SUPPORT VECTOR MACHINES

Prof. G. Panda

IIT Bhubaneswar

- Support Vector Machines (SVM)
  - Introduction
  - Linear Discriminant
    - Linearly Separable Case
    - Linearly Non Separable Case
  - Kernel Trick
    - Non Linear Discriminant

SVM

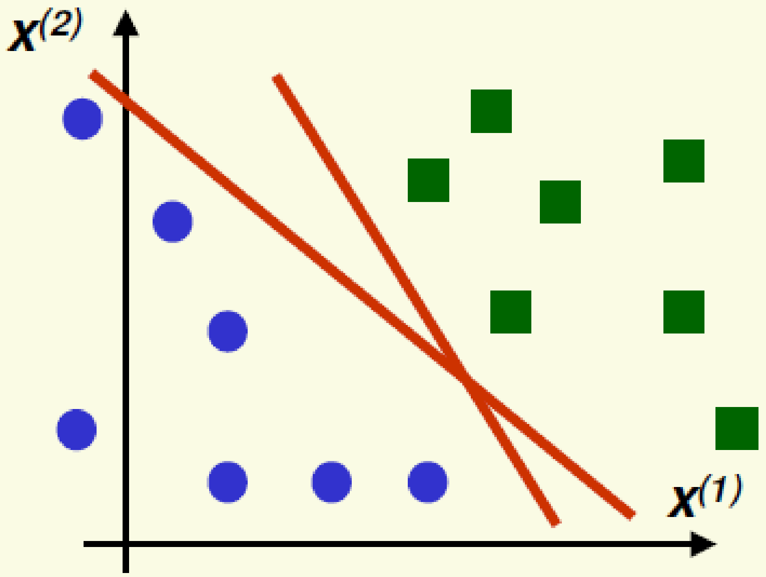
- Said to start in 1979 with Vladimir Vapnik's paper
- Major developments throughout 1990's
- Elegant theory
  - Has good generalization properties
- Have been applied to diverse problems very successfully in the last 10-15 years
- One of the most important developments in pattern recognition in the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

Linear Discriminant Functions

- A discriminant function is linear if it can be written as

$$g(x) = w^t x + w_0$$

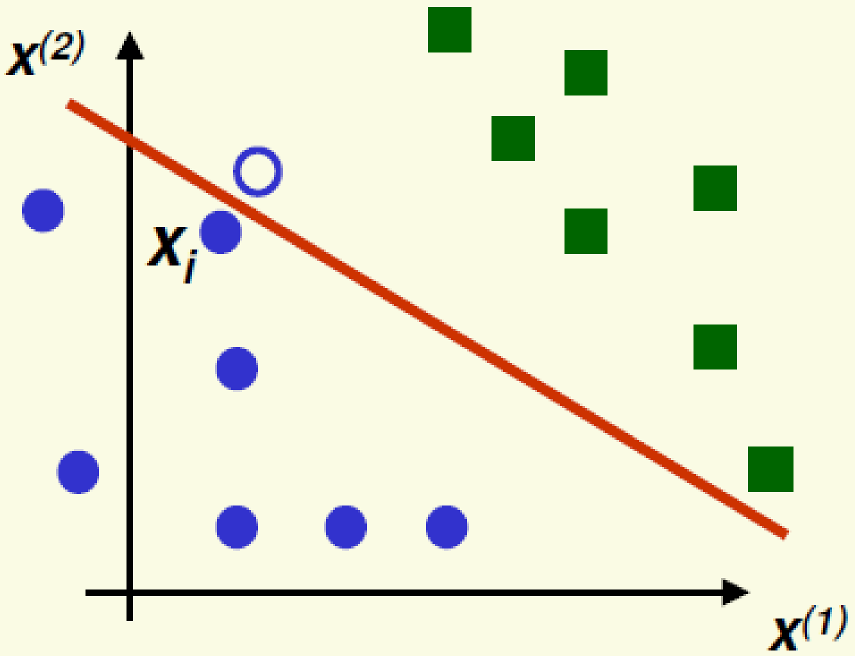
$$\boxed{\begin{array}{ll} g(x) > 0 & \Rightarrow x \in class\ 1 \\ g(x) < 0 & \Rightarrow x \in class\ 2 \end{array}}$$



- which separating hyperplane should we choose?

Linear Discriminant Functions

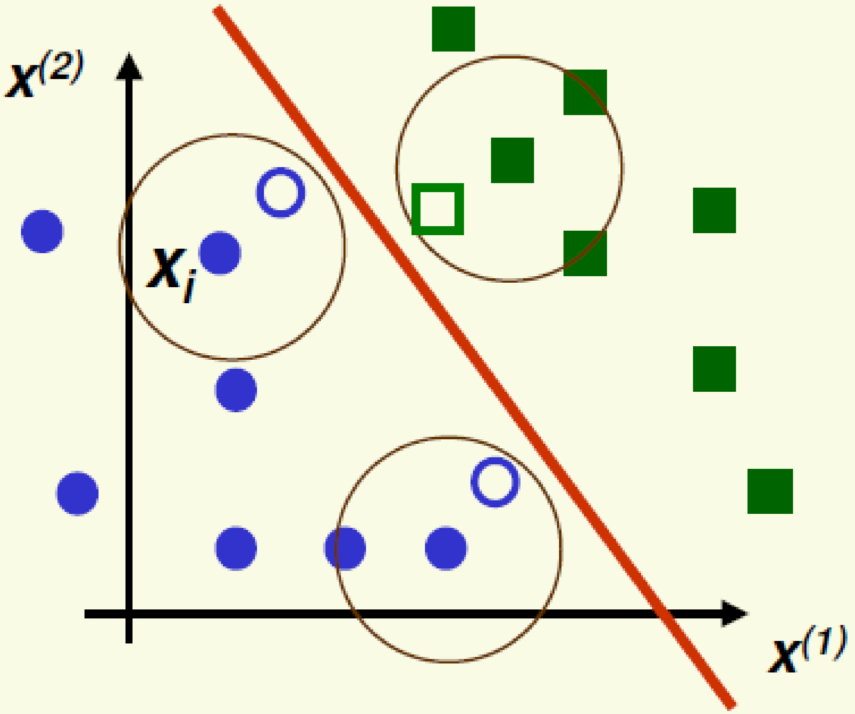
- Training data is just a subset of of all possible data
- Suppose hyperplane is close to sample x_i
- If we see new sample close to sample ***i***, it is likely to be on the wrong side of the hyperplane



- Poor generalization (performance on unseen data)

Linear Discriminant Functions

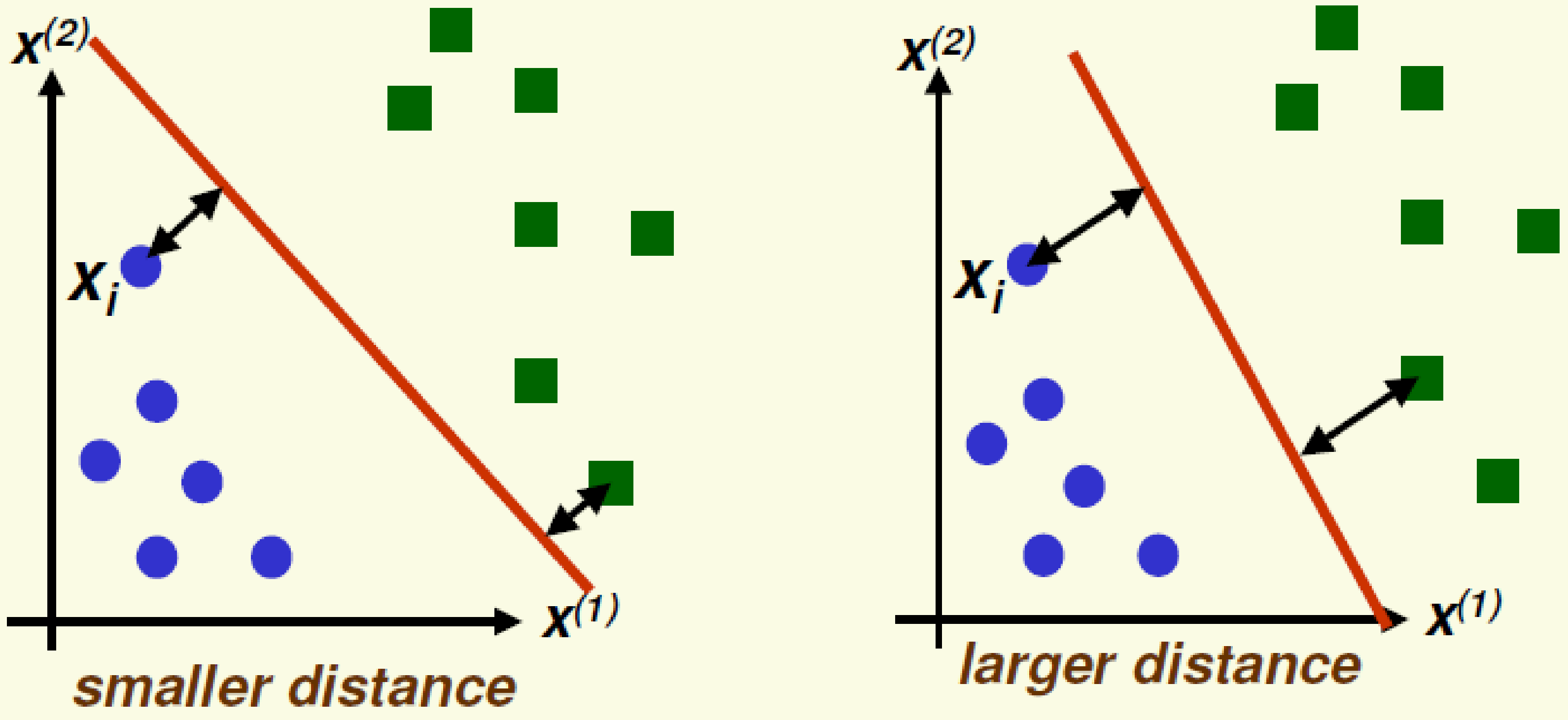
- Hyperplane as far as possible from any sample



- New samples close to the old samples will be classified correctly
- Good generalization

SVM

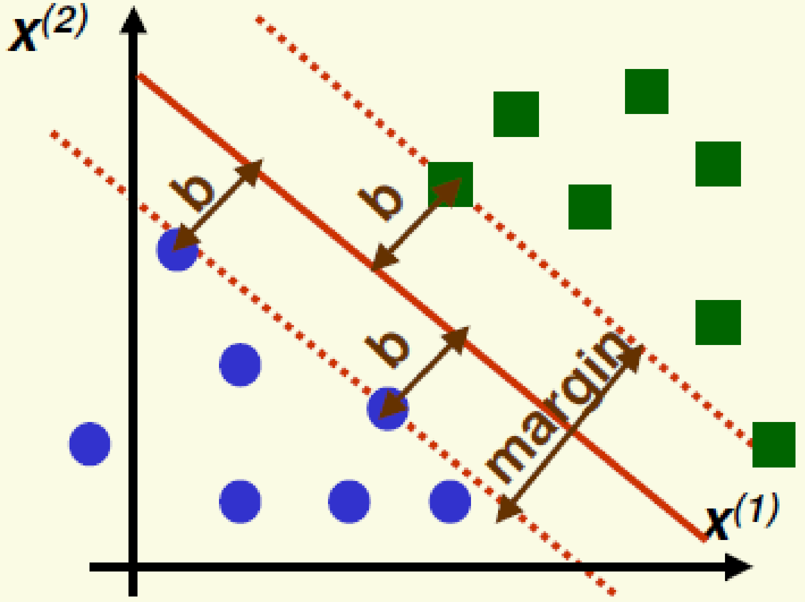
- Idea: maximize distance to the closest example



- For the optimal hyperplane
  - distance to the closest negative example = distance to the closest positive example

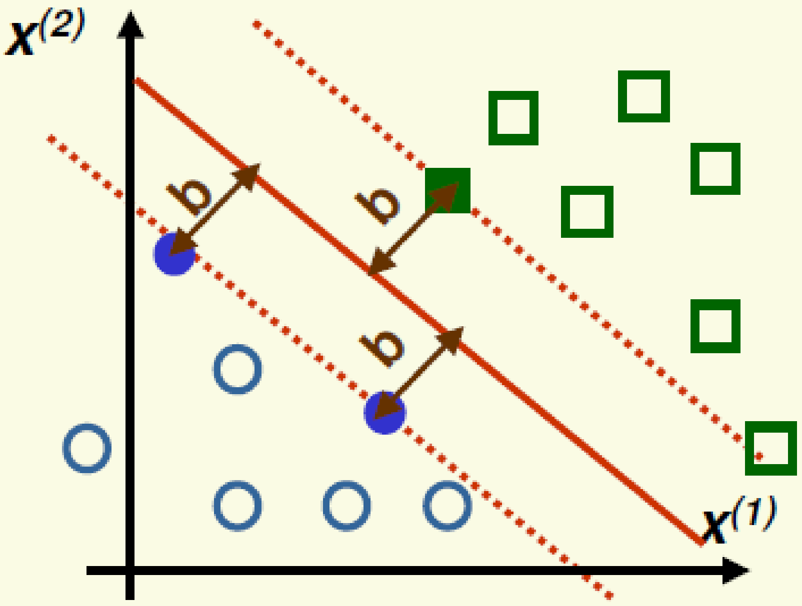
SVM: Linearly Separable Case

- SVM: maximize the ***margin***



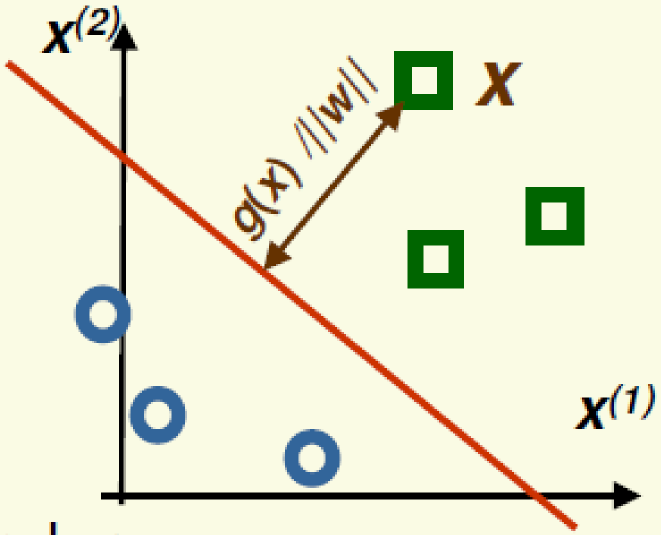
- ***margin*** is twice the absolute value of distance ***b*** of the closest example to the separating hyperplane
- Better generalization (performance on test data)
  - in practice
  - and in theory

SVM: Linearly Separable Case



- ***Support vectors*** are the samples closest to the separating hyperplane
  - they are the most difficalt patterns to classify
  - Optimal hyperplane is completely defined by support vectors
    - of course, we do not know which samples are support vectors without finding the optimal hyperplane

SVM: 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 $g_1(x)=\alpha g(x)$

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

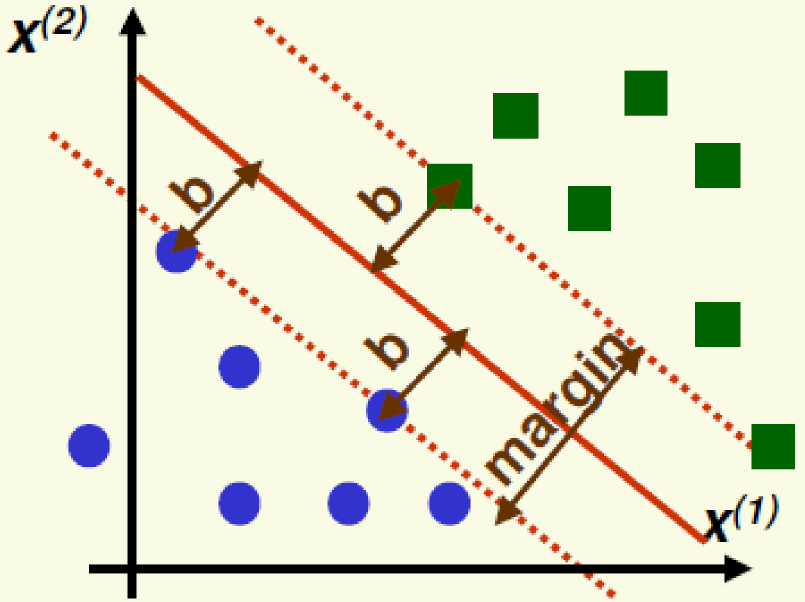
- Let x_i be an example closest to the boundary. Set

$$|w^t x_i + w_0| = 1$$

- Now the largest margin hyperplane is unique

SVM: Linearly Separable Case

- SVM: 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 practice
  - and in theory

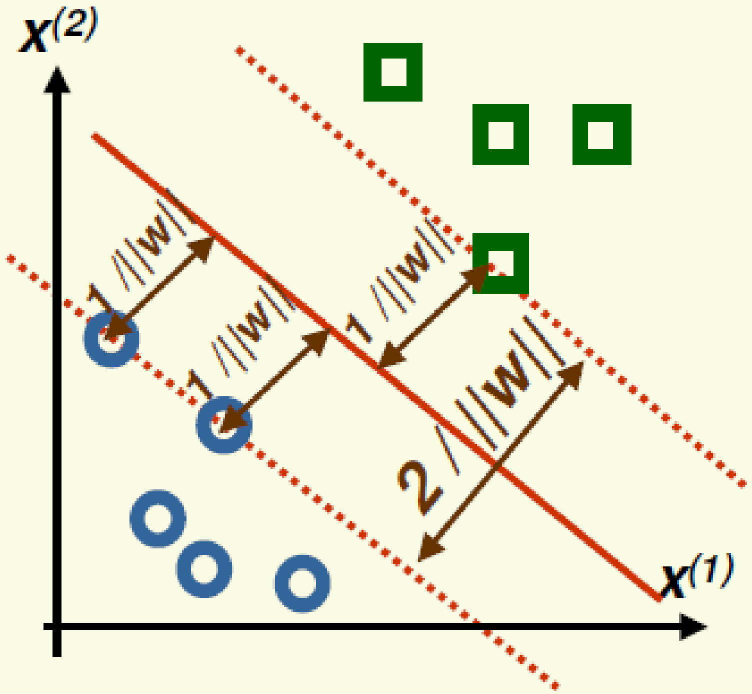
SVM: Formula for the Margin

- For uniqueness, set $|w^t x_i + w_0| = 1$ for any example x_i closest to the boundary
- now distance from closest sample x_i to $g(x) = 0$ is

$$\frac{|w^t x_i + w_0|}{\|w\|} = \frac{1}{\|w\|}$$

- Thus the margin is

$$\boxed{m = \frac{2}{\|w\|}}$$



SVM: Optimal Hyperplane

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

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

- Let $\begin{cases} z_i = 1 & \text{if } x_i \text{ is positive example} \\ z_i = -1 & \text{if } x_i \text{ is negative example} \end{cases}$

- Can convert our problem to

$$\boxed{\begin{array}{ll} \text{minimize} & J(w) = \frac{1}{2}\|w\|^2 \\ \text{constrained to} & z_i\left(w^t x_i + w_0\right) \geq 1 \quad \forall i \end{array}}$$

- $J(w)$ is a quadratic function, thus there is a single global minimum

SVM: Optimal Hyperplane

- Use Kuhn-Tucker theorem to convert our problem to:

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

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

- $\alpha = \{\alpha_1, \ldots, \alpha_n\}$ are new variables, one for each sample
- Can rewrite $L_D(\alpha)$ using n by n matrix 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 ith row and jth column of H is

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

SVM: Optimal Hyperplane

- Use Kuhn-Tucker theorem to convert our problem to:

$$\boxed{\begin{array}{ll} \text{maximize} & L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j x_i^t x_j \\ \text{constrained to} & \alpha_i \geq 0 \;\; \forall i \;\; \textit{and} \;\; \sum_{i=1}^{n} \alpha_i z_i = 0 \end{array}}$$

- $\alpha = \{\alpha_1, \ldots, \alpha_n\}$ are new variables, one for each sample
- $L_D(\alpha)$ can be optimized by quadratic programming
- $L_D(\alpha)$ formulated in terms of α
  - it depends on w and w_0 indirectly

SVM: Optimal Hyperplane

- After finding the optimal $\alpha = \{\alpha_1, \ldots, \alpha_n\}$
  - For every sample ***i***, one of the following must hold
    - $\alpha_i = 0$ (sample ***i*** is not a support vector)
    - $\alpha_i \neq 0$ and $z_i(w^t x_i + w_0 - 1) = 0$ (sample ***i*** is support vector)
  - can find ***w*** using $w = \sum_{i=1}^{n} \alpha_i z_i x_i$
  - can solve for w_0 using any $\alpha_i > 0$ and $\alpha_i [z_i (w^t x_i + w_0) - 1] = 0$

$$w_0 = \frac{1}{z_i} - w^t x_i$$

- Final discriminant function:

$$g(x) = \left(\sum_{x_i \in S} \alpha_i z_i x_i \right)^t x + w_0$$

  - where ***S*** is the set of support vectors

$$S = \{x_i \mid \alpha_i \neq 0\}$$

SVM: Optimal Hyperplane

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

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

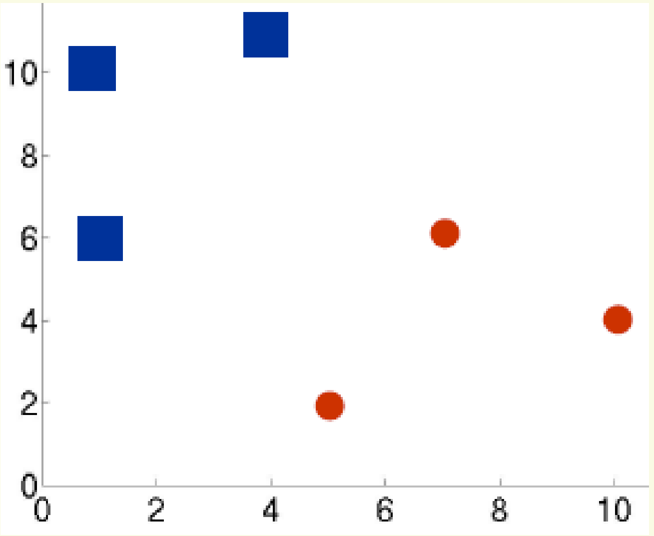
- $L_D(\alpha)$ depends on the number of samples, not on dimension of samples
- samples appear only through the dot products $\mathbf{x}_i^t \mathbf{x}_j$
- This will become important when looking for a ***nonlinear*** discriminant function, as we will see soon

SVM: Example using Matlab

- Class 1: [1,6], [1,10], [4,11]
- Class 2: [5,2], [7,6], [10,4]
- Let's pile all data into array $\boldsymbol{X}$

$$X = \begin{bmatrix} 1 & 6 \\ 1 & 10 \\ 4 & 11 \\ 5 & 2 \\ 7 & 6 \\ 10 & 4 \end{bmatrix}$$

- Pile $\boldsymbol{z}_i$'s into vector $z = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$



- Matrix $\boldsymbol{H}$ with $\boldsymbol{H}_{ij} = \boldsymbol{z}_i \boldsymbol{z}_j \boldsymbol{x}_i^t \boldsymbol{x}_j$, in matlab use $\boldsymbol{H} = (\boldsymbol{x} * \boldsymbol{x}').*(\boldsymbol{z} * \boldsymbol{z}')$

$$H = \begin{bmatrix} 37 & 61 & 70 & -17 & -43 & -34 \\ 61 & 101 & 114 & -25 & -67 & -50 \\ 70 & 114 & 137 & -42 & -94 & -84 \\ -17 & -25 & -42 & 29 & 47 & 58 \\ -43 & -67 & -94 & 47 & 85 & 94 \\ -34 & -50 & -84 & 58 & 94 & 116 \end{bmatrix}$$

SVM: Example using Matlab

- Matlab expects quadratic programming to be stated in the *canonical* (standard) form which is

$$\boxed{\begin{array}{rl} \text{minimize} & L_D(\alpha) = 0.5\alpha^t H\alpha + f^t\alpha \\ \text{constrained to} & A\alpha \le a \text{ and } B\alpha = b \end{array}}$$

  - where A, B, H are n by n matrices and f, a, b are vectors

- Need to convert our optimization problem to canonical form

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

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

SVM: Example using Matlab

- Multiply by –1 to convert to minimization:

$$\text{minimize} \quad L_D(\alpha) = -\sum_{i=1}^{n} \alpha_i + \frac{1}{2}\alpha^t H \alpha$$

- Let $f = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix} = -\mathbf{ones(6,1)}$, then can write

$$\text{minimize} \quad L_D(\alpha) = f^t \alpha + \frac{1}{2}\alpha^t H \alpha$$

- First constraint is $\alpha_i \geq 0 \quad \forall i$
- Let $A = \begin{bmatrix} -1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -1 \end{bmatrix} = -\mathbf{eye}(6)$, $a = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} = \mathbf{zeros(6,1)}$
- Rewrite the first constraint in canonical form:

$$A\alpha \leq a$$

SVM: Example using Matlab

- Our second constraint is $\sum_{i=1}^{n} \alpha_i z_i = 0$
- Let $B = \begin{bmatrix} z_1 & \cdots & \cdots & z_6 \\ 0 & \cdots & \cdots & 0 \\ \vdots & \cdots & \cdots & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix} = [[z];\ [zeros(5,6)]]$

  and $b = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} = zeros(6,1)$
- Second constraint in canonical form is:

$$B\alpha = b$$

- Thus our problem is in canonical form and can be solved by matlab:

$$\boxed{\begin{array}{rl} \text{minimize} & L_D(\alpha) = 0.5\alpha^t H\alpha + f^t\alpha \\ \text{constrained to} & A\alpha \le a \text{ and } B\alpha = b \end{array}}$$

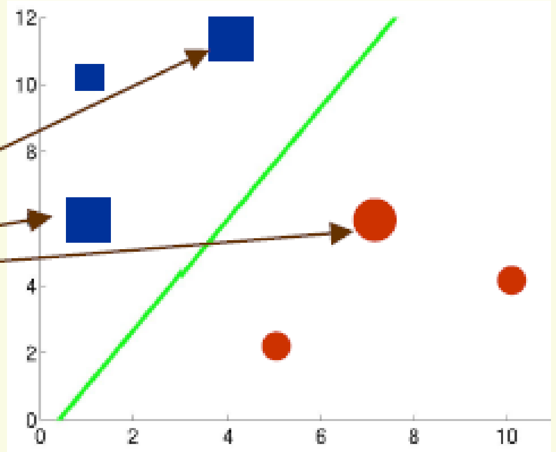
SVM: Example using Matlab

- α = quadprog($\boldsymbol{H}$+eye(6)*0.001, $\boldsymbol{f}$, $\boldsymbol{A}$, $\boldsymbol{a}$, $\boldsymbol{B}$, $\boldsymbol{b}$)

  for stability

- Solution $\alpha = \begin{bmatrix} 0.036 \\ 0 \\ 0.039 \\ 0 \\ 0.076 \\ 0 \end{bmatrix}$

  support vectors



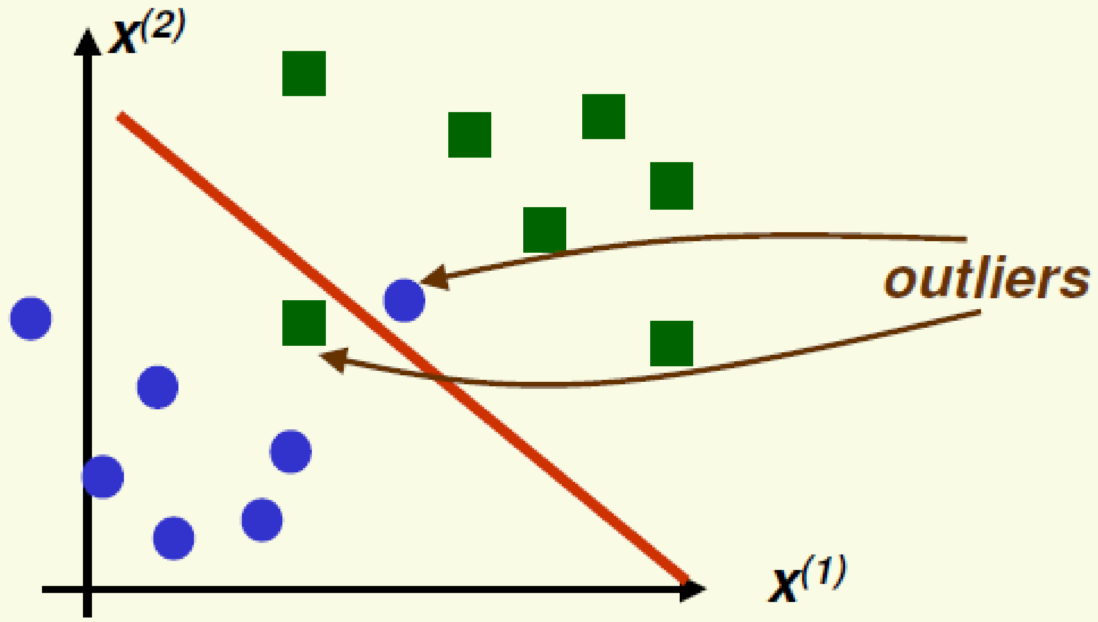
- find $\boldsymbol{w}$ using $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i z_i \boldsymbol{x}_i = (\alpha .* \boldsymbol{z})^t \boldsymbol{x} = \begin{bmatrix} -0.33 \\ 0.20 \end{bmatrix}$

- since $\alpha_1 > 0$, can find $\boldsymbol{w_0}$ using

$$w_0 = \frac{1}{z_1} - w^t x_1 = 0.13$$

SVM: Non Separable Case

- Data is most likely to be not linearly separable, but linear classifier may still be appropriate



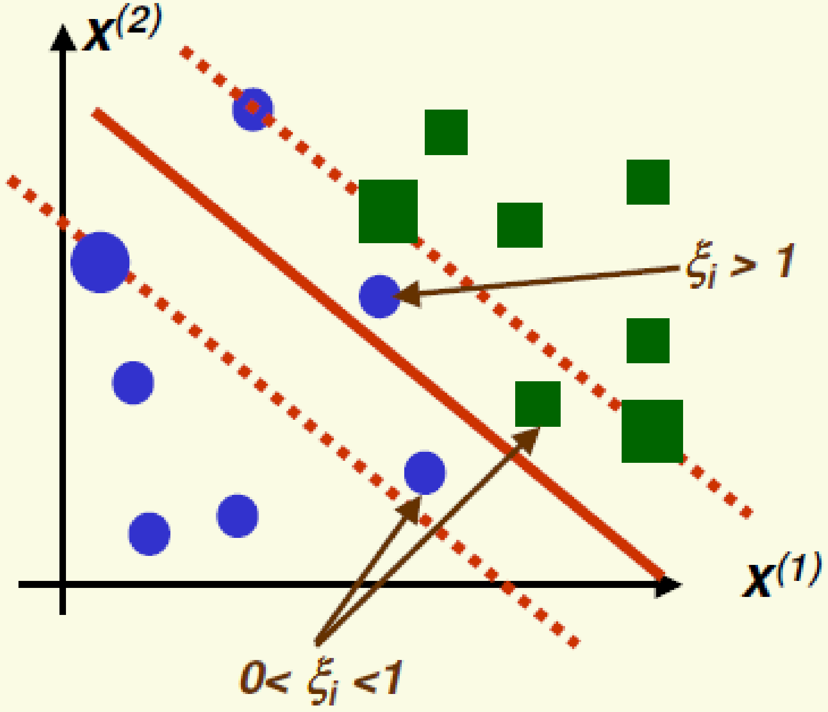
- Can apply SVM in non linearly separable case
  - data should be "almost" linearly separable for good performance

SVM: Non Separable Case

- Use slack variables $\xi_1, \ldots, \xi_n$ (one for each sample)
- Change constraints from $z_i\left(w^t x_i + w_0\right) \geq 1 \quad \forall i$ to

$$z_i\left(w^t x_i + w_0\right) \geq 1 - \xi_i \quad \forall i$$

- 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***



SVM: Non Separable Case

- Would like to minimize

$$J(w,\xi_1,\dots,\xi_n)=\frac{1}{2}\|w\|^2+\beta\sum_{i=1}^{n}I(\xi_i>0)$$

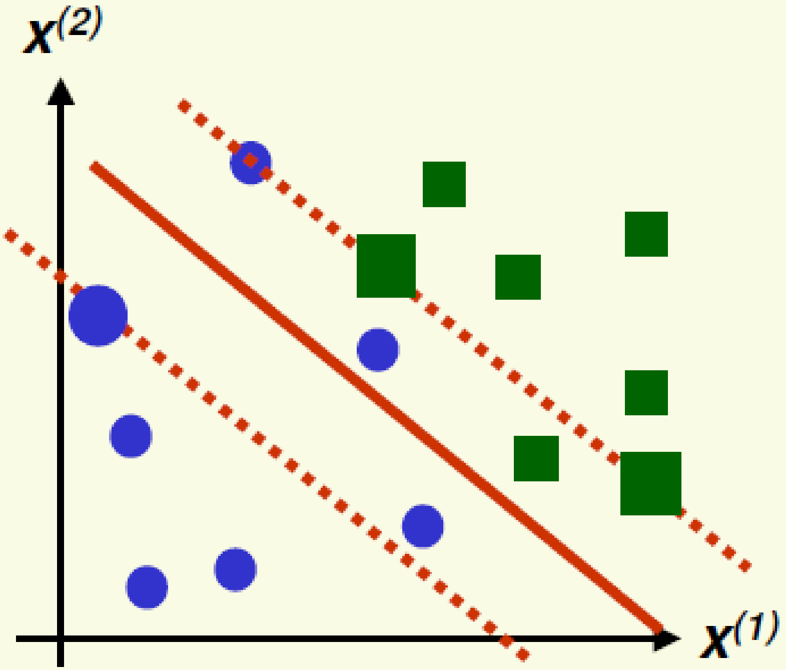
# of samples not in ideal location

  - where $I(\xi_i>0)=\begin{cases}1 & \text{if } \xi_i>0\\ 0 & \text{if } \xi_i\le 0\end{cases}$
  - constrained to $z_i\left(w^t x_i+w_0\right)\ge 1-\xi_i$ and $\xi_i\ge 0 \quad \forall i$
  - β is a constant which measures relative weight of the first and second terms
    - if β is small, we allow a lot of samples not in ideal position
    - if β is large, we want to have very few samples not in ideal positon

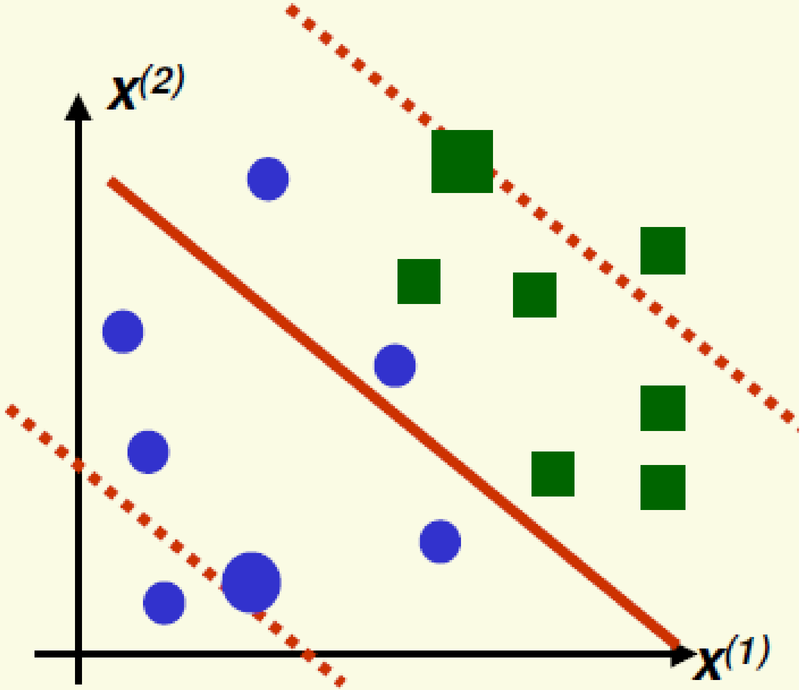
SVM: Non Separable Case

$$J(w,\xi_1,\dots,\xi_n)=\frac{1}{2}\|w\|^2+\beta\sum_{i=1}^{n} I(\xi_i>0)$$

# of examples not in ideal location



large β, few samples not in ideal position



small β, a lot of samples not in ideal position

SVM: Non Separable Case

- Unfortunately this minimization problem is NP-hard due to discontinuity of functions $I(\xi_i)$

$$J(w,\xi_1,\dots,\xi_n)=\frac{1}{2}\|w\|^2+\beta\sum_{i=1}^{n}I(\xi_i>0)$$

# of examples not in ideal location

  - where $I(\xi_i>0)=\begin{cases}1 & \text{if } \xi_i>0\\ 0 & \text{if } \xi_i\le 0\end{cases}$
  - constrained to $z_i\left(w^t x_i+w_0\right)\ge 1-\xi_i$ and $\xi_i\ge 0 \quad \forall i$

SVM: Non Separable Case

- Instead we minimize

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

a measure of # of misclassified examples

  - constrained to $\begin{cases} z_i(w^t x_i + w_0) \geq 1 - \xi_i & \forall i \\ \xi_i \geq 0 & \forall i \end{cases}$

- Can use Kuhn-Tucker theorem to converted to

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

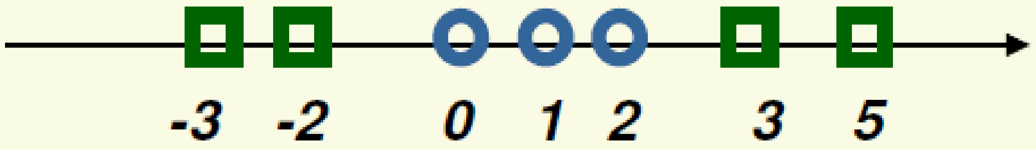
constrained to $0 \leq \alpha_i \leq \beta \quad \forall i \quad$ *and* $\sum_{i=1}^{n}\alpha_i z_i = 0$

  - find ***w*** using $w = \sum_{i=1}^{n}\alpha_i z_i x_i$
  - solve for w_0 using any $0 < \alpha_i < \beta$ and $\alpha_i\left[z_i\left(w^t x_i + w_0\right) - 1\right] = 0$

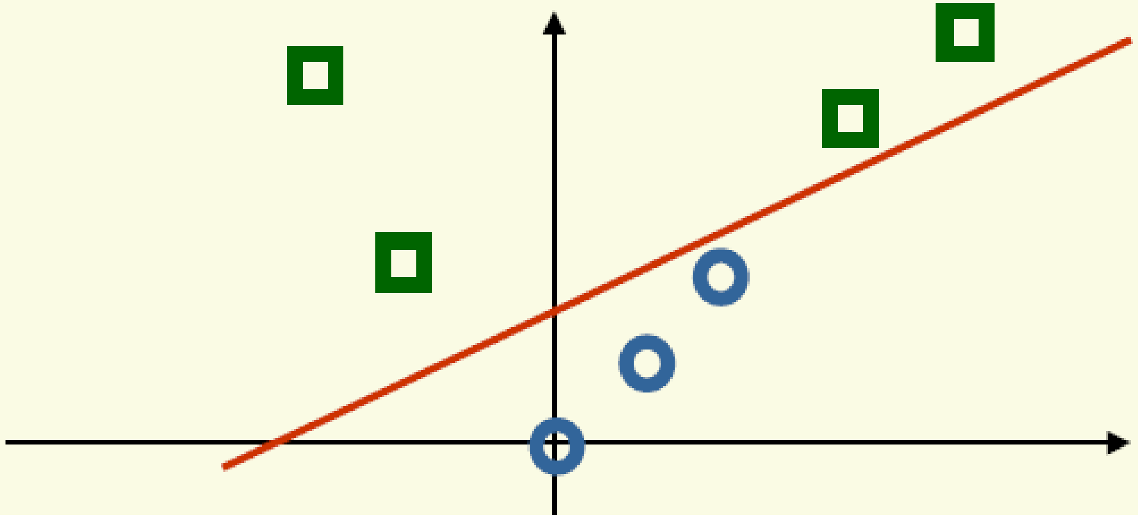
Non Linear Mapping

- Cover's theorem:
  - "*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

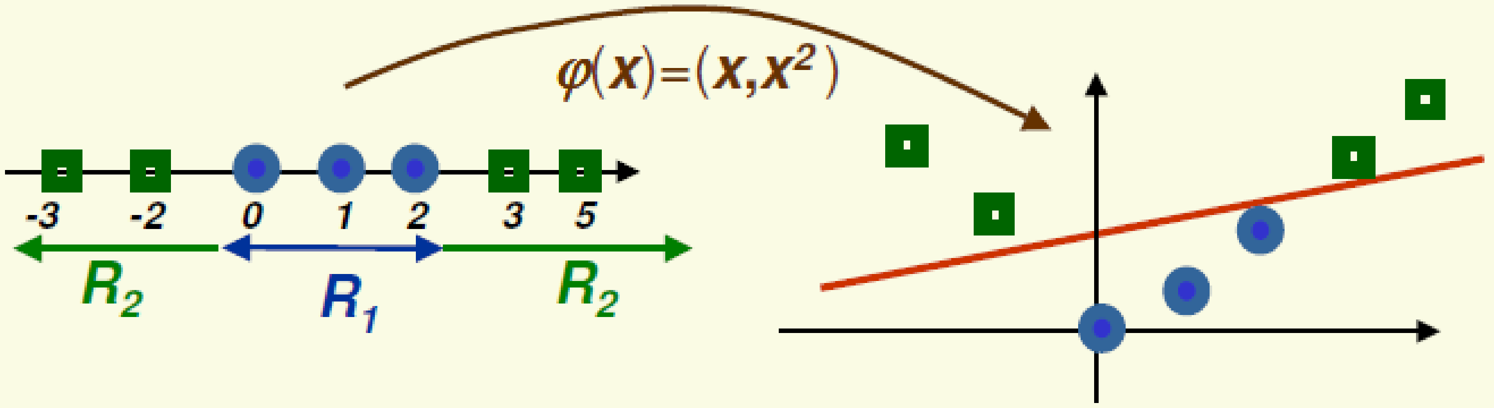


- Lift to two dimensional space with $\varphi(x)=(x,x^2)$



Non Linear Mapping

- To solve a non linear classification problem with a linear classifier
  1. Project data $\boldsymbol{x}$ to high dimension using function $\varphi(\boldsymbol{x})$
  2. Find a linear discriminant function for transformed data $\varphi(\boldsymbol{x})$
  3. Final nonlinear discriminant function is $g(\boldsymbol{x}) = \boldsymbol{w}^t \varphi(\boldsymbol{x}) + w_0$

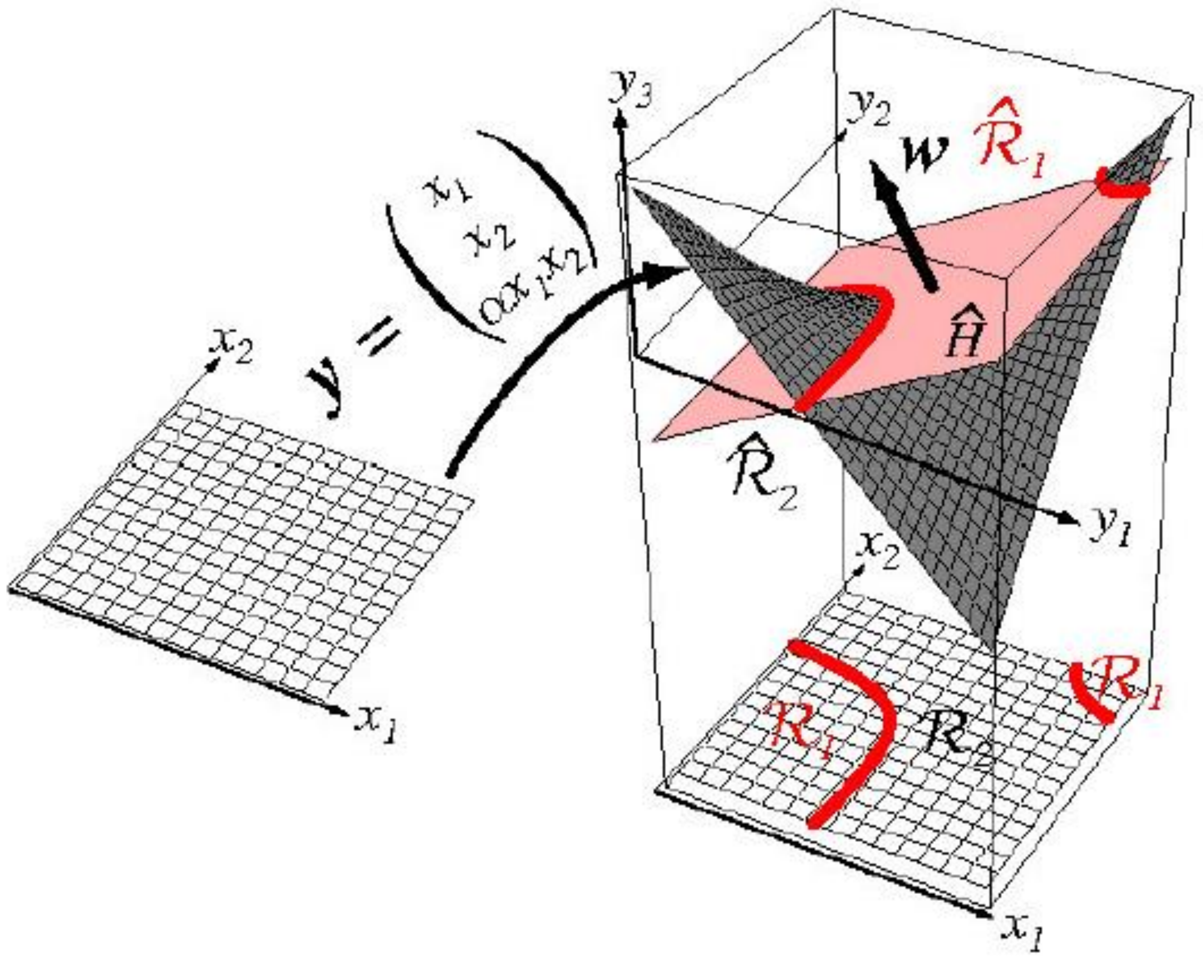


- In 2D, discriminant function is linear

$$g\left(\begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix}\right) = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix} + w_0$$

- In 1D, discriminant function is not linear $g(x) = w_1 x + w_2 x^2 + w_0$

Non Linear Mapping: Another Example



Non Linear SVM

- Can use any linear classifier after lifting data into a higher dimensional space. However we will have to deal with the "curse of dimensionality"
  1. poor generalization to test data
  2. computationally expensive

- SVM avoids the "curse of dimensionality" problems by
  1. enforcing largest margin permits good generalization
     - It can be shown that generalization in SVM is a function of the margin, independent of the dimensionality
  2. computation in the higher dimensional case is performed only implicitly through the use of ***kernel*** functions

Non Linear SVM: Kernels

- Recall SVM optimization

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

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

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

- Idea: find ***kernel*** function $K(x_i, x_j)$ s.t.

$$K(x_i, x_j) = \varphi(x_i)^t \varphi(x_j)$$

Non Linear SVM: Kernels

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

- Then we only need to compute $\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ instead of $\varphi(\boldsymbol{x}_i)^t \varphi(\boldsymbol{x}_j)$
    - "kernel trick": do not need to perform operations in high dimensional space explicitly

Non Linear SVM: Kernels

- Suppose we have 2 features and $\boldsymbol{K}(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}^t\boldsymbol{y})^2$

- Which mapping $\varphi(\boldsymbol{x})$ does it correspond to?

$$K(x,y) = \left(x^t y\right)^2 = \left(\begin{bmatrix} x^{(1)} & x^{(2)} \end{bmatrix} \begin{bmatrix} y^{(1)} \\ y^{(2)} \end{bmatrix}\right)^2 = \left(x^{(1)}y^{(1)} + x^{(2)}y^{(2)}\right)^2$$

$$= \left(x^{(1)}y^{(1)}\right)^2 + 2\left(x^{(1)}y^{(1)}\right)\left(x^{(2)}y^{(2)}\right) + \left(x^{(2)}y^{(2)}\right)^2$$

$$= \left[\left(x^{(1)}\right)^2 \quad \sqrt{2}\,x^{(1)}x^{(2)} \quad \left(x^{(2)}\right)^2\right]\left[\left(y^{(1)}\right)^2 \quad \sqrt{2}\,y^{(1)}y^{(2)} \quad \left(y^{(2)}\right)^2\right]^t$$

- Thus $\varphi(x) = \left[\left(x^{(1)}\right)^2 \quad \sqrt{2}\,x^{(1)}x^{(2)} \quad \left(x^{(2)}\right)^2\right]$

Non Linear SVM: Kernels

- How to choose kernel function $\boldsymbol{K}(\boldsymbol{x}_i,\boldsymbol{x}_j)$?
  - $\boldsymbol{K}(\boldsymbol{x}_i,\boldsymbol{x}_j)$ should correspond to product $\varphi(\boldsymbol{x}_i)^t\varphi(\boldsymbol{x}_j)$ in a higher dimensional space
  - Mercer's condition tells us which kernel function can be expressed as dot product of two vectors
- Some common choices:
  - Polynomial kernel

$$K(x_i, x_j) = (x_i^t x_j + 1)^p$$

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

$$K(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2}\|x_i - x_j\|^2\right)$$

Non Linear SVM

- search for separating hyperplane in high dimension

$$w\varphi(x) + w_0 = 0$$

- Choose $\varphi(x)$ so that the first ("0"th) dimension is the augmented dimension with feature value fixed to 1

$$\varphi(x) = \begin{bmatrix} 1 & x^{(1)} & x^{(2)} & x^{(1)}x^{(2)} \end{bmatrix}^t$$

- Threshold parameter w_0 gets folded into the weight vector w

$$\begin{bmatrix} w_0 & w \end{bmatrix} \underbrace{\begin{bmatrix} 1 \\ * \end{bmatrix}}_{\varphi(x)} = 0$$

Non Linear SVM

- Will not use notation $\boldsymbol{a} = [\boldsymbol{w_0} \;\; \boldsymbol{w}]$, we'll use old notation $\boldsymbol{w}$ and seek hyperplane through the origin

$$\boldsymbol{w}\varphi(\boldsymbol{x}) = \boldsymbol{0}$$

- If the first component of $\varphi(\boldsymbol{x})$ is not $\boldsymbol{1}$, the above is equivalent to saying that the hyperplane has to go through the origin in high dimension
  - removes only one degree of freedom
  - But we have introduced many new degrees when we lifted the data in high dimension

Non Linear SVM Recepie

- Start with data $\boldsymbol{x_1},\ldots,\boldsymbol{x_n}$ which lives in feature space of dimension $\boldsymbol{d}$
- Choose kernel $\boldsymbol{K}(\boldsymbol{x_i},\boldsymbol{x_j})$ or function $\varphi(\boldsymbol{x_i})$ which takes sample $\boldsymbol{x_i}$ to a higher dimensional space
- Find the largest margin linear discriminant function in the higher dimensional space by using quadratic programming package to solve:

$$\boxed{\begin{array}{ll} \text{maximize} & L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j K(x_i, x_j) \\ \text{constrained to} & 0 \le \alpha_i \le \beta \quad \forall i \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i z_i = 0 \end{array}}$$

Non Linear SVM Recipe

- Weight vector $\boldsymbol{w}$ in the high dimensional space:

$$\boldsymbol{w} = \sum_{x_i \in S} \alpha_i z_i \varphi(x_i)$$

  - where $\boldsymbol{S}$ is the set of support vectors $S = \{x_i \mid \alpha_i \neq 0\}$

- Linear discriminant function of largest margin in the high dimensional space:

$$g(\varphi(x)) = w^t \varphi(x) = \left(\sum_{x_i \in S} \alpha_i z_i \varphi(x_i) \right)^t \varphi(x)$$

- Non linear discriminant function in the original space:

$$g(x) = \left(\sum_{x_i \in S} \alpha_i z_i \varphi(x_i) \right)^t \varphi(x) = \sum_{x_i \in S} \alpha_i z_i \varphi^t(x_i) \varphi(x) = \sum_{x_i \in S} \alpha_i z_i K(x_i, x)$$

- decide class 1 if $\boldsymbol{g}(\boldsymbol{x}) > 0$, otherwise decide class 2

Non Linear SVM

- Nonlinear discriminant function

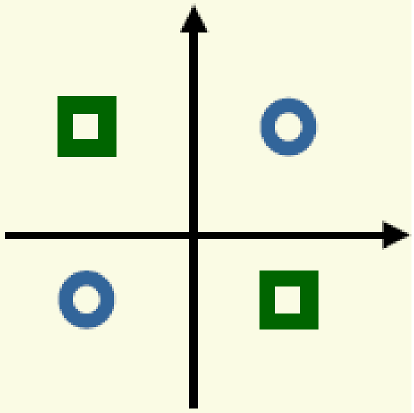
$$g(x) = \sum_{x_i \in S} \boxed{\alpha_i} \boxed{z_i} \boxed{K(x_i, x)}$$

$$g(x) = \sum \boxed{\text{weight of support vector } x_i} \boxed{\mp 1} \boxed{\text{"inverse distance" from } x \text{ to support vector } x_i}$$

most important training samples, i.e. support vectors

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

SVM Example: XOR Problem



- Class 1: $\mathbf{x_1} = [1,-1]$, $\mathbf{x_2} = [-1,1]$
- Class 2: $\mathbf{x_3} = [1,1]$, $\mathbf{x_4} = [-1,-1]$
- Use polynomial kernel of degree 2:
  - $K(x_i,x_j) = (x_i{}^t x_j + 1)^2$
  - This kernel corresponds to mapping

$$\varphi(x) = \begin{bmatrix} 1 & \sqrt{2}x^{(1)} & \sqrt{2}x^{(2)} & \sqrt{2}x^{(1)}x^{(2)} & \left(x^{(1)}\right)^2 & \left(x^{(2)}\right)^2 \end{bmatrix}^t$$

- Need to maximize

$$L_D(\alpha) = \sum_{i=1}^{4} \alpha_i - \frac{1}{2}\sum_{i=1}^{4}\sum_{j=1}^{4} \alpha_i \alpha_j z_i z_j \left(x_i^t x_j + 1\right)^2$$

constrained to $0 \le \alpha_i \quad \forall i \quad \text{and} \quad \alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 = 0$

SVM Example: XOR Problem

- Can rewrite $L_D(\alpha) = \sum_{i=1}^{4} \alpha_i - \frac{1}{2}\alpha^t H \alpha$
  - where $\alpha = [\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \alpha_4]^t$ and $H = \begin{bmatrix} 9 & 1 & -1 & -1 \\ 1 & 9 & -1 & -1 \\ -1 & -1 & 9 & 1 \\ -1 & -1 & 1 & 9 \end{bmatrix}$

- Take derivative with respect to α and set it to ***0***

$$\frac{d}{da} L_D(\alpha) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 9 & 1 & -1 & -1 \\ 1 & 9 & -1 & -1 \\ -1 & -1 & 9 & 1 \\ -1 & -1 & 1 & 9 \end{bmatrix} \alpha = 0$$

- Solution to the above is $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.25$
  - satisfies the constraints $\forall i,\ 0 \le \alpha_i$ *and* $\alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 = 0$
  - all samples are support vectors

SVM Example: XOR Problem

$$\varphi(x) = \left[1 \quad \sqrt{2}x^{(1)} \quad \sqrt{2}x^{(2)} \quad \sqrt{2}x^{(1)}x^{(2)} \quad \left(x^{(1)}\right)^2 \quad \left(x^{(2)}\right)^2\right]^t$$

- Weight vector $\boldsymbol{w}$ is:

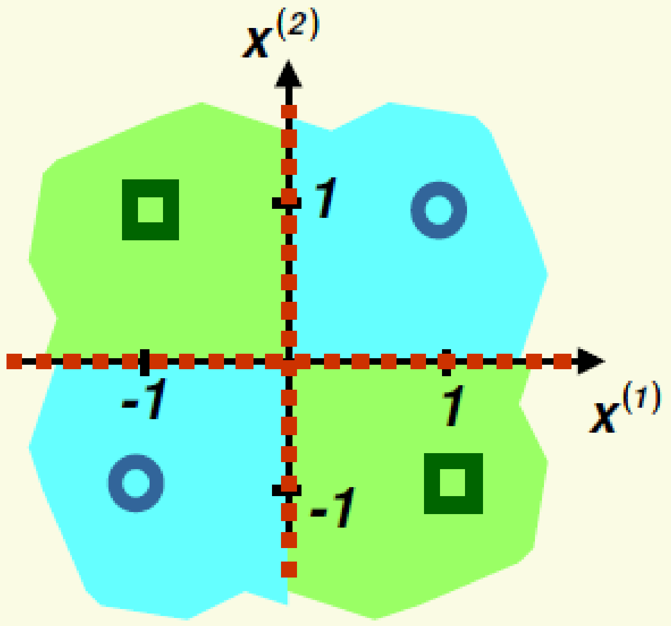
$$w = \sum_{i=1}^{4} \alpha_i z_i \varphi(x_i) = 0.25(\varphi(x_1) + \varphi(x_2) - \varphi(x_3) - \varphi(x_4))$$
$$= \left[0 \quad 0 \quad 0 \quad -\sqrt{2} \quad 0 \quad 0\right]$$

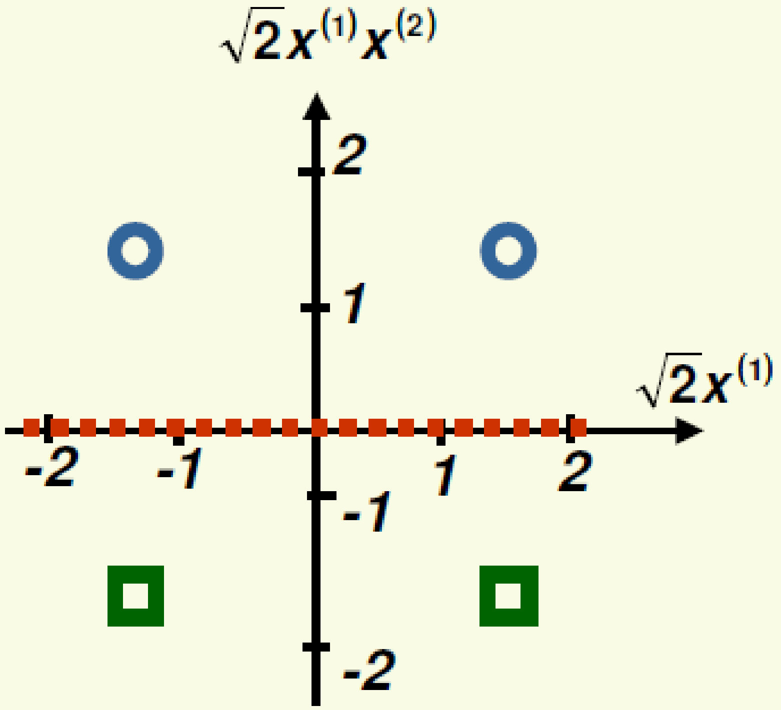
- Thus the nonlinear discriminant function is:

$$g(x) = w\varphi(x) = \sum_{i=1}^{6} w_i \varphi_i(x) = -\sqrt{2}\left(\sqrt{2}x^{(1)}x^{(2)}\right) = -2x^{(1)}x^{(2)}$$

SVM Example: XOR Problem

$$g(x) = -2x^{(1)}x^{(2)}$$





decision boundaries nonlinear

decision boundary is linear

SVM Summary

- Advantages:
  - Based on nice theory
  - excellent generalization properties
  - objective function has no local minima
  - can be used to find non linear discriminant functions
  - Complexity of the classifier is characterized by the number of support vectors rather than the dimensionality of the transformed space
- Disadvantages:
  - tends to be slower than other methods
  - quadratic programming is computationally expensive

Thank you