CS840a
Learning and Computer Vision
Prof. Olga Veksler

Lecture 3

Today

- Discuss Paper (plus video!)
  - *"Recognizing Action at a Distance" by A. Efros, A.Berg, G. Mori, Jitendra Malik*
- Support Vector Machines
- Mutual Information
- Preparation for the next time:
  - Read paper: "Object Recognition with Informative Features and Linear Classification" by M. Naquet and S. Ullman
    - Ignore section of tree-augmented network

SVM

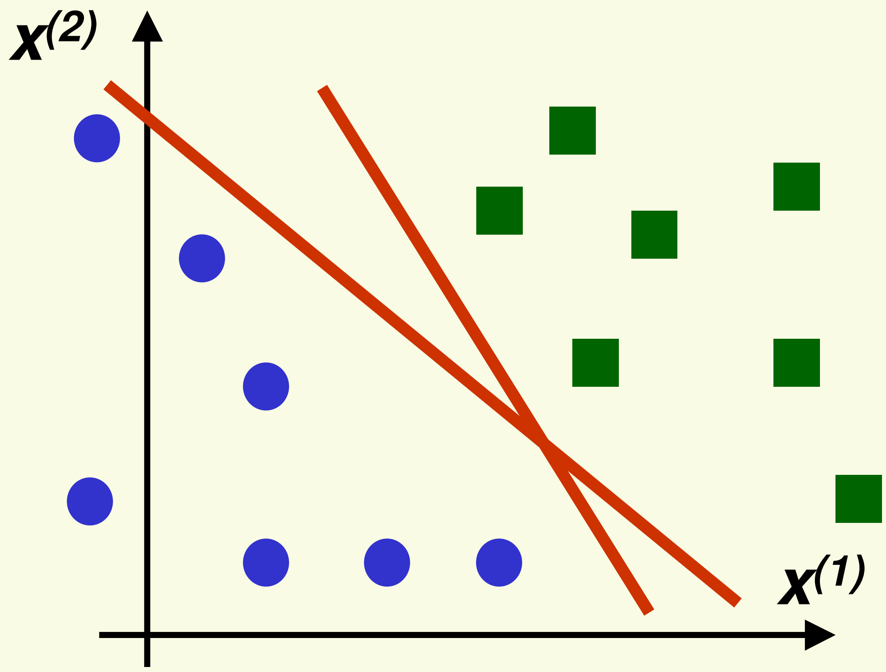
- Said to start in 1979 with Vladimir Vapnik’s paper
- Major developments throughout 1990’s
- Elegant theory
  - Has good generalization properties
- Have been applied to diverse problems very successfully in the last 10-15 years
- One of the most important developments in pattern recognition in the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

Linear Discriminant Functions

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

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0$$

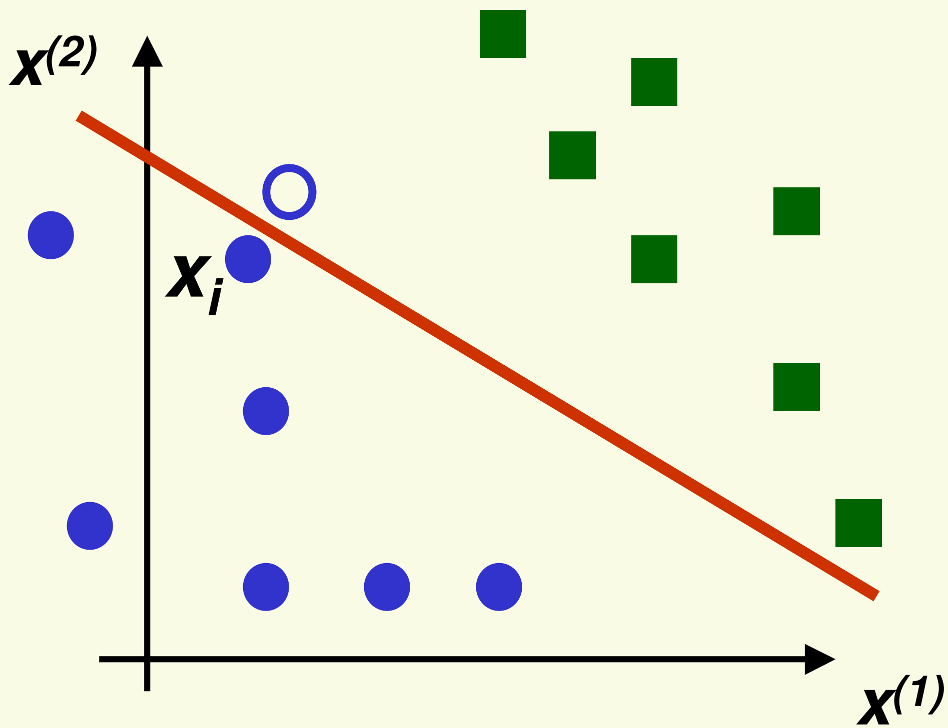
$$\boxed{\begin{array}{ll} g(\mathbf{x}) > 0 & \Rightarrow \mathbf{x} \in \textit{class } 1 \\ g(\mathbf{x}) < 0 & \Rightarrow \mathbf{x} \in \textit{class } 2 \end{array}}$$



- which separating hyperplane should we choose?

Linear Discriminant Functions

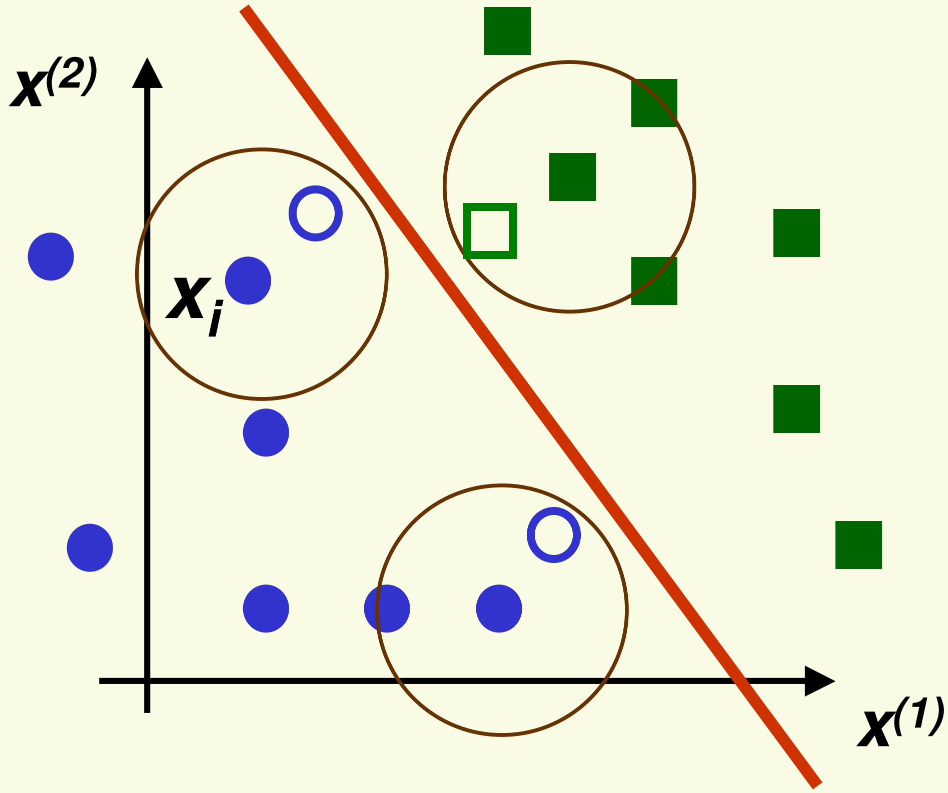
- Training data is just a subset of of all possible data
- Suppose hyperplane is close to sample $\boldsymbol{x_i}$
- If we see new sample close to sample ***i***, it is likely to be on the wrong side of the hyperplane



- Poor generalization (performance on unseen data)

Linear Discriminant Functions

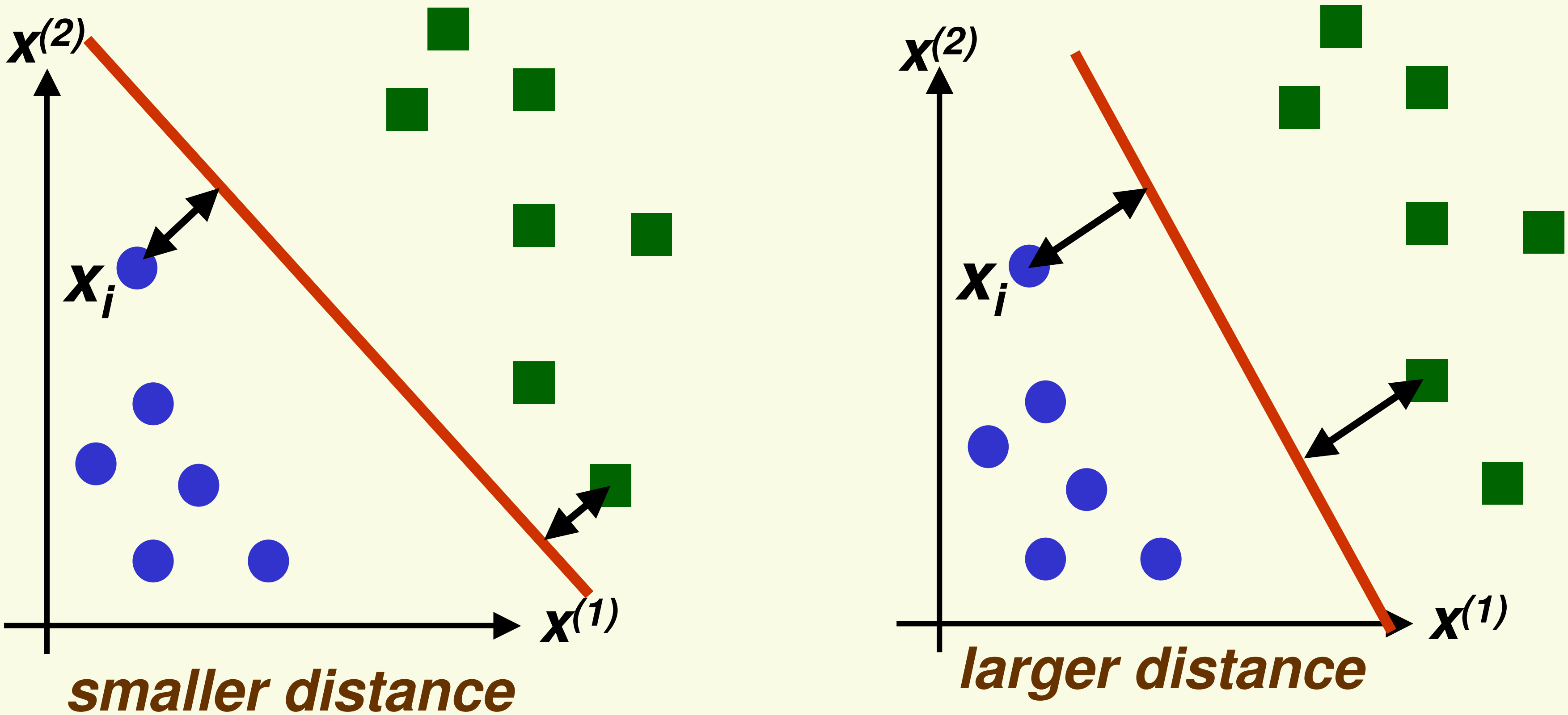
- Hyperplane as far as possible from any sample



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

SVM

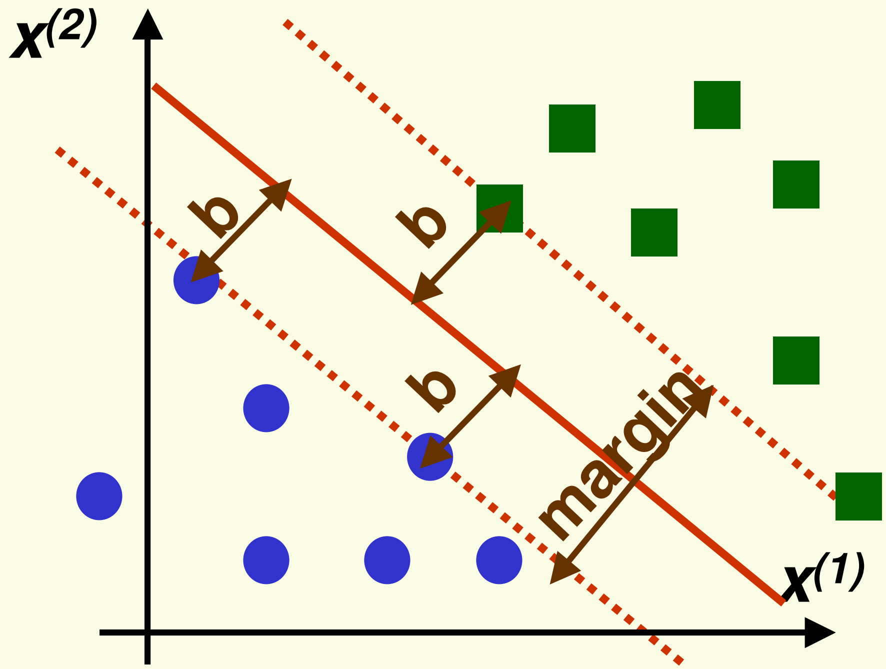
- Idea: maximize distance to the closest example



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

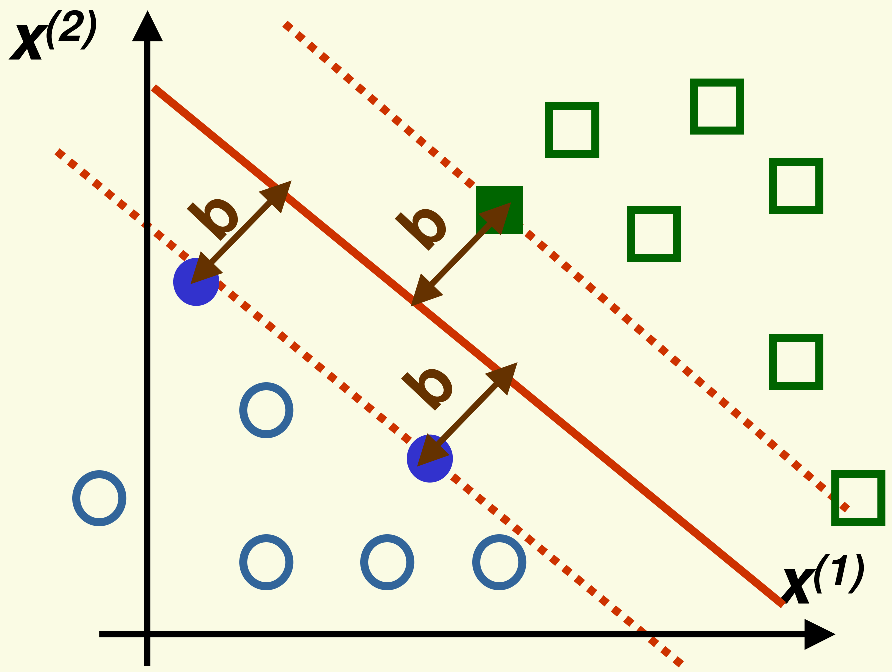
SVM: Linearly Separable Case

- 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

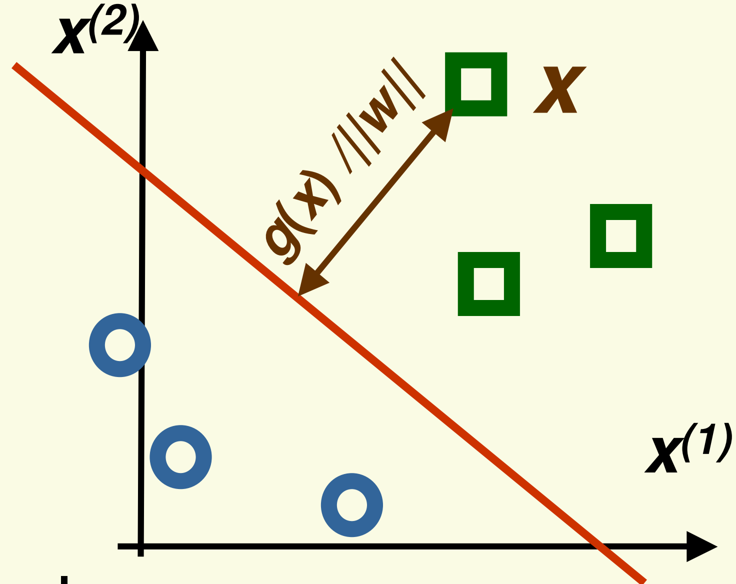


- ***Support vectors*** are the samples closest to the separating hyperplane
  - they are the most difficalt patterns to classify
  - Optimal hyperplane is completely defined by support vectors
    - of course, we do not know which samples are support vectors without finding the optimal hyperplane

SVM: Formula for the Margin

- $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{w}^t\boldsymbol{x} + \boldsymbol{w}_0$
- absolute distance between $\boldsymbol{x}$ and the boundary $\boldsymbol{g}(\boldsymbol{x}) = 0$

$$\frac{\left|\boldsymbol{w}^t\boldsymbol{x} + \boldsymbol{w}_0\right|}{\|\boldsymbol{w}\|}$$



- distance is unchanged for hyperplane $\boldsymbol{g}_1(\boldsymbol{x}) = \alpha\boldsymbol{g}(\boldsymbol{x})$

$$\frac{\left|\alpha\boldsymbol{w}^t\boldsymbol{x} + \alpha\boldsymbol{w}_0\right|}{\|\alpha\boldsymbol{w}\|} = \frac{\left|\boldsymbol{w}^t\boldsymbol{x} + \boldsymbol{w}_0\right|}{\|\boldsymbol{w}\|}$$

- Let $\boldsymbol{x}_i$ be an example closest to the boundary. Set

$$\left|\boldsymbol{w}^t\boldsymbol{x}_i + \boldsymbol{w}_0\right| = 1$$

- Now the largest margin hyperplane is unique

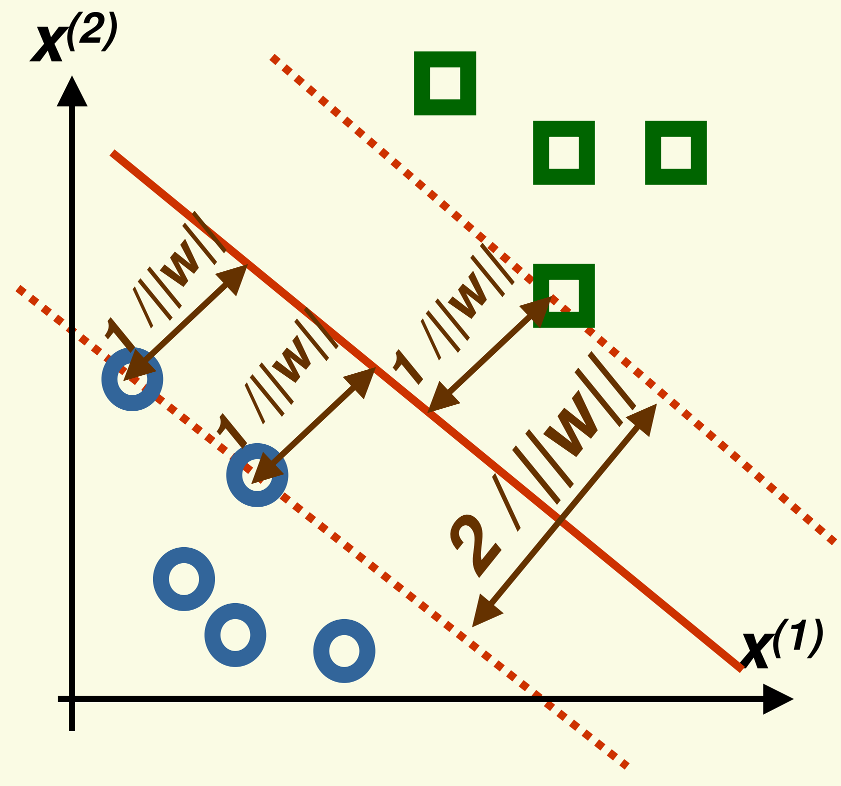
SVM: Formula for the Margin

- 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 $\quad \boldsymbol{m} = \dfrac{2}{\|\boldsymbol{w}\|}$
  - subject to constraints

$$\begin{cases} \boldsymbol{w}^t \boldsymbol{x}_i + \boldsymbol{w}_0 \geq 1 & \text{if } \boldsymbol{x}_i \text{ is positive example} \\ \boldsymbol{w}^t \boldsymbol{x}_i + \boldsymbol{w}_0 \leq -1 & \text{if } \boldsymbol{x}_i \text{ is negative example} \end{cases}$$

- Let $\begin{cases} \boldsymbol{z}_i = 1 & \text{if } \boldsymbol{x}_i \text{ is positive example} \\ \boldsymbol{z}_i = -1 & \text{if } \boldsymbol{x}_i \text{ is negative example} \end{cases}$

- Can convert our problem to

$$\boxed{\begin{array}{l} \text{minimize } \ \boldsymbol{J}(\boldsymbol{w}) = \dfrac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{constrained to } \ \boldsymbol{z}_i\left(\boldsymbol{w}^t \boldsymbol{x}_i + \boldsymbol{w}_0\right) \geq 1 \quad \forall \boldsymbol{i} \end{array}}$$

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

SVM: Optimal Hyperplane

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

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

- $\alpha = \{\alpha_1, \ldots, \alpha_n\}$ 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

$$H_{ij} = z_i z_j x_i^t x_j$$

SVM: Optimal Hyperplane

- Use Kuhn-Tucker theorem to convert our problem 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 $$\alpha_i \geq 0 \quad \forall i \quad \textit{and} \quad \sum_{i=1}^{n} \alpha_i z_i = 0$$

- $\alpha = \{\alpha_1, \ldots, \alpha_n\}$ 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 $\alpha_i\left[z_i\left(w^t x_i + w_0\right) - 1\right] = 0$

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

- 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

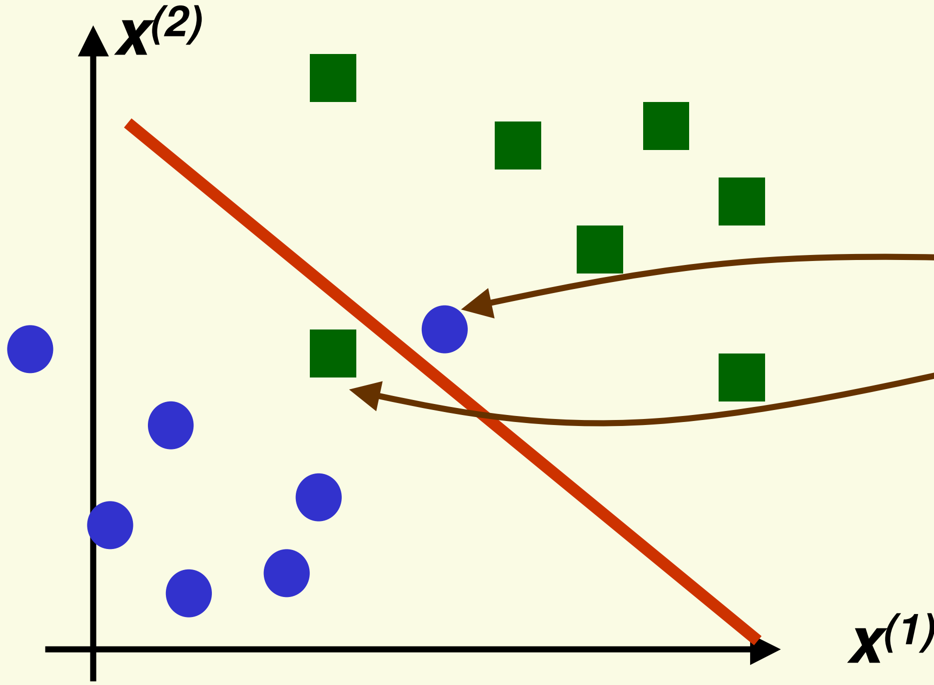
maximize $\quad L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j x_i^t x_j$

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 $x_i^t x_j$
- This will become important when looking for a ***nonlinear*** discriminant function, as we will see soon
- Code available on the web to optimize
  - I'll put a link on the web page in case you want to play with SVM for you final project

SVM: Non Separable Case

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



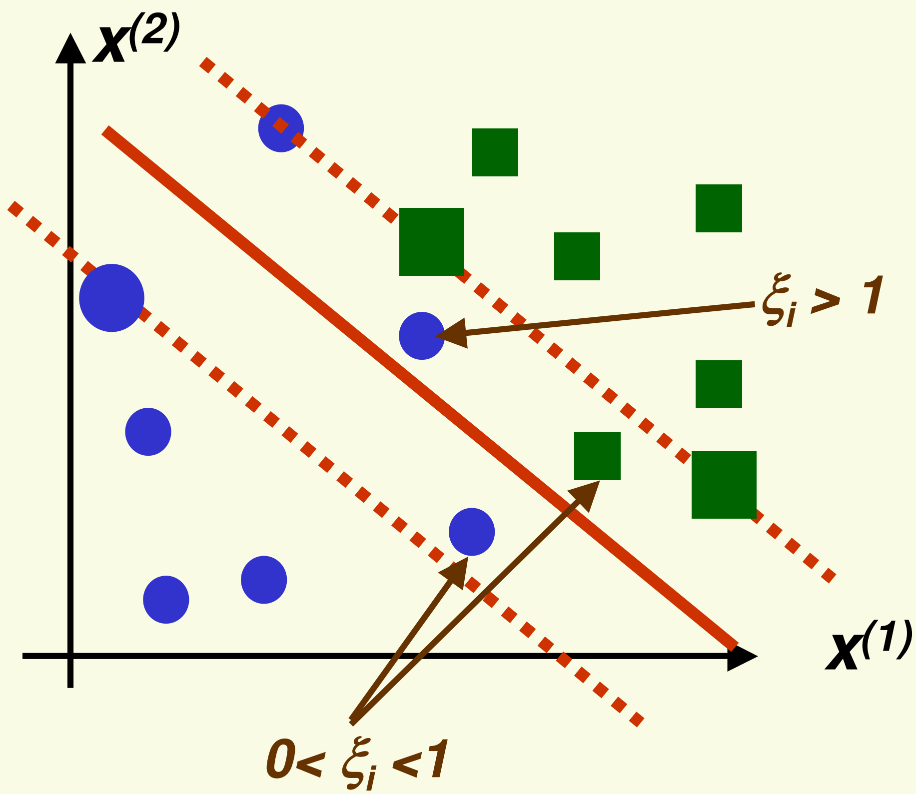
- Can apply SVM in non linearly separable case
  - data should be "almost" linearly separable for good performance

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

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

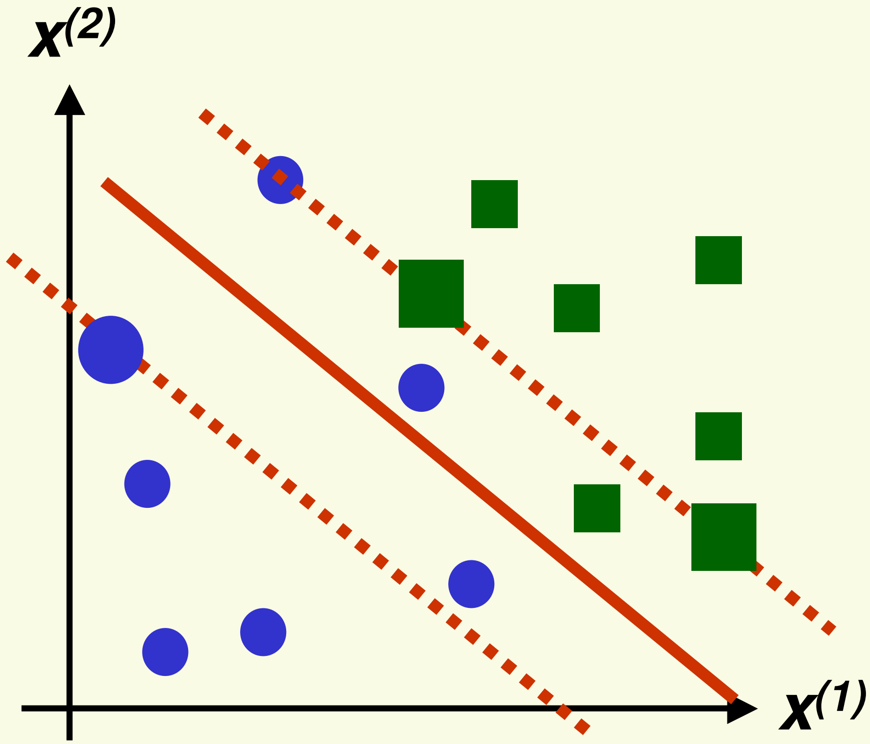
# of samples not in ideal location

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

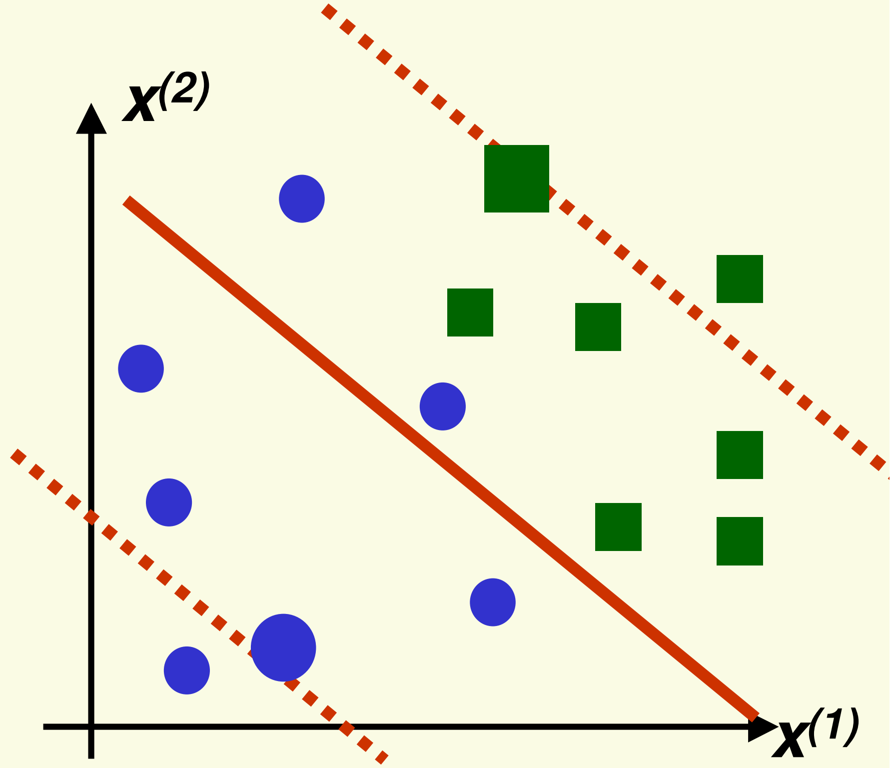
SVM: Non Separable Case

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

\# 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 $z_i(w^t x_i + w_0) \ge 1 - \xi_i$ and $\xi_i \ge 0 \quad \forall i$

SVM: Non Separable Case

- Instead we minimize

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

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

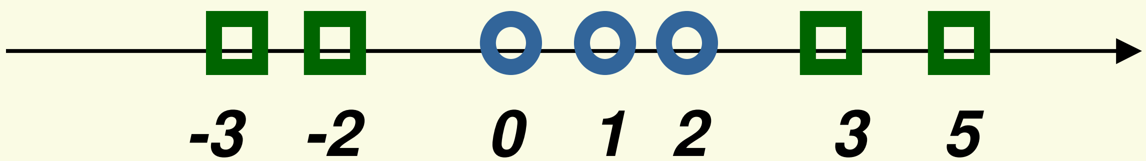
constrained to $0 \leq \alpha_i \leq \beta \quad \forall i$ ***and*** $\sum_{i=1}^{n} \alpha_i z_i = 0$

  - 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 $\alpha_i \left[z_i (w^t x_i + w_0) - 1 \right] = 0$

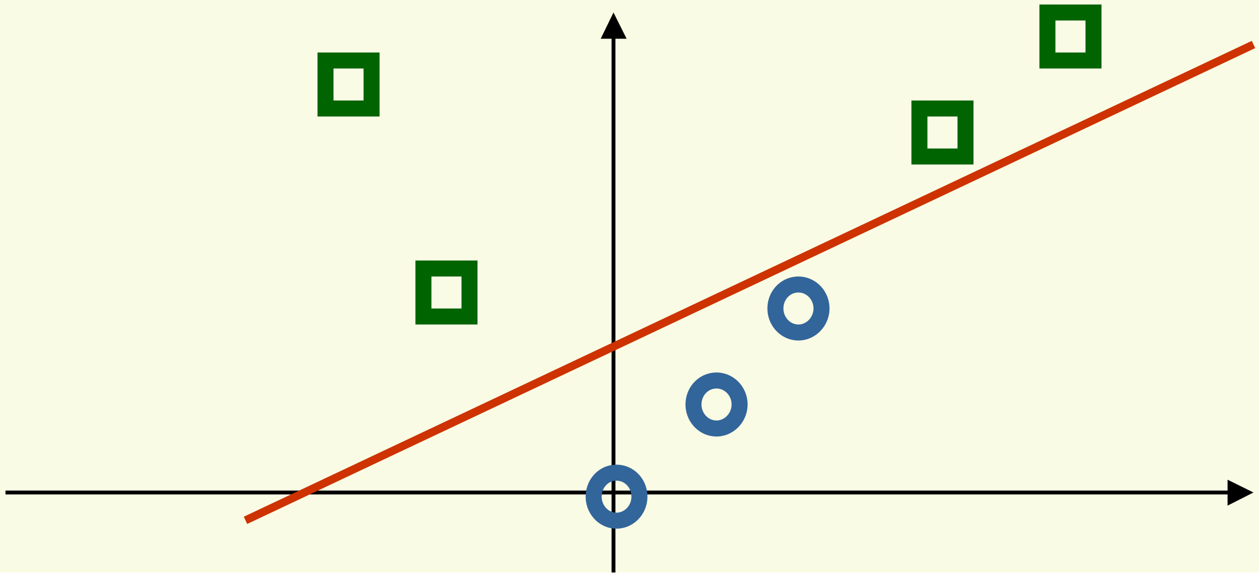
Non Linear Mapping

- 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

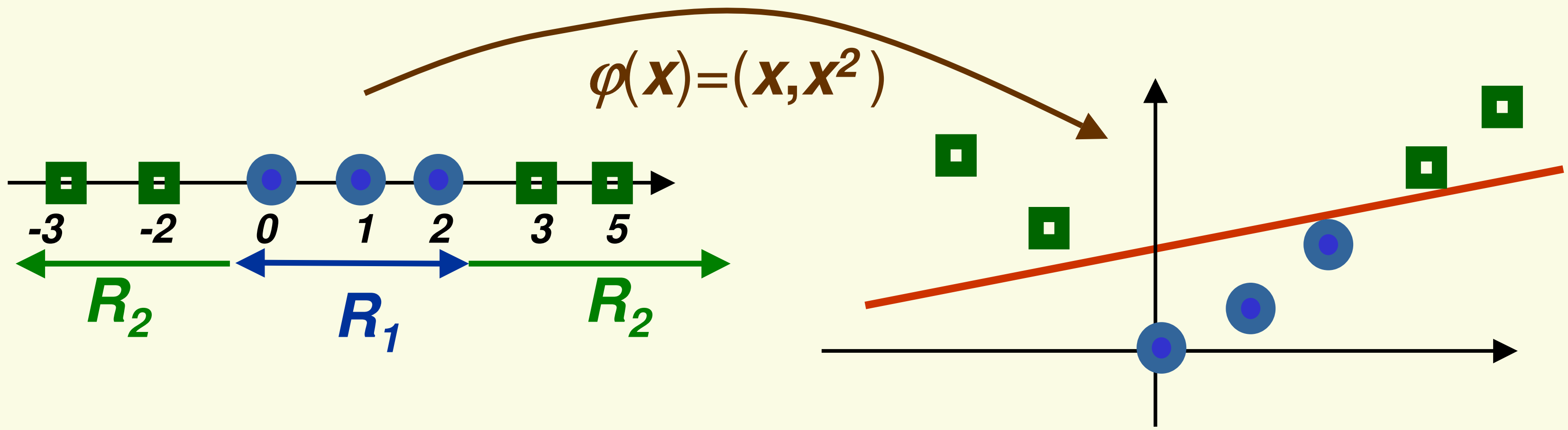


- Lift to two dimensional space with $\varphi(\boldsymbol{x})=(\boldsymbol{x},\boldsymbol{x}^2)$



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 $g(\boldsymbol{x}) = \boldsymbol{w}^t \varphi(\boldsymbol{x}) + w_0$

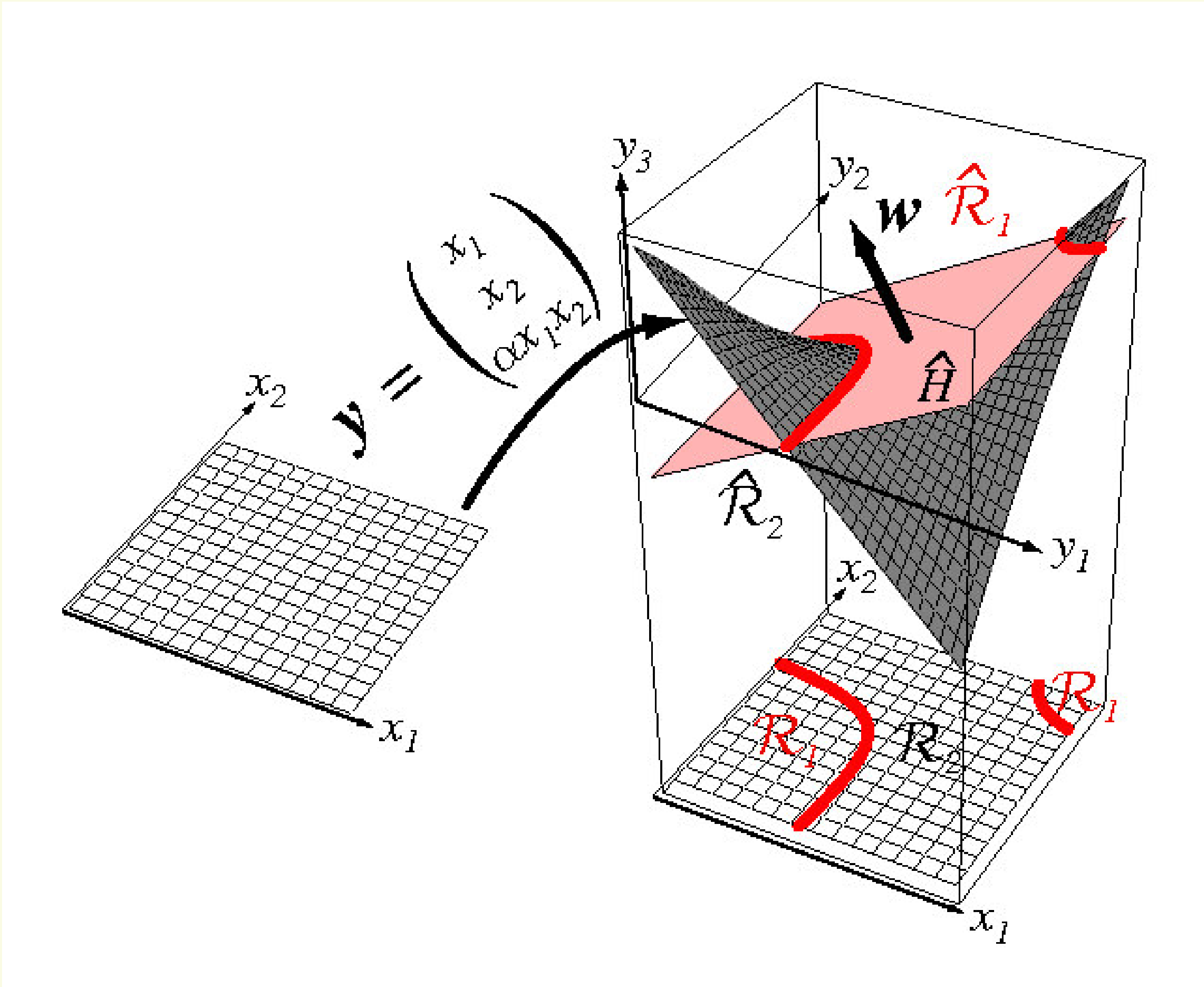


- In 2D, discriminant function is linear

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

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

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

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

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

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

  $K(\mathbf{x}_i, \mathbf{x}_j)$

- Idea: find ***kernel*** function $K(\mathbf{x}_i, \mathbf{x}_j)$ s.t.

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

Non Linear SVM: Kernels

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

- Then we only need to compute $K(x_i, x_j)$ instead of $\varphi(x_i)^t \varphi(x_j)$
  - "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?

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

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

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

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

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 $\varphi(\boldsymbol{x_i})^{\boldsymbol{t}}\varphi(\boldsymbol{x_j})$ 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

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

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

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

Non Linear SVM

- search for separating hyperplane in high dimension

$$\boldsymbol{w}\varphi(\boldsymbol{x}) + \boldsymbol{w_0} = \boldsymbol{0}$$

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

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

- Threshold parameter $\boldsymbol{w_0}$ gets folded into the weight vector $\boldsymbol{w}$

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

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:

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

Non Linear SVM Recipe

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

$$\boldsymbol{w} = \sum_{\boldsymbol{x}_i \in \boldsymbol{S}} \alpha_i \boldsymbol{z}_i \varphi(\boldsymbol{x}_i)$$

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

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

$$\boldsymbol{g}(\varphi(\boldsymbol{x})) = \boldsymbol{w}^t \varphi(\boldsymbol{x}) = \left(\sum_{\boldsymbol{x}_i \in \boldsymbol{S}} \alpha_i \boldsymbol{z}_i \varphi(\boldsymbol{x}_i) \right)^t \varphi(\boldsymbol{x})$$

- Non linear discriminant function in the original space:

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

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

Non Linear SVM

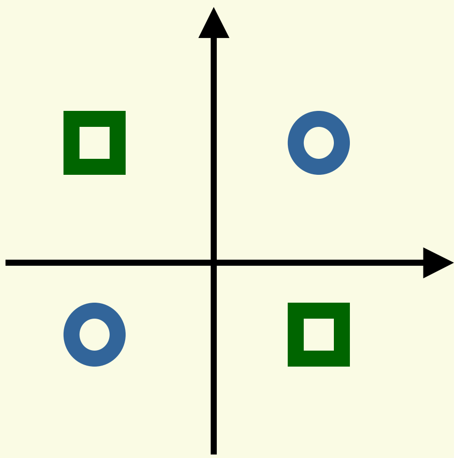
- Nonlinear discriminant function

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

$$g(x) = \sum_{\substack{\text{most important} \\ \text{training samples,} \\ \text{i.e. support vectors}}} \boxed{\text{weight of support vector } x_i}\ \boxed{\mp 1}\ \boxed{\begin{gathered}\text{"inverse distance"} \\ \text{from } x \text{ to} \\ \text{support vector } x_i\end{gathered}}$$

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

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

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

- Need to maximize

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

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

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 $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \mathbf{0.25}$
  - 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

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

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

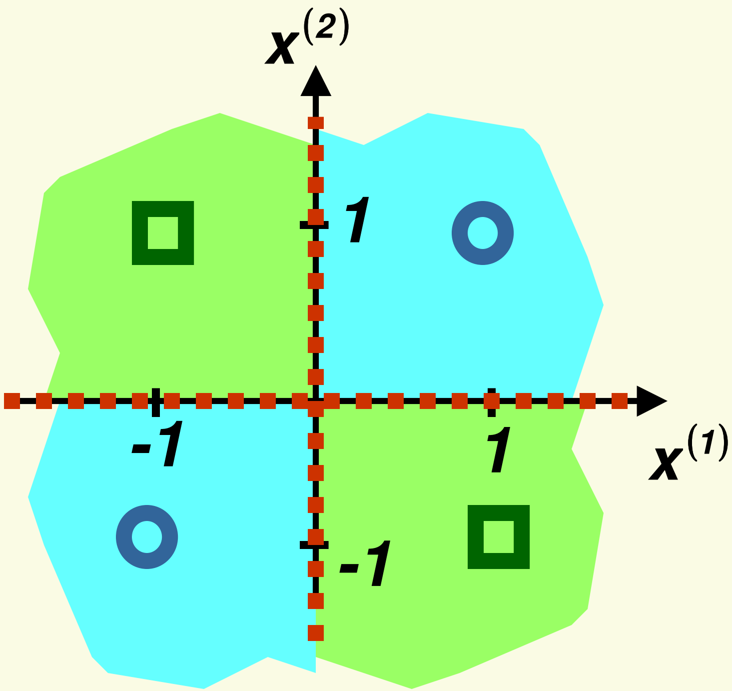
$$\boldsymbol{w} = \sum_{i=1}^{4} \alpha_i z_i \varphi(\boldsymbol{x}_i) = 0.25\left(\varphi(\boldsymbol{x}_1) + \varphi(\boldsymbol{x}_2) - \varphi(\boldsymbol{x}_3) - \varphi(\boldsymbol{x}_4)\right)$$
$$= \left[0 \quad 0 \quad 0 \quad -\sqrt{2} \quad 0 \quad 0\right]$$

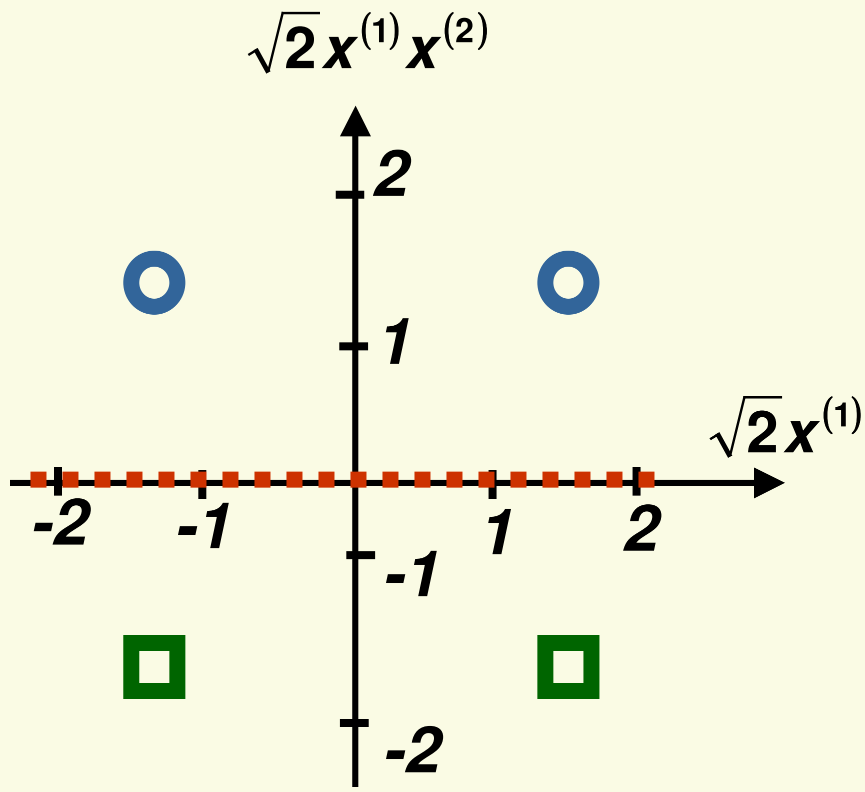
- Thus the nonlinear discriminant function is:

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

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

Information theory

- Let's think written numbers:
  - k digits → 10^k possible messages
- How about written English?
  - k letters → 26^k possible messages
  - k words → D^k possible messages, where D is English dictionary size

∴ Length ~ log(complexity)

Shannon's Entropy

$$H[p(x)] = \sum_x p(x) \frac{1}{\log p(x)} = E_x\left[\log_2 \frac{1}{p(x)}\right]$$

- weighs the information based on the probability that an outcome will occur
- second term shows the amount of information an event provides is inversely proportional to its probability of occurring

Interpretations of Entropy

$$H[x] = \sum_x p(x) \frac{1}{\log p(x)} = E_x\left[\log_2 \frac{1}{p(x)}\right]$$

- How much randomness (or uncertainty) is there in the signal with distribution p(x)
  - For uniform distribution (every event is equally likely), H[x] is high
  - If p(x) = 1 for some event x, then H[x] = 0
  - Systems with one very common event have less entropy than systems with many equally probable events
- The expected length (encoded in binary bits) of a message following distribution p(x)

Conditional Entropy of X given Y

$$H[x \mid y] = \sum_{x,y} p(x,y) \frac{1}{\log p(x \mid y)}$$

- Measures average uncertainty about x when y is known
- Property:
  - H[x] ≥ H[x|y], which means after seeing new data (y), the uncertainty about x is not increased, on average

Mutual Information of X and Y

$$\boldsymbol{I}\left[\boldsymbol{x},\boldsymbol{y}\right]=\boldsymbol{H}(\boldsymbol{x})-\boldsymbol{H}(\boldsymbol{x}\mid\boldsymbol{y})$$

- Measures the average reduction in uncertainty about x after y is known
- or, equivalently, it **measures the amount of information that x conveys about y**
- Properties
  - $I(x,y) = I(y,x)$
  - $I(x,y) \geq 0$
  - If x and y are independent, then $I(x,y) = 0$
  - $I(x,x) = H(x)$

MI for Feature Selecton

$$I[x,y] = H(x) - H(x \mid y)$$

- Let x be a proposed feature and y be the class
- If I[x,y] is high, we can expect feature x be good at predicting class y