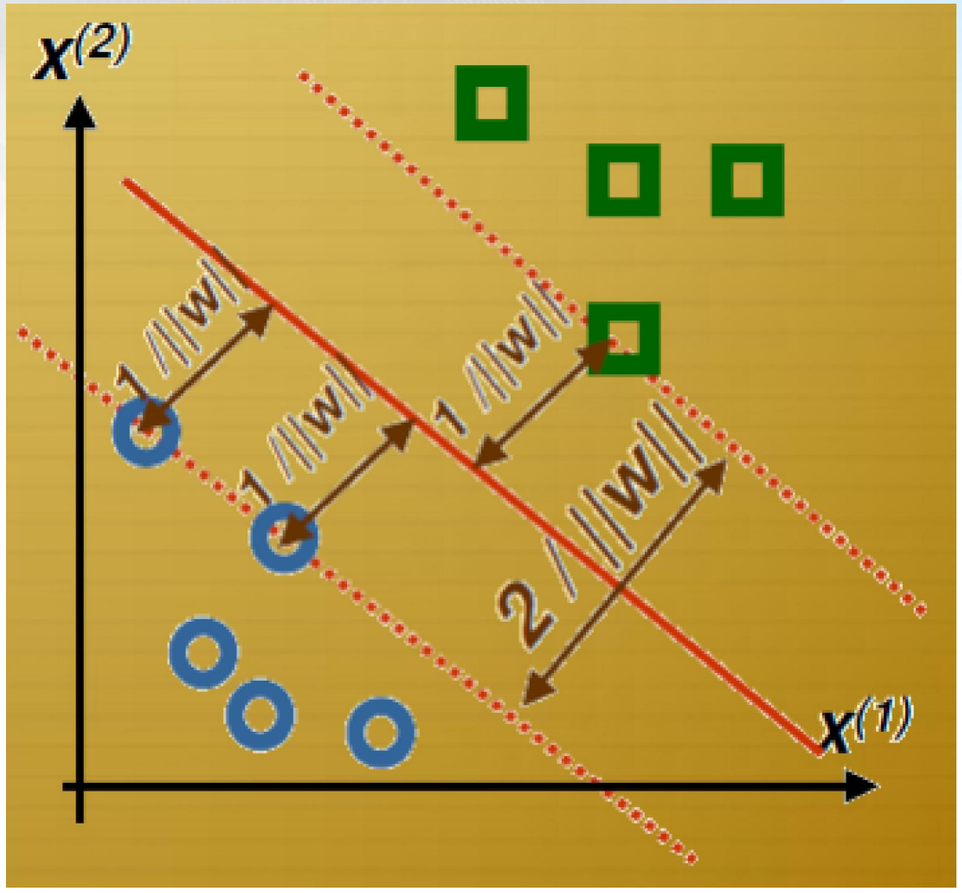
Introduction

- Formula for the Margin.
  - Now distance from closest sample x_i to g(x) = 0 is

$$\frac{\left|\mathbf{w}^t\mathbf{x}_i + w_0\right|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

  - Thus the margin is $m = \frac{2}{\|\mathbf{w}\|}$
  - Maximize margin $m = \frac{2}{\|\mathbf{w}\|}$ subject to constraints

$$\begin{cases} \mathbf{w}^t\mathbf{x}_i + w_0 \geq 1 \text{ if } \mathbf{x}_i \text{ is positive example} \\ \mathbf{w}^t\mathbf{x}_i + w_0 \leq -1 \text{ if } \mathbf{x}_i \text{ is negative example} \end{cases}$$



Introduction

- Let's label
  - $z_i = 1$ if x_i is positive example
  - $z_i = -1$ if x_i is negative example
- Convert to a *Constrained Optimization Problem*

$$\text{minimize} \quad J(\mathrm{w}) = \frac{1}{2}\|\mathrm{w}\|^2$$
$$\text{constrained to} \quad z_i\left(\mathrm{w}^t \mathrm{x}_i + w_0\right) \geq 1 \quad \forall i$$

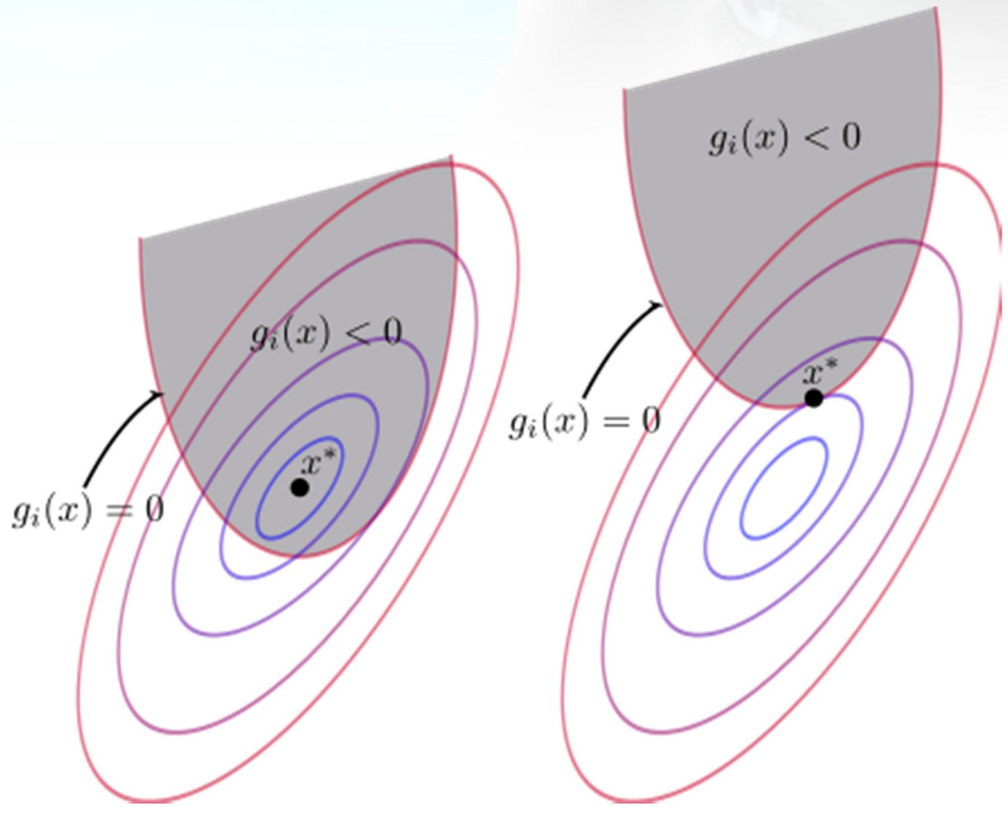
- J(w) is a quadratic function, thus there is a single global minimum and no local minima
- Properties:
  - Convex optimization.
  - Unique solution for linearly separable sample.
- To solve this problem, we will use classical Lagrangian optimization techniques together with the Kuhn-Tucker Theorem

$$\begin{cases} Min\, f(\underline{u}) = \underline{u}^T A\, \underline{u} \\ s.t. \sum_{j=1}^{n} u_j = 0, \sum_{i=1}^{n} u_i^2 = 1 \end{cases}$$

Applying Lagrangien multiplier method:

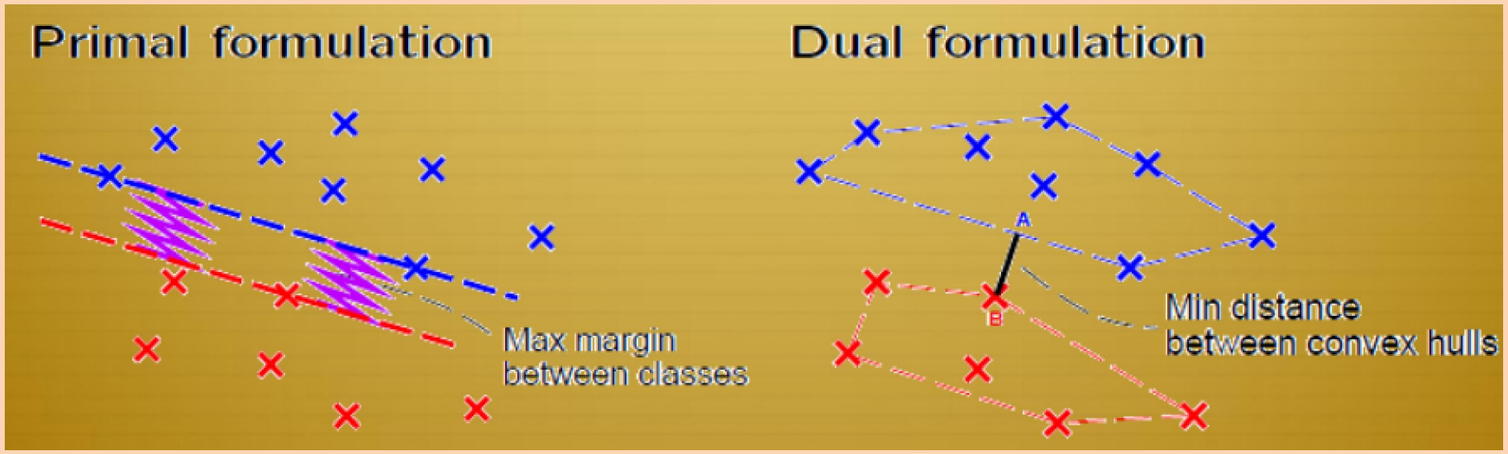
$$L(u,\lambda) = \sum_{i,j} A_{ij} u_i u_j - \lambda_1 \left(\sum_i u_i^2 - 1 \right) - \lambda_2 \left(\sum_j u_j \right)$$

Kuhn-Tucker Theorem?



Kuhn-Tucker Theorem

- Refined theory for convex optimization under constraints.
- Construct a dual optimization problem
  - whose constraints are simpler,
  - and whose solution is related to the solution we seek.



Primal Problem in Algebraic Form	Dual Problem in Algebraic Form
Max $Z = 3x_1 + 5x_2,$	**Min** $W = 4y_1 + 12y_2 + 18y_3,$
s.t. $x_1 \le 4$	**s.t.** $y_1 + 3y_3 \ge 3$
$2x_2 \le 12$	$2y_2 + 2y_3 \ge 5$
$3x_1 + 2x_2 \le 18$	$y_1 \ge 0, y_2 \ge 0, y_3 \ge 0$
$x_1 \ge 0, x_2 \ge 0$	

Kuhn-Tucker Theorem

- Given an optimization problem with convex domain $\Omega \subseteq R^N$:

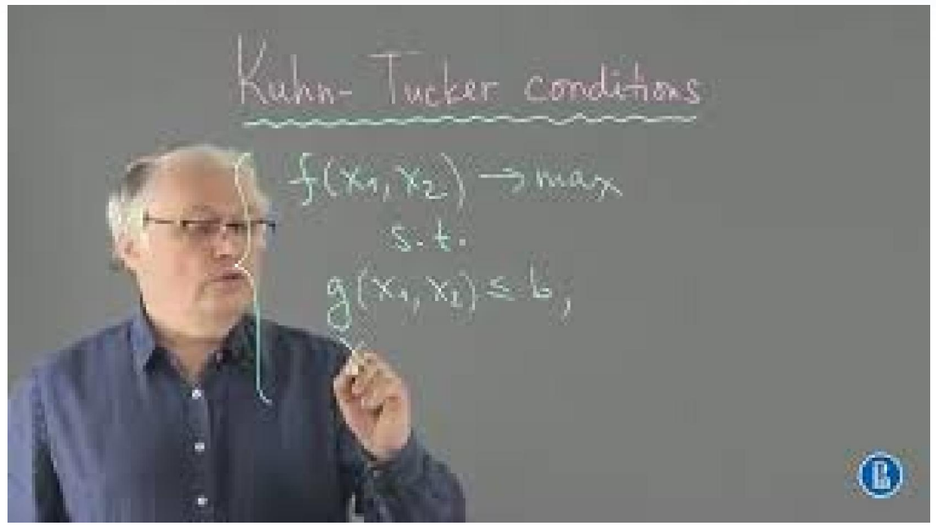
$$\begin{aligned} &\text{minimize} && f(z) && z \in \Omega \\ &\text{constrained to} && g_i(z) \le 0 && i = 1..k \\ &&& h_i(z) = 0 && i = 1..m \end{aligned}$$

- with f ∈ C^1 convex and g_i, h_i affine, necessary & sufficient conditions for a normal point $z*$ to be an optimum are the existence of α∗, β∗ such that

$$\begin{aligned} &\partial L(z^*, \alpha^*, \beta^*)/\partial z = 0 \\ &\partial L(z^*, \alpha^*, \beta^*)/\partial \beta = 0 \\ &\alpha_i^* g_i(z^*) = 0 \quad i = 1..k \\ &g_i(z^*) \le 0 \quad i = 1..k \\ &\alpha_i^* \ge 0 \quad i = 1..k \end{aligned}$$

where:

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



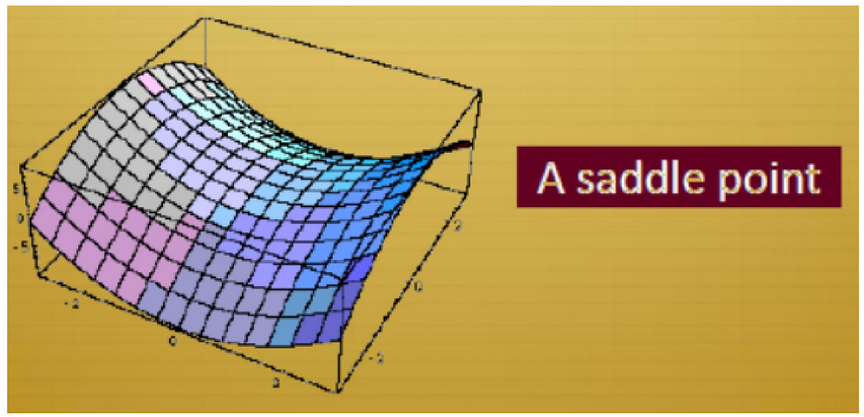
Kuhn-Tucker Theorem

- $L(z, \alpha, \beta)$ is known as a generalized Lagrangian function
- The third condition is known as the Karush-Kuhn-Tucker (KKT) complementary condition
  - It implies that for active constraints $\alpha_i \geq 0$; and for inactive constraints $\alpha_i = 0$
  - As we will see in a minute, the KKT condition allows us to identify the training examples that define the largest margin hyperplane. These examples will be known as Support Vectors
- The Lagrangian Primal Problem
  - Constrained minimization of $J(w) = ½||w||^2$ is solved by introducing the Lagrangian

$$L_p(\mathrm{w}, w_0, \alpha) = \frac{1}{2}\|\mathrm{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left[z_i \left(\mathrm{w}^t \mathrm{x}_i + w_0 \right) - 1 \right]$$

minimizing L_P w.r .t. the primal variables w and w_0, and

maximizing LP w.r .t. the dual variables $\alpha_i \geq 0$ (the Lagrange multipliers)



Kuhn-Tucker Theorem

- Solution
- To simplify the primal problem, we eliminate the primal variables (w,w_0) using the first Kuhn-Tucker condition : $\partial J/\partial z$=0
- Differentiating LP(w,w_0,α) with respect to w and w_0, and setting to zero yields

$$fL_P(w,w_0,\alpha)/fw=0 \qquad w=\sum_{i=1}^{N}\alpha_i z_i x_i$$

$$fL_P(w,w_0,\alpha)/fw_0=0 \qquad \sum_{i=1}^{N}\alpha_i z_i=0$$

- Expansion of L_P yields

$$L_p(\mathrm{w},w_0,\alpha)=\frac{1}{2}\mathrm{w}^t\mathrm{w}-\sum_{i=1}^{N}\alpha_i z_i \mathrm{w}^t\mathrm{x}_i - w_0\sum_{i=1}^{N}\alpha_i z_i+\sum_{i=1}^{N}\alpha_i$$

- Using the optimality condition $\partial J/\partial$ = 0, the first term in LP can be expressed as

$$\mathrm{w}^t\mathrm{w}=\mathrm{w}^t\sum_{i=1}^{N}\alpha_i z_i \mathrm{x}_i=\sum_{i=1}^{N}\alpha_i z_i \mathrm{w}^t\mathrm{x}_i$$

$$=\sum_{i=1}^{N}\alpha_i z_i\left(\sum_{j=1}^{N}\alpha_j z_j \mathrm{x}_j\right)\mathrm{x}_i=\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j z_i z_j \mathrm{x}_i^t\mathrm{x}_j$$

Merging these expressions together we obtain

$$L_D(\alpha)=\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j z_i z_j \mathrm{x}_i^t\mathrm{x}_j$$

Subject to the (simpler) constraints

- $\alpha \geq 0$ and
- $\sum_{i=1}^{N}\alpha_i z_i=0$

This is known as the Lagrangian dual problem.

Kuhn-Tucker Theorem

- The Lagrangian Dual Problem
- We have transformed the problem of finding a saddle point for LP(w,w_0) into the easier one of maximizing $L_D(\alpha)$
  - Notice that $L_D(\alpha)$ depends on the Lagrange multipliers , not on (w,w_0)
- The primal problem scales with dimensionality (has one coefficient for each dimension), whereas the dual problem scales with the amount of training data (there is one Lagrange multiplier per example)
- Moreover, in LD() training data appears only as dot products $(x_i)^T x_j$
- $L_D(\alpha)$ can be optimized by quadratic programming
- $L_D(\alpha)$ formulated in terms of α
  - it depends on w and w0 indirectly
- $L_D(\alpha)$ depends on the number of samples, not on dimension of samples

$$\text{maximize} \quad L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j \mathbf{x}_i^t \mathbf{x}_j$$

$$\text{constrained to} \quad \alpha_i \geq 0 \quad \forall i \quad \textit{and} \quad \sum_{i=1}^{n} \alpha_i z_i = 0$$

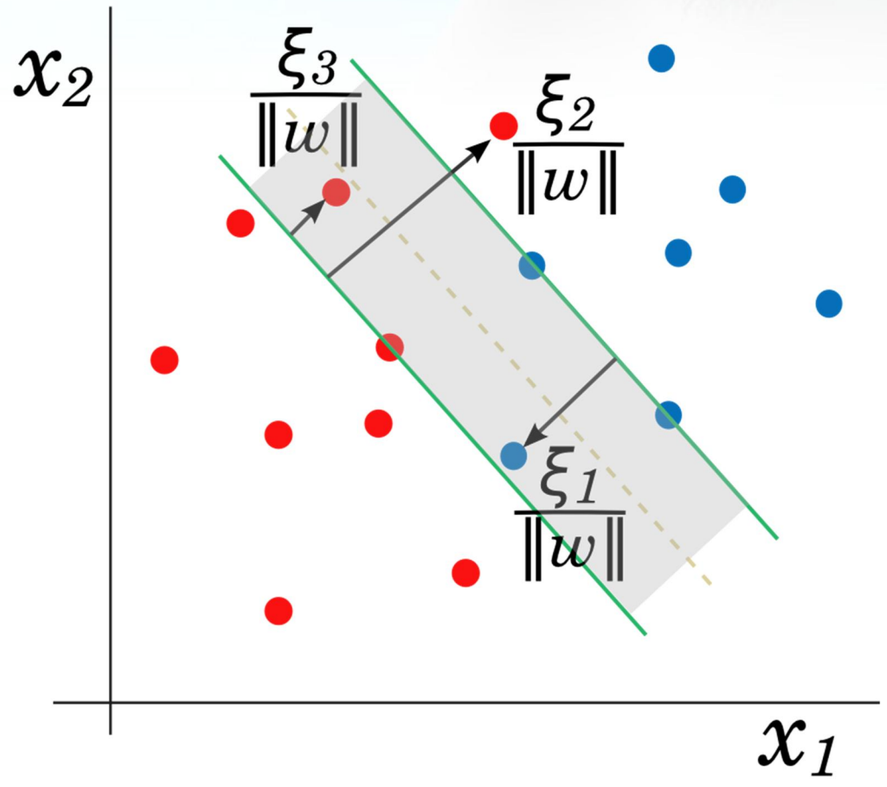
Can rewrite $\boldsymbol{L}_D(\alpha)$ using n by n matrix $\boldsymbol{H}$:

$$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}^t H \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}$$

where the value in the i^{th} row and j^{th} column of $\boldsymbol{H}$ is

$$H = z_i z_j \mathbf{x}_i^t \mathbf{x}_j$$

Linear SVM: Linear Non-Separable Case ?

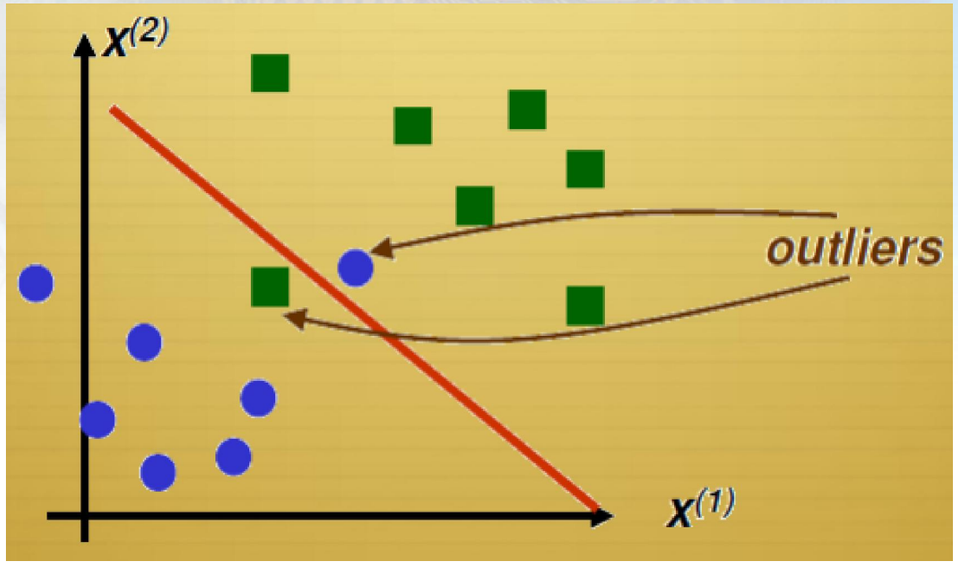


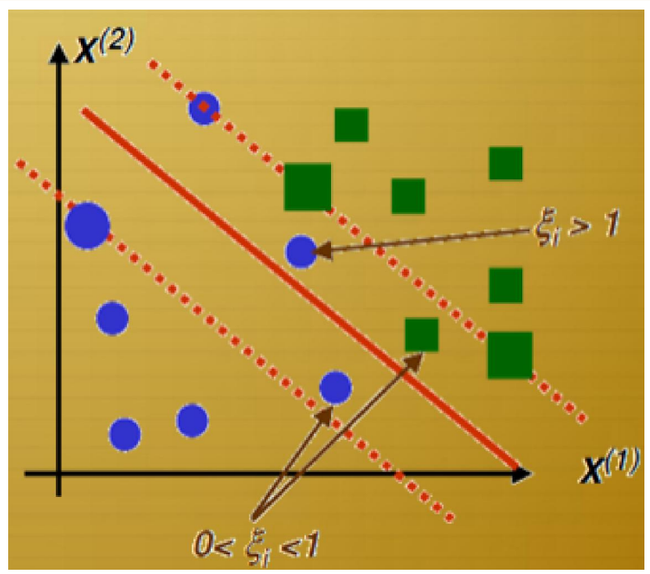
Linear SVM: Linear Non-Separable Case

- Data is most likely to be not linearly separable, but linear classifier may still be appropriate
- Can apply SVM in non linearly separable case
  - data should be "almost" linearly separable for good performance
- Use slack variables $\xi_1, \ldots, \xi_n$ (one for each sample)
- Change constraints from $z_i(w^t x_i + w_0) \geq 1$ for every I to

$$z_i(w^t x_i + w_0) \geq 1 - \xi_i$$

- ξ_i is a measure of deviation from the ideal for sample I
  - $\xi_i > 1$ sample i is on the wrong side of the separating hyperplane
  - $0 < \xi_i < 1$ sample i is on the right side of separating hyperplane but within the region of maximum margin
  - $\xi_i < 0$ is the ideal case for sample i



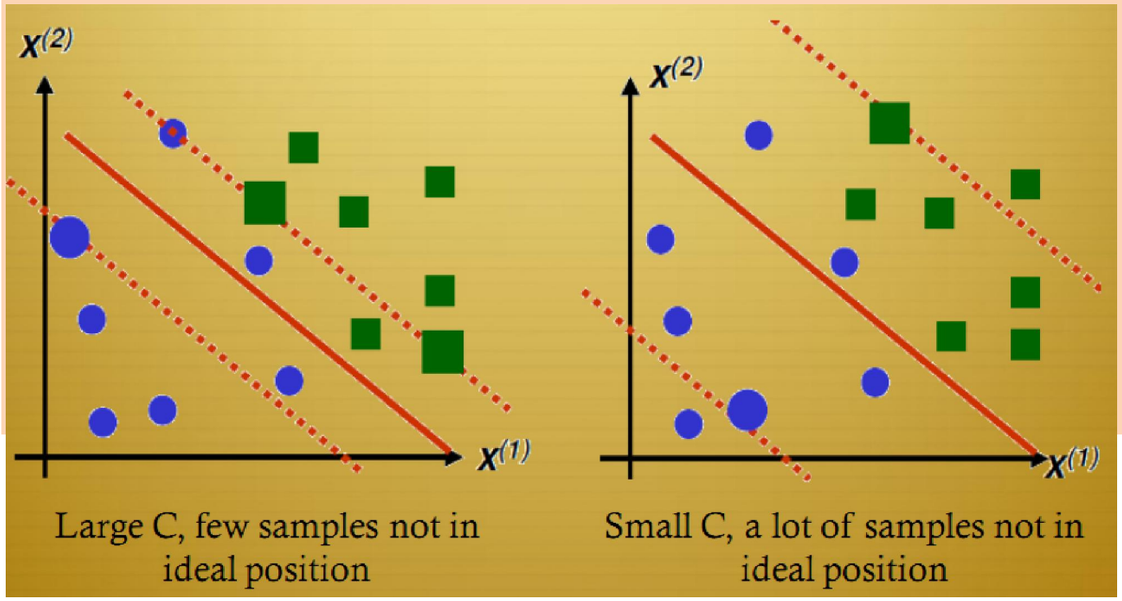


Linear SVM: Linear Non-Separable Case

- Would like to minimize

$$J(\mathrm{w}, \xi_1, \ldots, \xi_n) = \frac{1}{2}\|\mathrm{w}\|^2 + C\sum_{i=1}^{n} \xi_i$$

- constrained to $z_i(w^t x_i + w_0) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for every I
- C is a constant which controls the tradeoff between margin and errors
  - if C is small, we allow a lot of samples not in ideal position
  - if C is large, we want to have very few samples not in ideal position



Large C, few samples not in ideal position

Small C, a lot of samples not in ideal position

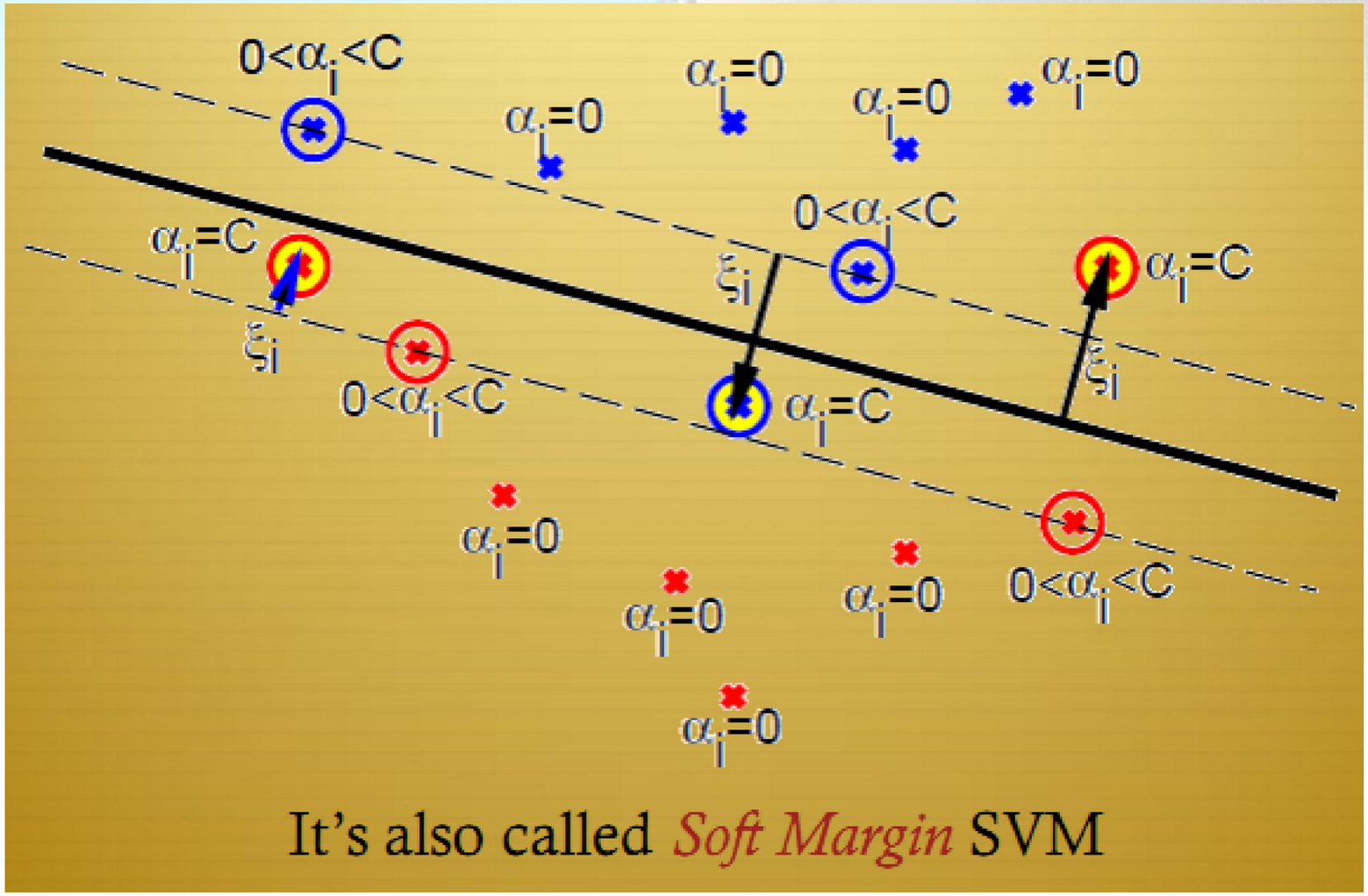
Can use Kuhn-Tucker theorem to converted to

$$\text{maximize} \quad L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j x_i^t x_j$$

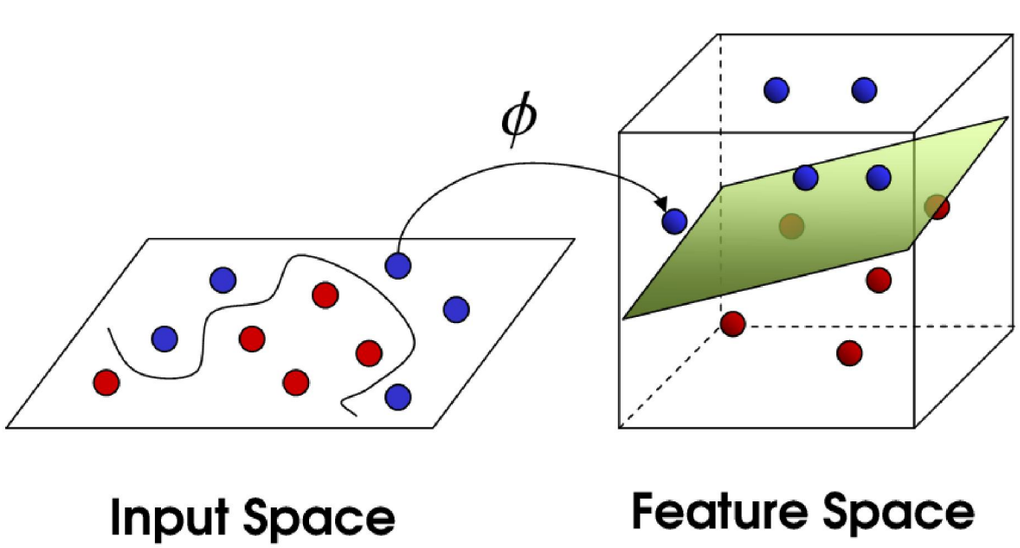
$$\text{constrained to} \quad 0 \leq \alpha_i \leq C \quad \forall i \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i z_i = 0$$

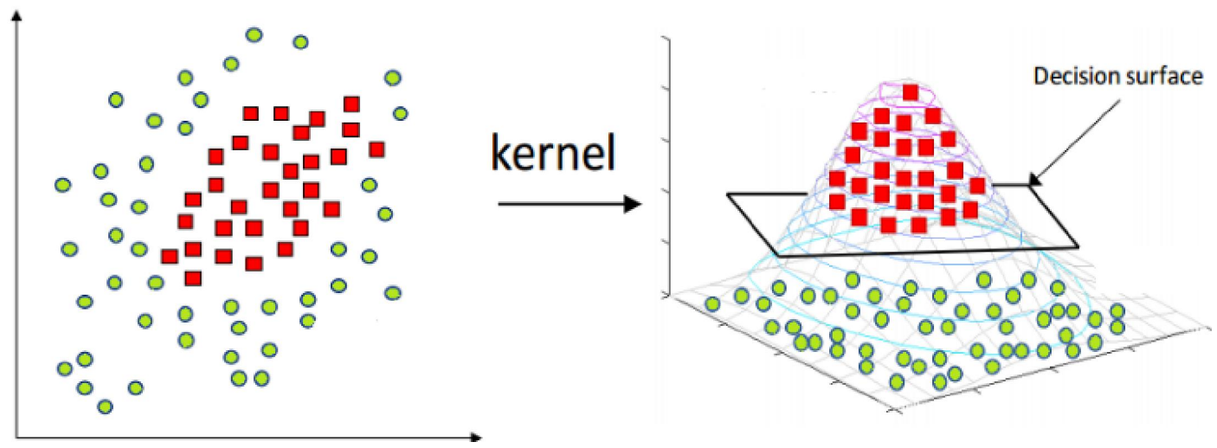
- find **w** using $\mathrm{w} = \sum_{i=1}^{n} \alpha_i z_i x_i$
- solve for w_0 using any $0 < \alpha_i < C$ and $\alpha_i[\mathbf{z}_i(\mathbf{w}^t\mathbf{x}_i + w_0) - 1)] = 0$

Linear SVM: Linear Non-Separable Case



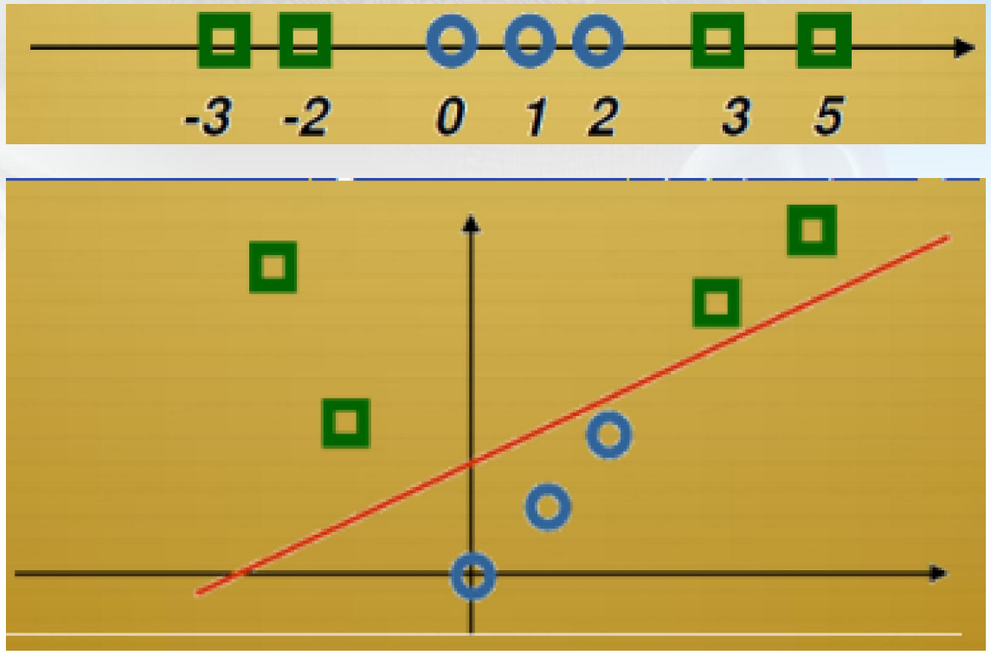
Non-linear SVM & Kernel Trick?

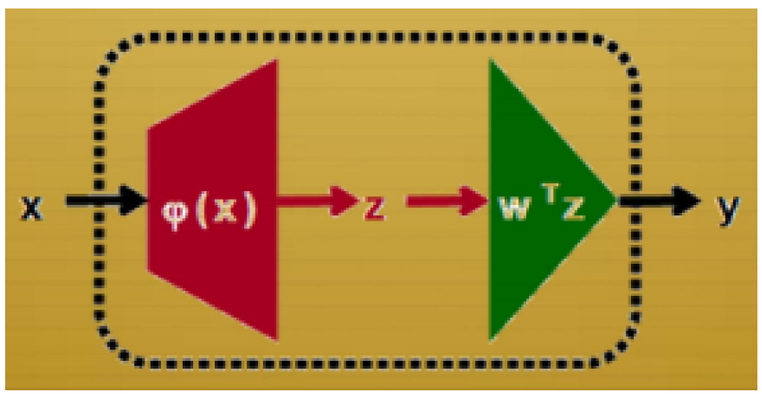




Non-linear SVM & Kernel Trick?

- Cover's theorem:
- "A complex pattern-classification problem cast in a high-dimensional space non-linearly is more likely to be linearly separable than in a low-dimensional space "
- One dimensional space, not linearly separable
- Lift to two dimensional space with $\varphi(x)=(x,x^2)$
- Can use any linear classifier after lifting data into a higher dimensional space → SVM helps avoid the "curse of dimensionality" problems.
- SVMs operate in two stages
  - Perform a non-linear mapping of the feature vector x onto a high-dimensional space that is hidden from the inputs or the outputs
  - Construct an optimal separating hyperplane in the high-dim space





Non-linear SVM & Kernel Trick?

- Recall SVM optimization

$$\text{maximize} \quad L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j \mathrm{x}_i^{\mathrm{t}} \mathrm{x}_j$$

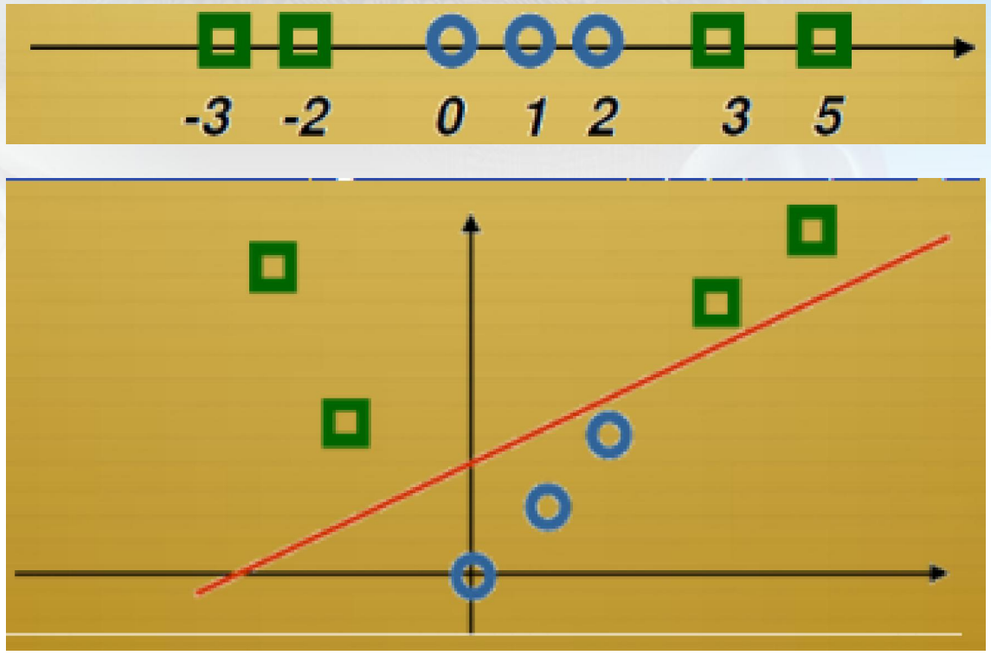
- Note this optimization depends on samples x_i only through the dot product $(x_i)^t x_j$
- If we lift xi to high dimension using φ(x), need to compute high dimensional product $\varphi(x_i)^t \varphi(x_j)$

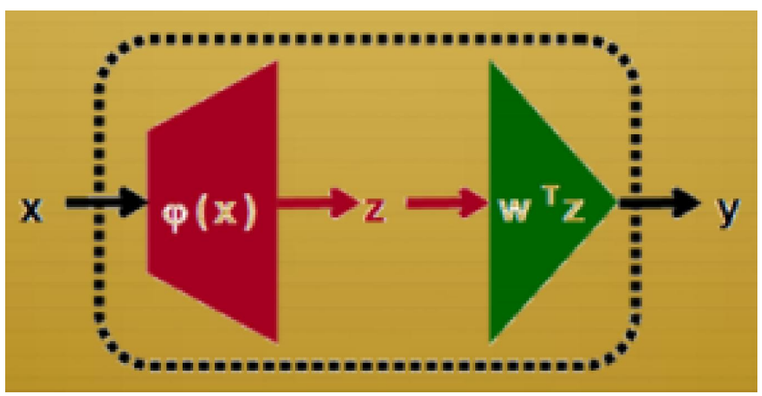
$$\text{maximize} \quad L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j \boxed{\varphi(\mathrm{x}_i)^{\mathrm{t}} \varphi(\mathrm{x}_j)}$$

- Idea: find kernel function in SVM terminology K(xi,xj) s.t.

$$K(\mathrm{x}_i, \mathrm{x}_j) = \varphi(\mathrm{x}_i)^{\mathrm{t}} \varphi(\mathrm{x}_j)$$

  - Recall that the SVM solution depends only on the dot product $\varphi(x_i)^t \varphi(x_j)$ between training examples.





Non-linear SVM & Kernel Trick?

- Therefore, we only need to compute K(xi,xj) instead of $\varphi(x_i)^t\varphi(x_j)$
  - "kernel trick": do not need to perform operations in high dimensional space explicitly
- How to choose kernel function K(xi,xj)?
  - K(xi,xj) should correspond to product $\varphi(x_i)^t\varphi(x_j)$ in a higher dimensional space
  - Mercer's condition tells us which kernel function can be expressed as dot product of two vectors

- *Polynomial kernel*

$$K(\mathbf{x}_i,\mathbf{x}_j)=(\mathbf{x}_i^t\mathbf{x}_j+1)^p$$

- *Gaussian radial Basis kernel* (data is lifted in infinite dimension)

$$K(\mathbf{x}_i,\mathbf{x}_j)=\exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}_i-\mathbf{x}_j\|^2\right)$$

$$\text{maximize}\quad L_D(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j z_i z_j\varphi(\mathbf{x}_i)^t\varphi(\mathbf{x}_j)$$

$$\text{constrained to}\quad 0\le\alpha_i\le\beta\quad\forall i\quad and\quad\sum_{i=1}^{n}\alpha_i z_i=0$$

Non linear discriminant function:

$$g(\mathbf{x})=\sum_{\mathbf{x}_i\in S}\alpha_i\, z_i\, K(\mathbf{x}_i,\mathbf{x})$$

$g(\mathbf{x}) = \sum$ (weight of support vector $\mathbf{x}_i$) ± 1 ("inverse distance" from $\mathbf{x}$ to support vector $\mathbf{x}_i$)

most important training samples, i.e. support vectors

$$K(\mathbf{x}_i,\mathbf{x})=\exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}_i-\mathbf{x}\|^2\right)$$


Thank you!
grammarly