Lecture Slides for

INTRODUCTION TO

Machine Learning

ETHEM ALPAYDIN

Edited for CS 536 Fall 2005 – Rutgers University
Ahmed Elgammal

This is a draft version. The final version will be posted later

alpaydin@boun.edu.tr
http://www.cmpe.boun.edu.tr/~ethem/i2ml

CHAPTER 10:

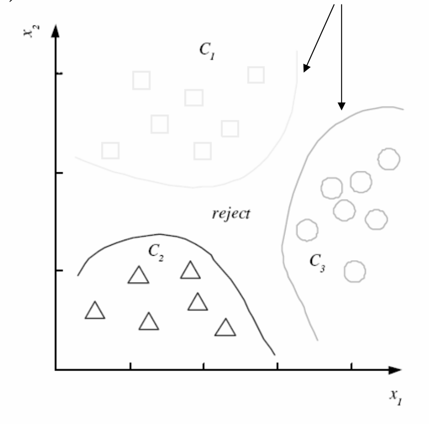
Linear Discrimination
Kernel Methods
Support Vector Machines

Recall - Discriminant Functions

choose C_i if $g_i(\boldsymbol{x}) = \max_k g_k(\boldsymbol{x})$

$g_i(\boldsymbol{x}), i = 1, \ldots, K$

$$g_i(\boldsymbol{x}) = \begin{cases} -R(\alpha_i \mid \boldsymbol{x}) \\ P(C_i \mid \boldsymbol{x}) \\ p(\boldsymbol{x} \mid C_i)P(C_i) \end{cases}$$



K decision regions $\mathcal{R}_1, \ldots, \mathcal{R}_K$

$$\mathcal{R}_i = \{\boldsymbol{x} \mid g_i(\boldsymbol{x}) = \max_k g_k(\boldsymbol{x})\}$$

Likelihood- vs. Discriminant-based Classification

- Likelihood-based: Assume a model for $p(\boldsymbol{x}|C_i)$, use Bayes' rule to calculate $P(C_i|\boldsymbol{x})$

  $g_i(\boldsymbol{x}) = \log P(C_i|\boldsymbol{x})$
- Discriminant-based: Assume a model for $g_i(\boldsymbol{x}|\Phi_i)$; no assumption about the densities; no density estimation
- Discriminant-based is non-parametric (w.r.t. the class densities)
- Estimating the boundaries is enough; no need to accurately estimate the densities inside the boundaries
- Learning: optimization of the parameters Φ_i to maximize the classification accuracy given labeled training data (or minimize error function)
- Inductive bias ?

Linear Discriminant

- Linear discriminant:

$$g_i(\boldsymbol{x} \mid \boldsymbol{w}_i, w_{i0}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0} = \sum_{j=1}^{d} w_{ij} x_j + w_{i0}$$

- Advantages:
  - Simple: O(d) space/computation
  - Knowledge extraction: Weighted sum of attributes; positive/negative weights, magnitudes (credit scoring)
  - Optimal when $p(\boldsymbol{x}|C_i)$ are Gaussian with shared cov matrix; useful when classes are (almost) linearly separable

Generalized Linear Model

- Quadratic discriminant:

$$g_i(\boldsymbol{x} \mid \mathbf{W}_i, \boldsymbol{w}_i, w_{i0}) = \boldsymbol{x}^T \mathbf{W}_i \boldsymbol{x} + \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0}$$

- Higher-order (product) terms:

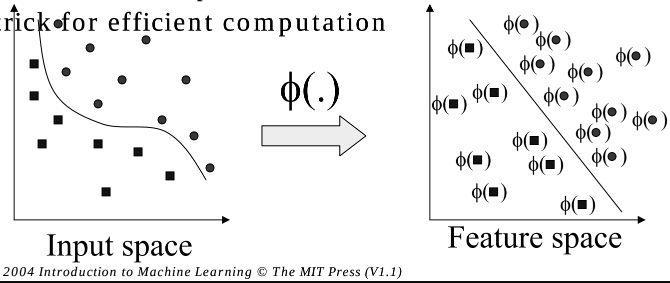
$$z_1 = x_1,\ z_2 = x_2,\ z_3 = x_1^2,\ z_4 = x_2^2,\ z_5 = x_1 x_2$$

Map from ***x*** to ***z*** using nonlinear basis functions and use a linear discriminant in ***z***-space

$$g(\mathbf{x}) = \sum_{j=1}^{k} w_j \phi_j(\mathbf{x})$$

Extension to Non-linear

- Key idea: transform $\mathbf{x}_i$ to a higher dimensional space to "make life easier"
  - Input space: the space containing $\mathbf{x}_i$
  - Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
- Why transform?
  - Linear operation in the feature space is equivalent to non-linear operation in input space
  - The classification task can be "easier" with a proper transformation. Example: XOR
- Kernel trick for efficient computation



Kernel Trick

- The relationship between the kernel function K and the mapping $\phi(.)$ i $K(\mathbf{x},\mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$
  - This is known as the *kernel trick*
- In practice, we specify K, thereby specifying $\phi(.)$ indirectly, instead of choosing $\phi(.)$
- Intuitively, $K(\mathbf{x},\mathbf{y})$ represents our desired notion of similarity between data $\mathbf{x}$ and $\mathbf{y}$ and this is from our prior knowledge
- $K(\mathbf{x},\mathbf{y})$ needs to satisfy a technical condition (Mercer condition) in order for $\phi(.)$ to exist

Kernel Trick

- Define the kernel function $K(\mathbf{x},\mathbf{y})$ as

$$K(\mathbf{x},\mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- Consider the following transformation

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

$$\langle\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}), \phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix})\rangle = (1 + x_1y_1 + x_2y_2)^2 = K(\mathbf{x},\mathbf{y})$$

- The inner product can be computed by *K* without going through the map $\phi(.)$

Example Kernel Functions

- Polynomial kernel with degree *d*

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$

- Radial basis function kernel with width σ

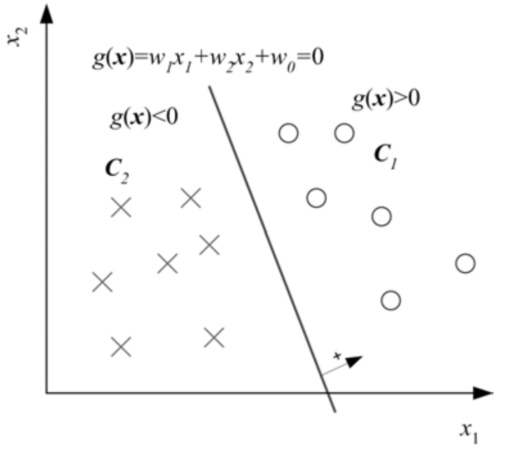
$$K(\mathbf{x},\mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2))$$

  - Closely related to radial basis function neural networks
- Sigmoid with parameter κ and θ

$$K(\mathbf{x},\mathbf{y}) = \tanh(\kappa\mathbf{x}^T\mathbf{y} + \theta)$$

  - It does not satisfy the Mercer condition on all κ and θ
- Research on different kernel functions in different applications is *very* active

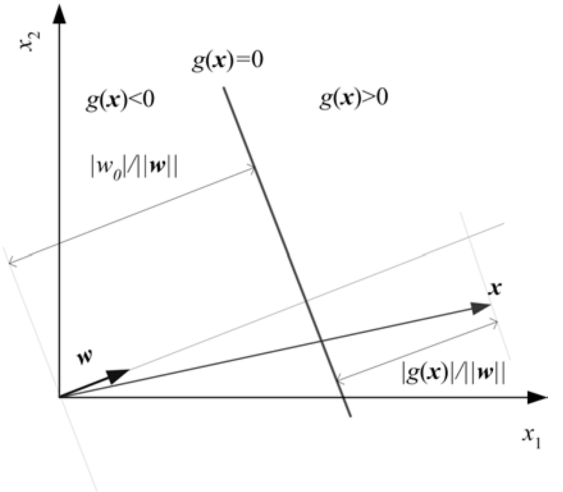
Two Classes



$$
\begin{aligned}
g(\mathbf{x}) &= g_1(\mathbf{x}) - g_2(\mathbf{x}) \\
&= \left(\mathbf{w}_1^T\mathbf{x} + w_{10}\right) - \left(\mathbf{w}_2^T\mathbf{x} + w_{20}\right) \\
&= \left(\mathbf{w}_1 - \mathbf{w}_2\right)^T\mathbf{x} + \left(w_{10} - w_{20}\right) \\
&= \mathbf{w}^T\mathbf{x} + w_0
\end{aligned}
$$

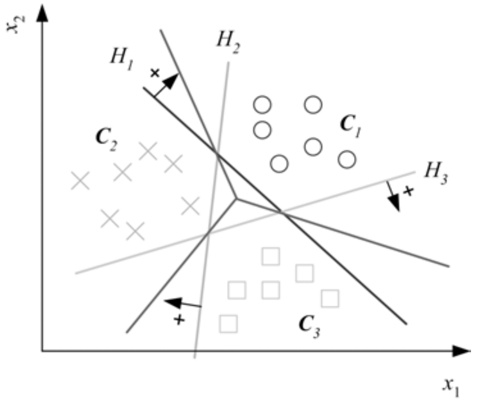
$$
\text{choose} \begin{cases} C_1 & \text{if } g(\mathbf{x}) > 0 \\ C_2 & \text{otherwise} \end{cases}
$$

Geometry



Multiple Classes

$$g_i(\boldsymbol{x} \mid \boldsymbol{w}_i, w_{i0}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0}$$



Choose C_i if

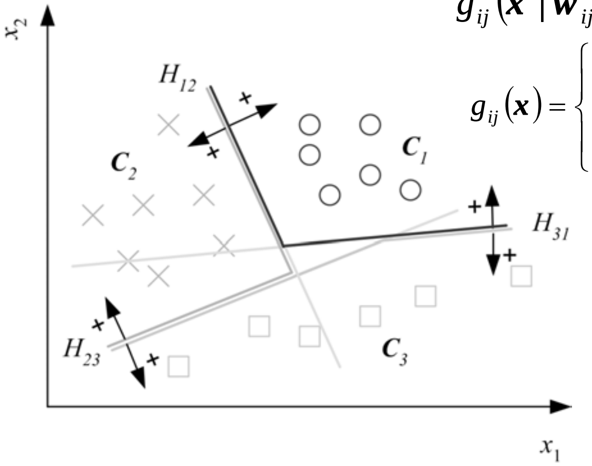
$$g_i(\boldsymbol{x}) = \max_{j=1}^{K} g_j(\boldsymbol{x})$$

Classes are linearly separable

Pairwise Separation

$$g_{ij}(\boldsymbol{x} \mid \boldsymbol{w}_{ij}, w_{ij0}) = \boldsymbol{w}_{ij}^T \boldsymbol{x} + w_{ij0}$$

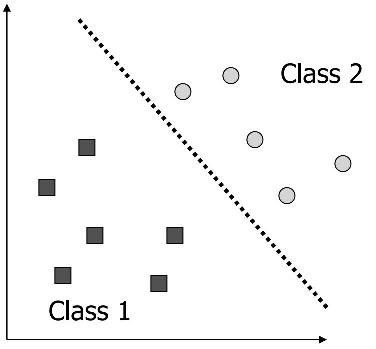
$$g_{ij}(\boldsymbol{x}) = \begin{cases} > 0 & \text{if } \boldsymbol{x} \in C_i \\ \leq 0 & \text{if } \boldsymbol{x} \in C_j \\ \text{don't care} & \text{otherwise} \end{cases}$$



choose C_i if

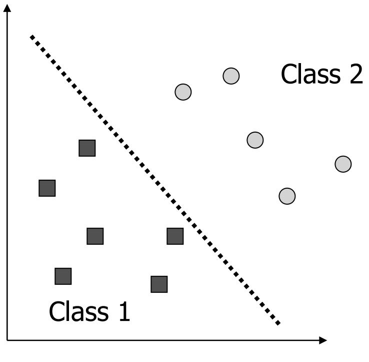
$\forall j \neq i, g_{ij}(\boldsymbol{x}) > 0$

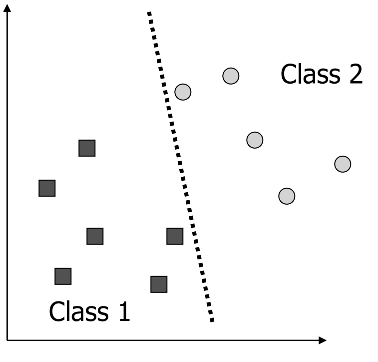
Linear Separable Case



- Many decision boundaries can separate these two classes
- Which one should we choose?

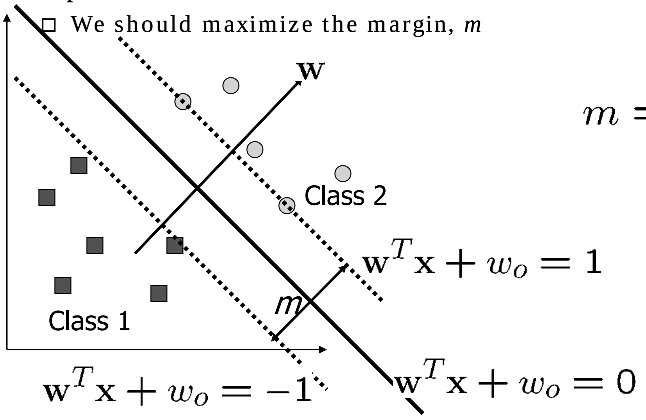
Bad Decision Boundaries





Margin Should Be Large

- The decision boundary should be as far away from the data as possible
  - We should maximize the margin, *m*



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

Optimal Separating Hyperplane

$$\mathcal{X} = \{\boldsymbol{x}^t, r^t\}_t \text{ where } r^t = \begin{cases} +1 & \text{if } \boldsymbol{x}^t \in C_1 \\ -1 & \text{if } \boldsymbol{x}^t \in C_2 \end{cases}$$

find $\boldsymbol{w}$ and w_0 such that

$$\boldsymbol{w}^T\boldsymbol{x}^t + w_0 \geq +1 \text{ for } r^t = +1$$

$$\boldsymbol{w}^T\boldsymbol{x}^t + w_0 \leq +1 \text{ for } r^t = -1$$

which can be rewritten as

$$r^t\left(\boldsymbol{w}^T\boldsymbol{x}^t + w_0\right) \geq +1$$

(Cortes and Vapnik, 1995; Vapnik, 1995)

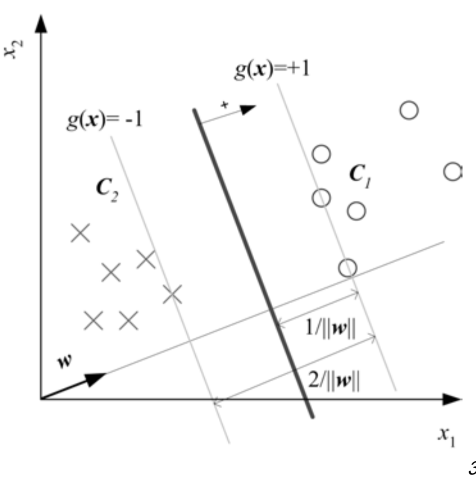
Margin

- Distance from the discriminant to the closest instances on either side
- Distance of x to the hyperplane is $\dfrac{\left|\boldsymbol{w}^T\boldsymbol{x}^t + w_0\right|}{\|\boldsymbol{w}\|}$
- We require $\dfrac{r^t\left(\boldsymbol{w}^T\boldsymbol{x}^t + w_0\right)}{\|\boldsymbol{w}\|} \geq \rho, \forall t$
- For a unique sol'n, fix $\rho\|\boldsymbol{w}\| = 1$ and to max margin

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 \text{ subject to } r^t\left(\boldsymbol{w}^T\boldsymbol{x}^t + w_0\right) \geq +1, \forall t$$

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 \text{ subject to } r^t\left(\boldsymbol{w}^T\boldsymbol{x}^t + w_0\right) \geq +1, \forall t$$

- Standard quadratic optimization problem, whose complexity depends on *d*



Another formulation: complexity depends on N

Unconstrained optimization using Lagrange multipliers α^t. We need to Minimize w.r.t. w, w_o and Maximize w.r.t. $\alpha^t \geq 0$

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) \geq +1, \forall t$$

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{t=1}^{N} \alpha^t \left[r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) - 1\right]$$

$$= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{t=1}^{N} \alpha^t r^t \left(\mathbf{w}^T\mathbf{x}^t + w_0\right) + \sum_{t=1}^{N} \alpha^t$$

This is a quadratic optimization problem. Equivalently, we can solve the dual problem: Maximize L_p w.r.t. α^t subject to the constraints:

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{t=1}^{N} \alpha^t r^t \mathbf{x}^t$$

$$\frac{\partial L_p}{\partial w_0} = 0 \Rightarrow \sum_{t=1}^{N} \alpha^t r^t = 0$$

Maximize

$$L_d = \frac{1}{2}\left(\boldsymbol{w}^T\boldsymbol{w}\right) - \boldsymbol{w}^T \sum_t \alpha^t r^t \boldsymbol{x}^t - w_0 \sum_t \alpha^t r^t + \sum_t \alpha^t$$

$$= \frac{1}{2}\left(\boldsymbol{w}^T\boldsymbol{w}\right) + \sum_t \alpha^t$$

$$= \frac{1}{2}\sum_t \sum_s \alpha^t \alpha^s r^t r^s \left(\boldsymbol{x}^t\right)^T \boldsymbol{x}^s + \sum_t \alpha^t$$

$$\text{subject to } \sum_t \alpha^t r^t = 0 \text{ and } \alpha^t \geq 0, \forall t$$

- This is an optimization problem in α^t only
- This can be solved using quadratic optimization.
- This depends on the sample size *N* and not on the the input dimension *d* Very important feature!! Why ?
- Most α^t are 0 and only a small number have $\alpha^t > 0$; they are the support vectors

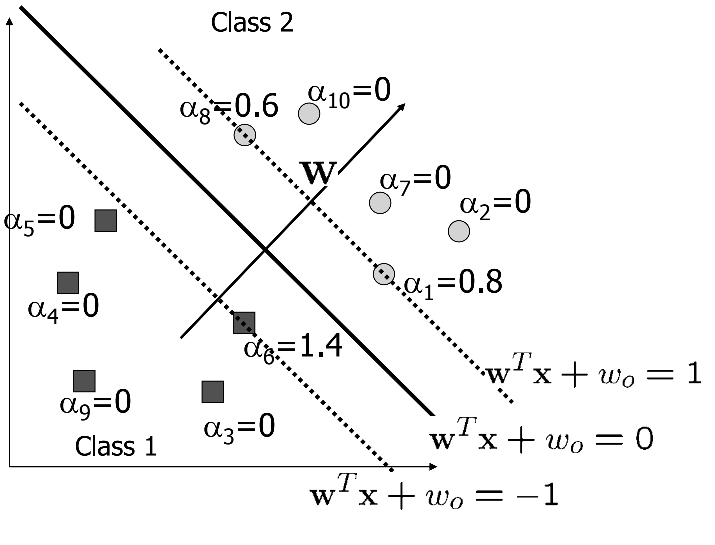
Characteristics of the Solution

- Many of the α^t are zero
  - **w** is a linear combination of a small number of examples
  - Sparse representation
- $\mathbf{x}^t$ with non-zero α^t are called support vectors (SV)
  - The decision boundary is determined only by the SV
  - Let t_j ($j=1, ..., s$) be the indices of the s support vectors. We can write

$$\mathbf{w} = \sum_{j=1}^{s} \alpha^j r^j \mathbf{x}^j \qquad w_o = r^j - \mathbf{w}^T \mathbf{x}^j$$

- For testing with a new data **z**
  - Comput $\mathbf{w}^T\mathbf{z} + w_o = \sum_{j=1}^{s} \alpha^j r^j (\mathbf{x}^{j^T}\mathbf{z}) + w_o$
  - and classify **z** as class 1 if the sum is positive, and class 2 otherwise

A Geometric Interpretation



Soft Margin Hyperplane

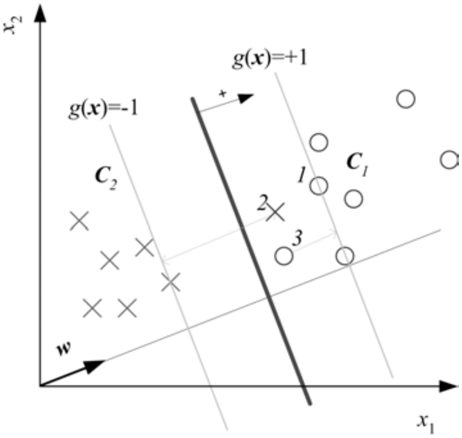
- Not linearly separable: use slack variables

  $$r^t\left(\boldsymbol{w}^T x^t + w_0\right) \geq 1 - \xi^t$$

- Soft error

  $$\sum_t \xi^t$$

- New primal is



$$L_p = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_t \xi^t - \sum_t \alpha^t\left[r^t\left(\boldsymbol{w}^T x^t + w_0\right) - 1 + \xi^t\right] - \sum_t \mu^t \xi^t$$

The Optimization Problem

- The dual of the problem is

$$\text{max. } L(\boldsymbol{\alpha}) = \sum_t \alpha^t - \frac{1}{2}\sum_{t,s}^{n} \alpha^t \alpha^s r^t r^s (\mathbf{x}^t)^T \mathbf{x}^s$$

$$\text{subject to } C \geq \alpha^t \geq 0, \sum_t \alpha^t r^t = 0$$

- **w** is also recovered as $\mathbf{w} = \sum_{j=1}^{s} \alpha^j r^j \mathbf{x}^j$
- The only difference with the linearly separable case is that there is an upper bound C on α^t
- Once again, a QP solver can be used to find α^i

Kernel Machines

- Preprocess input $\boldsymbol{x}$ by basis functions

$$\boldsymbol{z} = \varphi(\boldsymbol{x}) \qquad g(\boldsymbol{z}) = \boldsymbol{w}^T \boldsymbol{z}$$
$$g(\boldsymbol{x}) = \boldsymbol{w}^T \varphi(\boldsymbol{x})$$

- The SVM solution

$$\boldsymbol{w} = \sum_t \alpha^t r^t \boldsymbol{z}^t = \sum_t \alpha^t r^t \boldsymbol{\varphi}(\boldsymbol{x}^t)$$
$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\varphi}(\boldsymbol{x}) = \sum_t \alpha^t r^t \boxed{\boldsymbol{\varphi}(\boldsymbol{x}^t)^T \boldsymbol{\varphi}(\boldsymbol{x})}$$
$$g(\boldsymbol{x}) = \sum_t \alpha^t r^t \boxed{K(\boldsymbol{x}^t, \boldsymbol{x})}$$

Kernel Functions

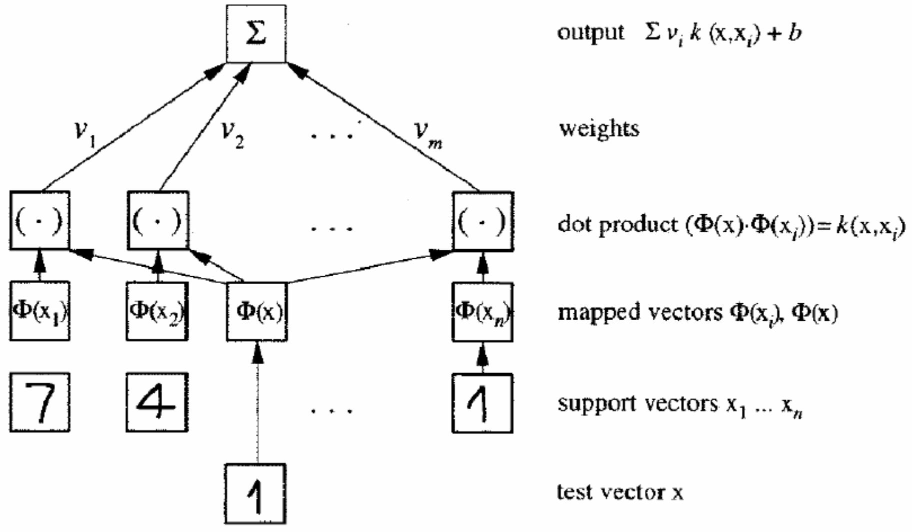
- Polynomials of degree q: $K(\boldsymbol{x}^t, \boldsymbol{x}) = (\boldsymbol{x}^T \boldsymbol{x}^t + 1)^q$

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^T \boldsymbol{y} + 1)^2$$
$$= (x_1 y_1 + x_2 y_2 + 1)^2$$
$$= 1 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 x_2 y_1 y_2 + x_1^2 y_1^2 + x_2^2 y_2^2$$
$$\phi(\boldsymbol{x}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2]^T$$

- Radial-basis functions: $K(\boldsymbol{x}^t, \boldsymbol{x}) = \exp\left[-\frac{\|\boldsymbol{x}^t - \boldsymbol{x}\|^2}{\sigma^2}\right]$
- Sigmoidal functions: $K(\boldsymbol{x}^t, \boldsymbol{x}) = \tanh(2\boldsymbol{x}^T \boldsymbol{x}^t + 1)$

(Cherkassky and Mulier, 1998)

Handwriting Recognition



Using Kernel Functions

- Change all inner products to kernel functions
- For training,

Original

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0,\ \sum_{i=1}^{n} \alpha_i y_i = 0$$

With kernel function

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } C \geq \alpha_i \geq 0,\ \sum_{i=1}^{n} \alpha_i y_i = 0$$

Using Kernel Functions

- For testing, the new data **z** is classified as Class 1 if $f \geq 0$, and as Class 2 if $f < 0$

Original

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

With kernel function

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

Example

- Suppose we have 5 1D data points
  - $x_1=1$, $x_2=2$, $x_3=4$, $x_4=5$, $x_5=6$, with 1, 2, 6 as class 1 and 4, 5 as class 2 $\Rightarrow$ $y_1=1$, $y_2=1$, $y_3=-1$, $y_4=-1$, $y_5=1$
- We use the polynomial kernel of degree 2
  - $K(x,y) = (xy+1)^2$
  - C is set to 100
- We first find α_i ($i=1, \ldots, 5$) by

$$\max. \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$$

Example

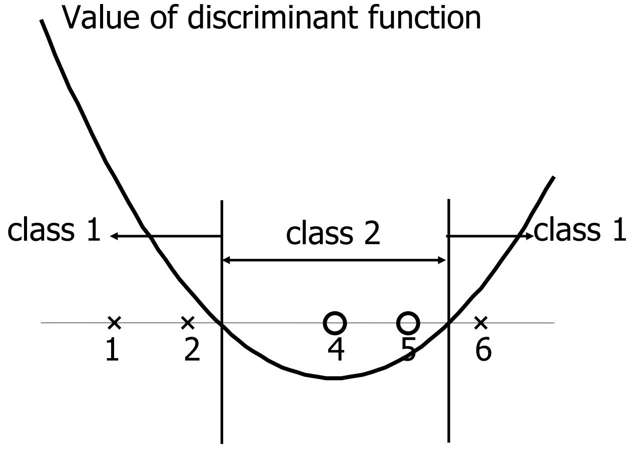
- By using a QP solver, we get
  - $\alpha_1=0$, $\alpha_2=2.5$, $\alpha_3=0$, $\alpha_4=7.333$, $\alpha_5=4.833$
  - Note that the constraints are indeed satisfied
  - The support vectors are $\{x_2=2, x_4=5, x_5=6\}$
- The discriminant function is

$$f(y) = 2.5(1)(2y+1)^2 + 7.333(-1)(5y+1)^2 + 4.833(1)(6y+1)^2 + b$$
$$= 0.6667x^2 - 5.333x + b$$

- b is recovered by solving f(2)=1 or by f(5)=-1 or by f(6)=1, as x_2, x_4, x_5 lie on $y_i(\mathbf{w}^T\phi(z)+b)=1$ and all give $b=9$

$$\Longrightarrow \quad f(y) = 0.6667x^2 - 5.333x + 9$$

Example



Multi-class Classification

- SVM is basically a two-class classifier
- One can change the QP formulation to allow multi-class classification
- More commonly, the data set is divided into two parts "intelligently" in different ways and a separate SVM is trained for each way of division
- Multi-class classification is done by combining the output of all the SVM classifiers
  - Majority rule
  - Error correcting code
  - Directed acyclic graph

Software

- A list of SVM implementations can be found at http://www.kernel-machines.org/software.html
- Some implementations (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementations of SVM
- Several Matlab toolboxes for SVM are also available

Lecture Notes for E Alpaydın 2004 Introduction to Machine Learning © The MIT Press (V1.1)

Steps for Classification

- Prepare the pattern matrix
- Select the kernel function to use
- Select the parameter of the kernel function and the value of *C*
  - You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter
- Execute the training algorithm and obtain the α_i values
- Unseen data can be classified using the α_i values and the support vectors

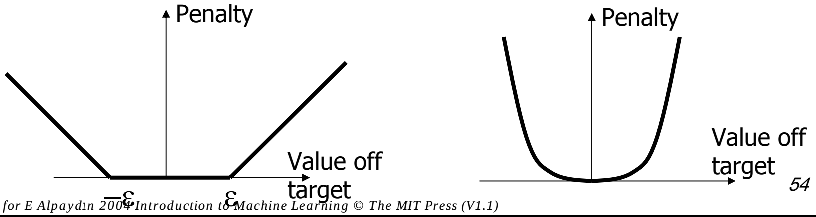
SVM Strengths & Weaknesses

- Strengths
  - Training is relatively easy
    - No local optimal, unlike in neural networks
  - It scales relatively well to high dimensional data
  - Tradeoff between classifier complexity and error can be controlled explicitly
  - Non-traditional data like strings and trees can be used as input to SVM, instead of feature vectors
- Weaknesses
  - Need a “good” kernel function

ε *Support Vector Regression*

- Linear regression in feature space
- Unlike in least square regression, the error function is ε-insensitive loss function
  - Intuitively, mistake less than ε is ignored
  - This leads to sparsity similar to SVM

ε-insensitive loss function　　　　Square loss function



ε *Support Vector Regression*

- Given: a data set $\{x_1, \ldots, x_n\}$ with target values $\{u_1, \ldots, u_n\}$, we want to do ε-SVR
- The optimization problem is

$$\text{Min } \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

$$\text{subject to } \begin{cases} u_i - \mathbf{w}^T\mathbf{x}_i - b \le \epsilon + \xi_i \\ \mathbf{w}^T\mathbf{x}_i + b - u_i \le \epsilon + \xi_i^* \\ \xi_i \ge 0, \xi_i^* \ge 0 \end{cases}$$

- Similar t… programming problem

ε Support Vector Regression

- C is a parameter to control the amount of influence of the error
- The $||w||^2$ term serves as controlling the complexity of the regression function
  - This is similar to ridge regression
- After training (solving the QP), we get values of α_i and α_i^*, which are both zero if $\mathbf{x}_i$ does not contribute to the error function
- For a new instance $\mathbf{z}$,

$$f(\mathbf{z}) = \sum_{j=1}^{s} (\alpha_{t_j} - \alpha_{t_j}^*) K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

Other Kernel Methods

- A lesson learned in SVM: a linear algorithm in the feature space is equivalent to a non-linear algorithm in the input space
- Classic linear algorithms can be generalized to its non-linear version by going to the feature space
  - Kernel principal component analysis, kernel independent component analysis, kernel canonical correlation analysis, kernel k-means, 1-class SVM are some examples

Conclusion

- SVM is a useful method for classification
- Two key concepts of SVM: maximize the margin and the kernel trick
- Much active research is taking place on areas related to SVM
- Many SVM implementations are available on the web for you to try on your data set!

Resources

- http://www.kernel-machines.org/
- http://www.support-vector.net/
- http://www.support-vector.net/icml-tutorial.pdf
- http://www.kernel-machines.org/papers/tutorial-nips.ps.gz
- http://www.clopinet.com/isabelle/Projects/SVM/applist.html

SVM for Regression

- Use a linear model (possibly kernelized)

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0$$

- Use the ϵ-sensitive error function

$$e_\varepsilon\left(r^t, f\left(\boldsymbol{x}^t\right)\right) = \begin{cases} 0 & \text{if } \left|r^t - f\left(\boldsymbol{x}^t\right)\right| < \varepsilon \\ \left|r^t - f\left(\boldsymbol{x}^t\right)\right| - \varepsilon & \text{otherwise} \end{cases}$$

- $$\min \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_t \left(\xi_+^t + \xi_-^t\right)$$

$$r^t - \left(\boldsymbol{w}^T\boldsymbol{x} + w_0\right) \le \varepsilon + \xi_+^t$$

$$\left(\boldsymbol{w}^T\boldsymbol{x} + w_0\right) - r^t \le \varepsilon + \xi_-^t$$

$$\xi_+^t, \xi_-^t \ge 0$$