CS840a
Learning and Computer Vision
Prof. Olga Veksler

Lecture 3

Information Theory (a little BIT)

SVM

Some pictures from C. Burges

Information theory

- Information Theory regards information as only those symbols that are uncertain to the receiver
  only infrmatn esentil to understnd mst b tranmitd
- Shannon made clear that uncertainty is the very commodity of communication
- The amount of information, or uncertainty, output by an information source is a measure of its entropy
- In turn, a source's entropy determines the amount of bits per symbol required to encode the source's information
- Messages are encoded with strings of 0 and 1 (bits)

Today

- Mutual Information
- Preparation for the next time:
  - paper: "Object Recognition with Informative Features and Linear Classification" by M. Naquet and S. Ullman
    - Ignore section of tree-augmented network
- Support Vector Machines

Information theory

- Suppose we toss a **fair** die with 8 sides
  - need 3 bits to transmit the results of each toss
  - 1000 throws will need 3000 bits to transmit
- Suppose the die is biased
  - side A occurs with probability 1/2, chances of throwing B are 1/4, C are 1/8, D are 1/16, E are 1/32, F 1/64, G and H are 1/128
  - Encode A= 0, B = 10, C = 110, D = 1110,…, so on until G = 1111110, H = 1111111
  - We need, on average, 1/2+2/4+3/8+4/16+5/32+6/64+7/128+7/128 = 1.984 bits to encode results of a toss
  - 1000 throws require 1984 bits to transmit
  - Less bits to send = less "information"
  - Biased die tosses contain less "information" than unbiased die tosses (know in advance biased sequence will have a lot of A's)
  - What's the number of bits in the best encoding?
- Extreme case: if a die always shows side A, a sequence of 1,000 tosses has no information, 0 bits to encode

Information theory

- if a die is fair (any side is equally likely, or uniform distribution), for any toss we need log(8) = 3 bits
- Suppose any of n events is equally likely (uniform distribution)
  - P(x) = 1/n, therefore -log P = -log(1/n) = log n
- In the "good" encoding strategy for our biased die example, every side x has -log p(x) bits in its code
- Expected number of bits is

$$-\sum_x p(x)\log p(x)$$

Shannon's Entropy

$$H[p(x)] = -\sum_x p(x)\, log\; p(x) = \sum_x p(x)\, log \frac{1}{p(x)}$$

- How much randomness (or uncertainty) is there in the value of signal x if it has distribution p(x)
  - For uniform distribution (every event is equally likely), H[x] is maximum
  - 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
- Gives the expected length of optimal encoding (in binary bits) of a message following distribution p(x)
  - doesn't actually give this optimal encoding

Conditional Entropy of X given Y

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

- 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

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

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

MI for Feature Selection

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

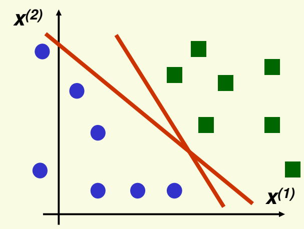
- Let x be a proposed feature and c be the class
- If I[x,c] is high, we can expect feature x be good at predicting class c

Linear Discriminant Functions

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

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

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



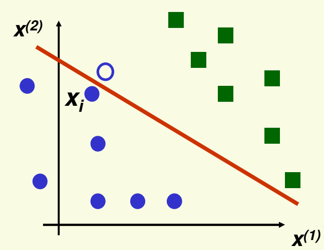
- which separating hyperplane should we choose?

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

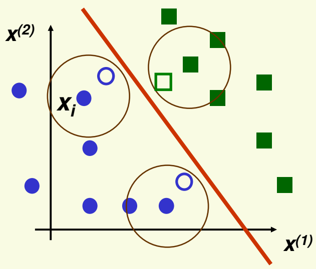
- Training data is just a subset of of all possible data
- Suppose hyperplane is close to sample x_i
- If we see new sample close to sample i, it is likely to be on the wrong side of the hyperplane



- Poor generalization (performance on unseen data)

Linear Discriminant Functions

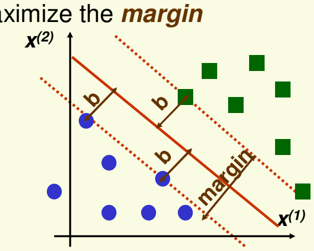
- Hyperplane as far as possible from any sample



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

SVM: Linearly Separable Case

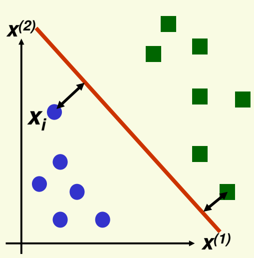
- SVM: maximize the ***margin***



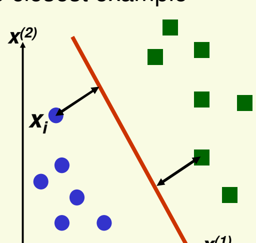
- ***margin*** is twice the absolute value of distance ***b*** of the closest example to the separating hyperplane
- Better generalization (performance on test data)
  - in practice
  - and in theory

SVM

- Idea: maximize distance to the closest example



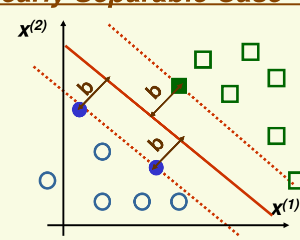
smaller distance



larger distance

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

SVM: Linearly Separable Case

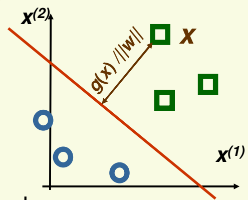


- ***Support vectors*** are the samples closest to the separating hyperplane
  - they are the most difficalt patterns to classify
  - Optimal hyperplane is completely defined by support vectors
    - of course, we do not know which samples are support vectors without finding the optimal hyperplane

SVM: Formula for the Margin

- $g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0$
- absolute distance between $\mathbf{x}$ and the boundary $g(\mathbf{x}) = 0$

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



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

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

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

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

- Now the largest margin hyperplane is unique

SVM: Optimal Hyperplane

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

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

- Let $\begin{cases} z_i = 1 & \text{if } \mathbf{x}_i \text{ is positive example} \\ z_i = -1 & \text{if } \mathbf{x}_i \text{ is negative example} \end{cases}$
- Can convert our problem to

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

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

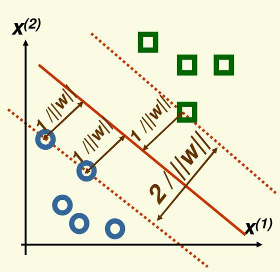
SVM: Formula for the Margin

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

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

- Thus the margin is

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



SVM: Optimal Hyperplane

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

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

- $\alpha = \{\alpha_1, \ldots, \alpha_n\}$ 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 $H_{ij} = z_i z_j \mathbf{x}_i^t\mathbf{x}_j$

SVM: Optimal Hyperplane

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

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

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

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

- $L_D(\alpha)$ 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

SVM: Optimal Hyperplane

- After finding the optimal $\alpha = \{\alpha_1, \dots, \alpha_n\}$
  - 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:

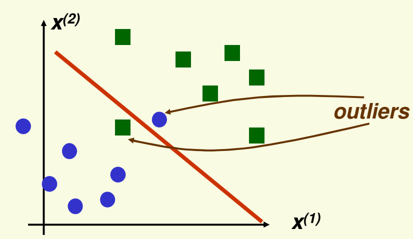
$$g(x) = \left(\sum_{x_i \in S} \alpha_i z_i x_i\right)^t x + w_0$$

  - where S is the set of support vectors

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

SVM: Non Separable Case

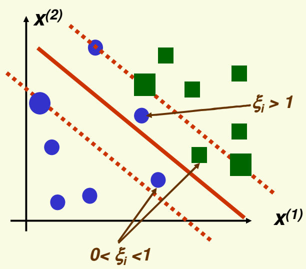
- Data is most likely to be not linearly separable, but linear classifier may still be appropriate



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

SVM: Non Separable Case

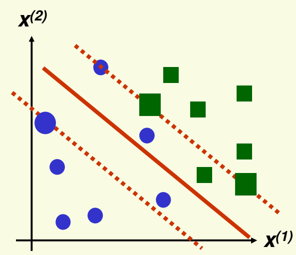
- Use non-negative slack variables $\xi_1,\dots,\xi_n$ (one for each sample)
- Change constraints from $z_i(w^t x_i + w_0) \geq 1 \quad \forall i$ to
  $z_i(w^t x_i + w_0) \geq 1 - \xi_i \quad \forall i$
- ξ_i is a measure of deviation from the ideal for sample i
  - $\xi_i > 1$ sample i is on the wrong 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



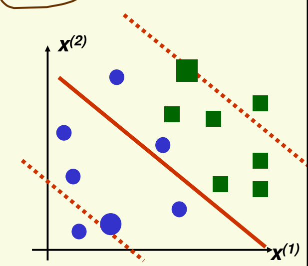
SVM: Non Separable Case

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

# 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

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

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

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

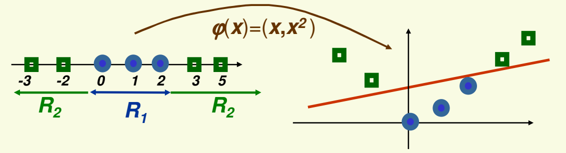
- Can use Kuhn-Tucker theorem to converted to

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

  - 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

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



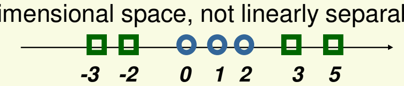
- In 2D, discriminant function is linear

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

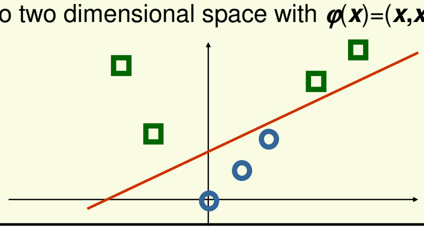
- In 1D, discriminant function is not linear $g(x)=w_1x+w_2x^2+w_0$

Non Linear Mapping

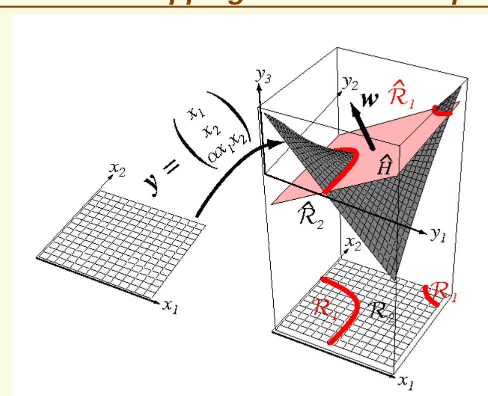
- Cover's theorem:
  - "*pattern-classification problem cast in a high dimensional space non-linearly is more likely to be linearly separable than in a low-dimensional space*"
- One dimensional space, not linearly separable



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



Non Linear Mapping: 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

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

$K(x_i, x_j)$

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

Non Linear SVM: Kernels

- Recall SVM optimization

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

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

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

  $K(x_i, x_j)$

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

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

Non Linear SVM: Kernels

- Suppose we have 2 features and $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^t \boldsymbol{y})^2$
- Which mapping $\boldsymbol{\varphi}(\boldsymbol{x})$ does it correspond to?

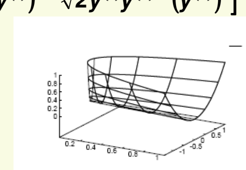
$$K(x,y) = (x^t y)^2 = \left(\begin{bmatrix} x^{(1)} & x^{(2)} \end{bmatrix} \begin{bmatrix} y^{(1)} \\ y^{(2)} \end{bmatrix} \right)^2 = \left(x^{(1)} y^{(1)} + x^{(2)} y^{(2)} \right)^2$$

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

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

- 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 $K(x_i,x_j)$?
  - $K(x_i,x_j)$ 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
  - Kernel's not satisfying Mercer's condition can be sometimes used, but no geometrical interpretation
- Some common choices (satisfying Mercer's condition):
  - 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

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

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

- If the first component of $\varphi(x)$ is not 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

- 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

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

Non Linear SVM Recepie

- Start with data $x_1,\dots,x_n$ which lives in feature space of dimension d
- Choose kernel $K(x_i,x_j)$ or function $\varphi(x_i)$ which takes sample 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 w in the high dimensional space:

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

  - where 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:

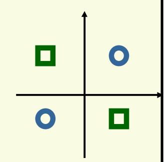
$$g(\varphi(x)) = w^t \varphi(x) = \left(\sum_{x_i \in S} \alpha_i z_i \varphi(x_i) \right)^t \varphi(x)$$

- 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 $g(x) > 0$, otherwise decide class 2

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

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

constrained to $0 \leq \alpha_i \quad \forall i \quad and \quad \alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 = 0$

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

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

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 $\forall i, \; 0 \leq \alpha_i \; and \; \alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 = 0$
  - all samples are support vectors

SVM Example: XOR Problem

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

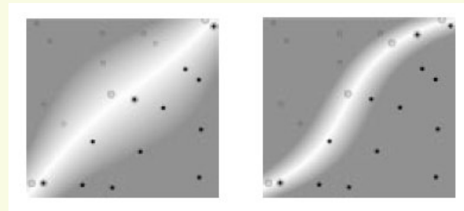
- Weight vector ***w*** is:

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

- Thus the nonlinear discriminant function is:

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

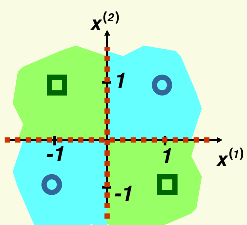
Degree 3 Polynomial Kernel

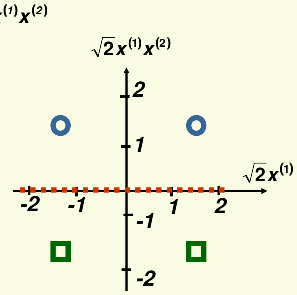


- In linearly separable case (on the left), decision boundary is roughly linear, indicating that dimensionality is controlled
- Nonseparable case (on the right) is handled by a polynomial of degree 3

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
  - Not clear how to choose the Kernel