



# CMSC726 Spring 2006: Support Vector Machines

readings: available from class page  
sources:  
Andrew Moore <http://www.cs.cmu.edu/~awm/tutorials>  
Tom Dietterich, Andrew Ng, Michael Littman, Rich Maclin




## 3 Views

- Geometric
  - Maximizing Margin
- Kernel Methods
  - Making nonlinear decision boundaries linear
  - Efficiently!
- Capacity
  - Structural Risk Minimization

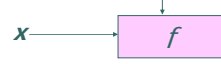


## SVM History

- SVM is a classifier derived from statistical learning theory by Vapnik and Chervonenkis
- SVM was first introduced by Boser, Guyon and Vapnik in COLT-92
- SVM became famous when, using pixel maps as input, it gave accuracy comparable to NNs with hand-designed features in a handwriting recognition task
- SVM is closely related to:
  - Kernel machines (a generalization of SVMs), large margin classifiers, reproducing kernel Hilbert space, Gaussian process, Boosting

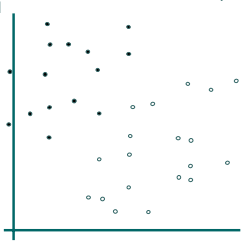


## Linear Classifiers

$\alpha$   

 $x \rightarrow f \rightarrow y^{est}$


$f(x, w, b) = \text{sign}(w \cdot x - b)$

- denotes +1
- denotes -1

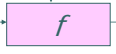


How would you classify this data?

Copyright © 2001, 2003, Andrew W. Moore

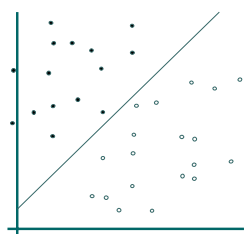


## Linear Classifiers

$\alpha$   

 $x \rightarrow f \rightarrow y^{est}$


$f(x, w, b) = \text{sign}(w \cdot x - b)$

- denotes +1
- denotes -1

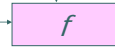


How would you classify this data?

Copyright © 2001, 2003, Andrew W. Moore

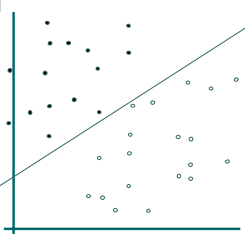


## Linear Classifiers

$\alpha$   

 $x \rightarrow f \rightarrow y^{est}$

$f(x, w, b) = \text{sign}(w \cdot x - b)$

- denotes +1
- denotes -1



How would you classify this data?

Copyright © 2001, 2003, Andrew W. Moore

### Linear Classifiers

$x \rightarrow f \rightarrow y_{est}$

$f(x, w, b) = \text{sign}(w \cdot x - b)$

- denotes +1
- denotes -1

How would you classify this data?

Copyright © 2001, 2003, Andrew W. Moore

### Linear Classifiers

$x \rightarrow f \rightarrow y_{est}$

$f(x, w, b) = \text{sign}(w \cdot x - b)$

- denotes +1
- denotes -1

Any of these would be fine..

..but which is best?

Copyright © 2001, 2003, Andrew W. Moore

### Classifier Margin

$x \rightarrow f \rightarrow y_{est}$

$f(x, w, b) = \text{sign}(w \cdot x - b)$

- denotes +1
- denotes -1

Define the **margin** of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Copyright © 2001, 2003, Andrew W. Moore

### Maximum Margin

$x \rightarrow f \rightarrow y_{est}$

$f(x, w, b) = \text{sign}(w \cdot x - b)$

- denotes +1
- denotes -1

The maximum margin linear classifier is the linear classifier with the maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Copyright © 2001, 2003, Andrew W. Moore

### Maximum Margin

$x \rightarrow f \rightarrow y_{est}$

$f(x, w, b) = \text{sign}(w \cdot x - b)$

- denotes +1
- denotes -1

The maximum margin linear classifier is the linear classifier with the maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Support Vectors are those datapoints that the margin pushes up against

Copyright © 2001, 2003, Andrew W. Moore

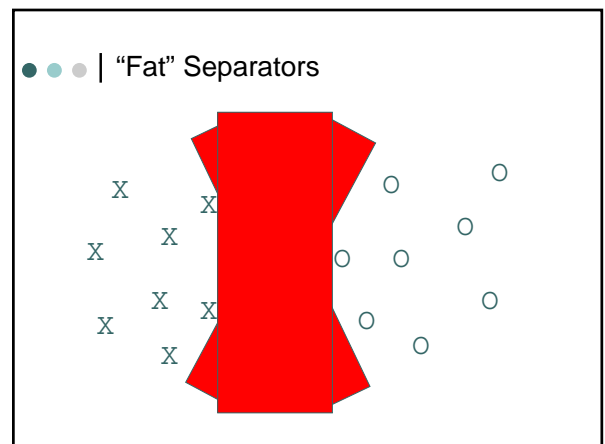
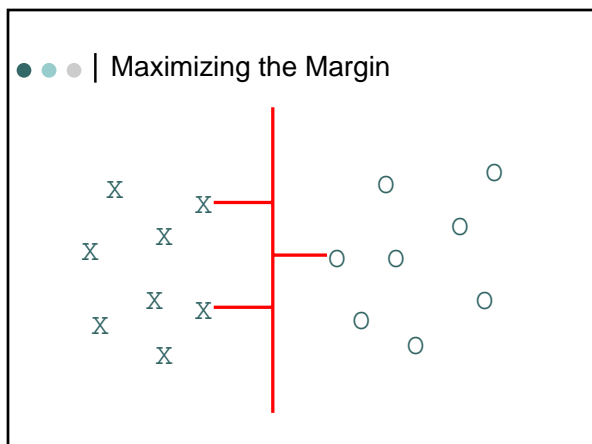
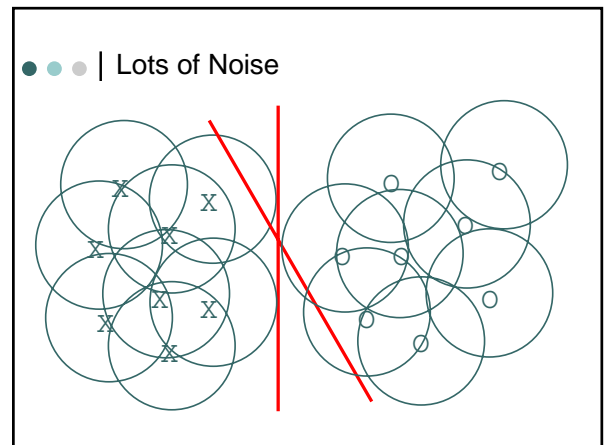
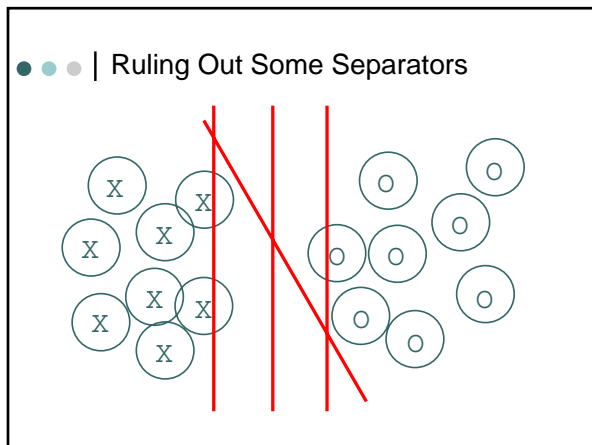
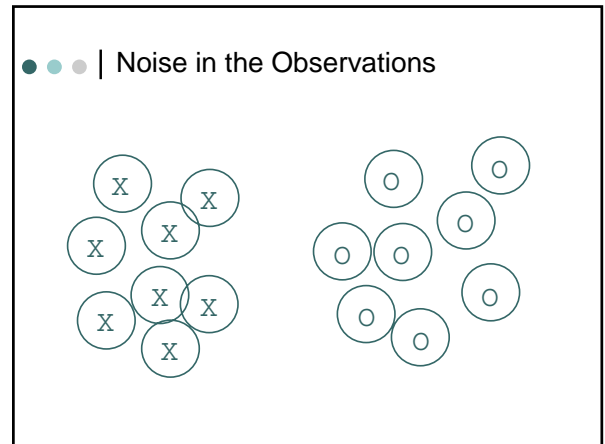
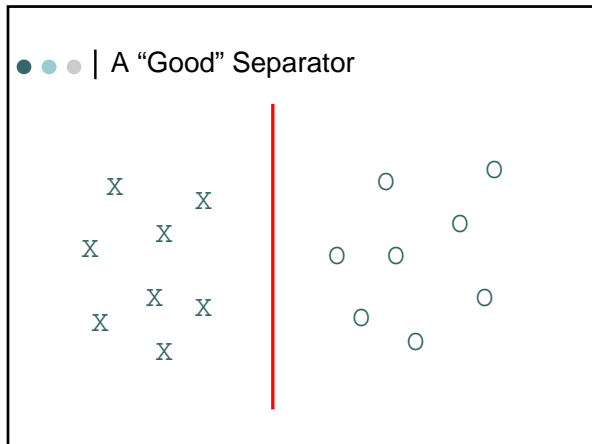
### Why Maximum Margin?

- denotes +1
- denotes -1

Support Vectors are those datapoints that the margin pushes up against

1. Intuitively this feels safest.
2. If we've made a small error in the location of the boundary (it's been jolted in its perpendicular direction) this gives us least chance of causing a misclassification.
3. LOOCV is easy since the model is immune to removal of any non-support-vector datapoints.
4. There's some theory (using VC dimension) that is related to (but not the same as) the proposition that this is a good thing.
5. Empirically it works very very well.

Copyright © 2001, 2003, Andrew W. Moore



### ● ● ● | Specifying a line and margin

- How do we represent this mathematically?
- ...in  $m$  input dimensions?

Copyright © 2001, 2003, Andrew W. Moore

### ● ● ● | Specifying a line and margin

- Plus-plane =  $\{x : w \cdot x + b = +1\}$
- Minus-plane =  $\{x : w \cdot x + b = -1\}$

Classify as..

+1	if	$w \cdot x + b \geq 1$
-1	if	$w \cdot x + b \leq -1$
Universe explodes	if	$-1 < w \cdot x + b < 1$

Copyright © 2001, 2003, Andrew W. Moore

### ● ● ● | Computing the margin width

- Plus-plane =  $\{x : w \cdot x + b = +1\}$
- Minus-plane =  $\{x : w \cdot x + b = -1\}$

Claim: The vector  $w$  is perpendicular to the Plus Plane. Why?

Copyright © 2001, 2003, Andrew W. Moore

### ● ● ● | Computing the margin width

- Plus-plane =  $\{x : w \cdot x + b = +1\}$
- Minus-plane =  $\{x : w \cdot x + b = -1\}$

Claim: The vector  $w$  is perpendicular to the Plus Plane. Why?

And so of course the vector  $w$  is also perpendicular to the Minus Plane. What is  $w \cdot (u - v)$ ?

Copyright © 2001, 2003, Andrew W. Moore

### ● ● ● | Computing the margin width

- Plus-plane =  $\{x : w \cdot x + b = +1\}$
- Minus-plane =  $\{x : w \cdot x + b = -1\}$
- The vector  $w$  is perpendicular to the Plus Plane
- Let  $x^-$  be any point on the minus plane
- Let  $x^+$  be the closest plus-plane-point to  $x^-$ .

Any location in  $\mathbb{R}^m$ : not necessarily a datapoint

Copyright © 2001, 2003, Andrew W. Moore

### ● ● ● | Computing the margin width

- Plus-plane =  $\{x : w \cdot x + b = +1\}$
- Minus-plane =  $\{x : w \cdot x + b = -1\}$
- The vector  $w$  is perpendicular to the Plus Plane
- Let  $x^-$  be any point on the minus plane
- Let  $x^+$  be the closest plus-plane-point to  $x^-$ .
- Claim:  $x^+ = x^- + \lambda w$  for some value of  $\lambda$ . Why?

Copyright © 2001, 2003, Andrew W. Moore

### Computing the margin width

How do we compute  $M$  in terms of  $w$  and  $b$ ?

The line from  $x^*$  to  $x^+$  is perpendicular to the planes.  
So to get from  $x^*$  to  $x^+$  travel some distance in direction  $w$ .

- Plus-plane =  $\{x : w \cdot x + b = 1\}$
- Minus-plane =  $\{x : w \cdot x + b = -1\}$
- The vector  $w$  is perpendicular to the planes
- Let  $x^*$  be any point on the minus plane
- Let  $x^+$  be the closest plus-plane-point to  $x^*$ .
- Claim:  $x^+ = x^* + \lambda w$  for some value of  $\lambda$ . Why?

Copyright © 2001, 2003, Andrew W. Moore

### Computing the margin width

What we know:

- $w \cdot x^+ + b = +1$
- $w \cdot x^* + b = -1$
- $x^+ = x^* + \lambda w$
- $|x^+ - x^*| = M$

It's now easy to get  $M$  in terms of  $w$  and  $b$

Copyright © 2001, 2003, Andrew W. Moore

### Computing the margin width

What we know:

- $w \cdot x^+ + b = +1$
- $w \cdot x^* + b = -1$
- $x^+ = x^* + \lambda w$
- $|x^+ - x^*| = M$

It's now easy to get  $M$  in terms of  $w$  and  $b$

$$w \cdot (x^* + \lambda w) + b = 1$$

$$\Rightarrow w \cdot x^* + b + \lambda w \cdot w = 1$$

$$\Rightarrow -1 + \lambda w \cdot w = 1$$

$$\Rightarrow \lambda = \frac{2}{w \cdot w}$$

Copyright © 2001, 2003, Andrew W. Moore

### Computing the margin width

What we know:

- $w \cdot x^+ + b = +1$
- $w \cdot x^* + b = -1$
- $x^+ = x^* + \lambda w$
- $|x^+ - x^*| = M$

$$M = |x^+ - x^*| = |\lambda w| = \lambda |w| = \lambda \sqrt{w \cdot w}$$

$$= \frac{2\sqrt{w \cdot w}}{w \cdot w} = \frac{2}{\sqrt{w \cdot w}}$$

Copyright © 2001, 2003, Andrew W. Moore

### Learning the Maximum Margin Classifier

Given a guess of  $w$  and  $b$  we can

- Compute whether all data points are in the correct half-planes
- Compute the width of the margin

So now we just need to write a program to search the space of  $w$ 's and  $b$ 's to find the widest margin that matches all the datapoints.

How?

Gradient descent? Simulated Annealing? Matrix Inversion? EM? Newton's Method?

Copyright © 2001, 2003, Andrew W. Moore

### Useful Stuff

Don't worry... it's good for you...

- Linear Programming

find  $w$   
 $\text{argmax } c \cdot w$   
 subject to  
 $w \cdot a_i \leq b_i$ , for  $i = 1, \dots, m$   
 $w_j \geq 0$  for  $j = 1, \dots, n$

There are fast algorithms for solving linear programs including the simplex algorithm and Karmarkar's algorithm

● ● ● | Learning via Quadratic Programming

- QP is a well-studied class of optimization algorithms to maximize a quadratic function of some real-valued variables subject to linear constraints.

Copyright © 2001, 2003, Andrew W. Moore

● ● ● | Quadratic Programming

Find  $\arg \max_{\mathbf{u}} c + \mathbf{d}^T \mathbf{u} + \frac{\mathbf{u}^T \mathbf{R} \mathbf{u}}{2}$  ← Quadratic criterion

Subject to  $\left. \begin{aligned} a_{11}u_1 + a_{12}u_2 + \dots + a_{1m}u_m &\leq b_1 \\ a_{21}u_1 + a_{22}u_2 + \dots + a_{2m}u_m &\leq b_2 \\ &\vdots \\ a_{n1}u_1 + a_{n2}u_2 + \dots + a_{nm}u_m &\leq b_n \end{aligned} \right\} n \text{ additional linear inequality constraints}$

And subject to  $\left. \begin{aligned} a_{(n+1)1}u_1 + a_{(n+1)2}u_2 + \dots + a_{(n+1)m}u_m &= b_{(n+1)} \\ a_{(n+2)1}u_1 + a_{(n+2)2}u_2 + \dots + a_{(n+2)m}u_m &= b_{(n+2)} \\ &\vdots \\ a_{(n+e)1}u_1 + a_{(n+e)2}u_2 + \dots + a_{(n+e)m}u_m &= b_{(n+e)} \end{aligned} \right\} e \text{ additional linear equality constraints}$

Copyright © 2001, 2003, Andrew W. Moore

● ● ● | Quadratic Programming

Find  $\arg \max_{\mathbf{u}} c + \mathbf{d}^T \mathbf{u} + \frac{\mathbf{u}^T \mathbf{R} \mathbf{u}}{2}$  ← Quadratic criterion

Subject to  $\left. \begin{aligned} a_{11}u_1 + a_{12}u_2 + \dots + a_{1m}u_m &\leq b_1 \\ a_{21}u_1 + a_{22}u_2 + \dots + a_{2m}u_m &\leq b_2 \\ &\vdots \\ a_{n1}u_1 + a_{n2}u_2 + \dots + a_{nm}u_m &\leq b_n \end{aligned} \right\} n \text{ additional linear inequality constraints}$

And subject to  $\left. \begin{aligned} a_{(n+1)1}u_1 + a_{(n+1)2}u_2 + \dots + a_{(n+1)m}u_m &= b_{(n+1)} \\ a_{(n+2)1}u_1 + a_{(n+2)2}u_2 + \dots + a_{(n+2)m}u_m &= b_{(n+2)} \\ &\vdots \\ a_{(n+e)1}u_1 + a_{(n+e)2}u_2 + \dots + a_{(n+e)m}u_m &= b_{(n+e)} \end{aligned} \right\} e \text{ additional linear equality constraints}$

There exist algorithms for finding such constrained quadratic optima much more efficiently and reliably than gradient ascent. (But they are very fiddly...you probably don't want to write one yourself)

Copyright © 2001, 2003, Andrew W. Moore

● ● ● | Learning the Maximum Margin Classifier

Given guess of  $\mathbf{w}, b$  we can

- Compute whether all data points are in the correct half-planes
- Compute the margin width

Assume  $R$  datapoints, each  $(\mathbf{x}_k, y_k)$  where  $y_k = +/- 1$

What should our quadratic optimization criterion be? **Minimize  $\mathbf{w} \cdot \mathbf{w}$**

How many constraints will we have?  $R$

What should they be?

$\mathbf{w} \cdot \mathbf{x}_k + b \geq 1$  if  $y_k = 1$   
 $\mathbf{w} \cdot \mathbf{x}_k + b \leq -1$  if  $y_k = -1$

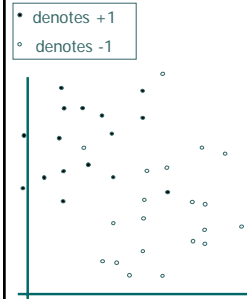
Copyright © 2001, 2003, Andrew W. Moore

● ● ● |

Yaye, we're done!!

● ● ● | Uh-oh!

This is going to be a problem!  
What should we do?



Copyright © 2001, 2003, Andrew W. Moore

Uh-oh!

• denotes +1  
○ denotes -1

This is going to be a problem!  
What should we do?

Idea 1:  
Find minimum  $w \cdot w$ , while minimizing number of training set errors.

Problem: Two things to minimize makes for an ill-defined optimization

Copyright © 2001, 2003, Andrew W. Moore

Uh-oh!

• denotes +1  
○ denotes -1

This is going to be a problem!  
What should we do?

Idea 1.1:  
Minimize  $w \cdot w + C (\#train\ errors)$

Tradeoff parameter

There's a serious practical problem that's about to make us reject this approach. Can you guess what it is?

Copyright © 2001, 2003, Andrew W. Moore

Uh-oh!

• denotes +1  
○ denotes -1

This is going to be a problem!  
What should we do?

Idea 1.1:  
Minimize  $w \cdot w + C (\#train\ errors)$

Tradeoff parameter

Can't be expressed as a Quadratic Programming problem.  
Solving it may be too slow.  
(Also, doesn't distinguish between disastrous errors and near misses)

serious practical problem to make us reject this approach.

So... any other ideas?

Copyright © 2001, 2003, Andrew W. Moore

Uh-oh!

• denotes +1  
○ denotes -1

This is going to be a problem!  
What should we do?

Idea 2.0:  
Minimize  $w \cdot w + C (\text{distance of error points to their correct place})$

Copyright © 2001, 2003, Andrew W. Moore

Learning Maximum Margin with Noise

Given guess of  $w, b$  we can

- Compute sum of distances of points to their correct zones
- Compute the margin width

Assume  $R$  datapoints, each  $(x_k, y_k)$  where  $y_k = +/- 1$

What should our quadratic optimization criterion be?

How many constraints will we have?

What should they be?

Copyright © 2001, 2003, Andrew W. Moore

Learning Maximum Margin with Noise

Given guess of  $w, b$  we can

- Compute sum of distances of points to their correct zones
- Compute the margin width

Assume  $R$  datapoints, each  $(x_k, y_k)$  where  $y_k = +/- 1$

What should our quadratic optimization criterion be?

How many constraints will we have?

What should they be?

Copyright © 2001, 2003, Andrew W. Moore

### Learning Maximum Margin with Noise

Given guess of  $\mathbf{w}$ ,  $b$  we can

- Compute sum of distances of points to their correct zones
- Compute the margin width

Assume  $R$  datapoints, each  $(\mathbf{x}_k, y_k)$  where  $y_k = +/- 1$

What should our quadratic optimization criterion be?

Minimize 
$$\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{k=1}^R \varepsilon_k$$

How many constraints will we have?  $R$

What should they be?

$\mathbf{w} \cdot \mathbf{x}_k + b \geq 1 - \varepsilon_k$  if  $y_k = 1$   
 $\mathbf{w} \cdot \mathbf{x}_k + b \leq -1 + \varepsilon_k$  if  $y_k = -1$

Copyright © 2001, 2003, Andrew W. Moore

### Learning Maximum Margin with Noise

Our original (noiseless data) QP had  $m+1$  variables:  $w_1, w_2, \dots, w_m, b$ .

Our new (noisy data) QP has  $m+1+R$  variables:  $w_1, w_2, \dots, w_m, b, \varepsilon_1, \varepsilon_2, \dots, \varepsilon_R$

What should our quadratic optimization criterion be?

Minimize 
$$\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{k=1}^R \varepsilon_k$$

How many constraints will we have?  $R$

What should they be?

$\mathbf{w} \cdot \mathbf{x}_k + b \geq 1 - \varepsilon_k$  if  $y_k = 1$   
 $\mathbf{w} \cdot \mathbf{x}_k + b \leq -1 + \varepsilon_k$  if  $y_k = -1$

Copyright © 2001, 2003, Andrew W. Moore

### Learning Maximum Margin with Noise

Given guess of  $\mathbf{w}$ ,  $b$  we can

- Compute sum of distances of points to their correct zones
- Compute the margin width

Assume  $R$  datapoints, each  $(\mathbf{x}_k, y_k)$  where  $y_k = +/- 1$

What should our quadratic optimization criterion be?

Minimize 
$$\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{k=1}^R \varepsilon_k$$

How many constraints will we have?  $R$

What should they be?

$\mathbf{w} \cdot \mathbf{x}_k + b \geq 1 - \varepsilon_k$  if  $y_k = 1$   
 $\mathbf{w} \cdot \mathbf{x}_k + b \leq -1 + \varepsilon_k$  if  $y_k = -1$

There's a bug in this QP. Can you spot it?

Copyright © 2001, 2003, Andrew W. Moore

### Learning Maximum Margin with Noise

Given guess of  $\mathbf{w}$ ,  $b$  we can

- Compute sum of distances of points to their correct zones
- Compute the margin width

Assume  $R$  datapoints, each  $(\mathbf{x}_k, y_k)$  where  $y_k = +/- 1$

What should our quadratic optimization criterion be?

Minimize 
$$\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{k=1}^R \varepsilon_k$$

How many constraints will we have?  $2R$

What should they be?

$\mathbf{w} \cdot \mathbf{x}_k + b \geq 1 - \varepsilon_k$  if  $y_k = 1$   
 $\mathbf{w} \cdot \mathbf{x}_k + b \leq -1 + \varepsilon_k$  if  $y_k = -1$   
 $\varepsilon_k \geq 0$  for all  $k$

Copyright © 2001, 2003, Andrew W. Moore

# Yaye, we're done!!

### An Equivalent Dual QP

Maximize 
$$\sum_{k=1}^R \alpha_k - \frac{1}{2} \sum_{k=1}^R \sum_{l=1}^R \alpha_k \alpha_l Q_{kl}$$
 where  $Q_{kl} = y_k y_l (\mathbf{x}_k \cdot \mathbf{x}_l)$

Subject to these constraints:  $0 \leq \alpha_k \leq C \quad \forall k \quad \sum_{k=1}^R \alpha_k y_k = 0$

Then define:

$$\mathbf{w} = \sum_{k=1}^R \alpha_k y_k \mathbf{x}_k$$

$$b = y_K (1 - \varepsilon_K) - \mathbf{x}_K \cdot \mathbf{w}_K$$

where  $K = \arg \max_k \alpha_k$

Then classify with:

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

Copyright © 2001, 2003, Andrew W. Moore

### An Equivalent Dual QP

Maximize  $\sum_{k=1}^R \alpha_k - \frac{1}{2} \sum_{k=1}^R \sum_{l=1}^R \alpha_k \alpha_l Q_{kl}$  where  $Q_{kl} = y_k y_l (\mathbf{x}_k \cdot \mathbf{x}_l)$

Subject to these constraints:  $0 \leq \alpha_k \leq C \quad \forall k \quad \sum_{k=1}^R \alpha_k y_k = 0$

Then define:

$$\mathbf{w} = \sum_{k=1}^R \alpha_k y_k \mathbf{x}_k$$

$$b = y_K (1 - \epsilon_K) - \mathbf{x}_K \cdot \mathbf{w}$$

where  $K = \arg \max_k \alpha_k$

Datapoints with  $\alpha_k > 0$  will be the support vectors

$f(\mathbf{x}; \mathbf{w}, b) = \text{sign}(\mathbf{x} \cdot \mathbf{w} - b)$

...so this sum only needs to be over the support vectors.

Copyright © 2001, 2003, Andrew W. Moore

### An Equivalent Dual QP

Maximize  $\sum_{k=1}^R \alpha_k - \frac{1}{2} \sum_{k=1}^R \sum_{l=1}^R \alpha_k \alpha_l Q_{kl}$  where  $Q_{kl} = y_k y_l (\mathbf{x}_k \cdot \mathbf{x}_l)$

Subject to these constraints:  $0 \leq \alpha_k \leq C \quad \forall k \quad \sum_{k=1}^R \alpha_k y_k = 0$

Then define:

$$\mathbf{w} = \sum_{k=1}^R \alpha_k y_k \mathbf{x}_k$$

$$b = y_K (1 - \epsilon_K) - \mathbf{x}_K \cdot \mathbf{w}$$

where  $K = \arg \max_k \alpha_k$

Why did I tell you about this equivalent QP?

- It's a formulation that QP packages can optimize more quickly
- Because of further developments you're about to learn.

Copyright © 2001, 2003, Andrew W. Moore

### Suppose we're in 1-dimension

What would SVMs do with this data?

Copyright © 2001, 2003, Andrew W. Moore

### Suppose we're in 1-dimension

Not a big surprise

Copyright © 2001, 2003, Andrew W. Moore

### Harder 1-dimensional dataset

That's wiped the smirk off SVM's face.

What can be done about this?

Copyright © 2001, 2003, Andrew W. Moore

### Harder 1-dimensional dataset

Remember how permitting non-linear basis functions made linear regression so much nicer?

Let's permit them here too

$$\mathbf{z}_k = (x_k, x_k^2)$$

Copyright © 2001, 2003, Andrew W. Moore

● ● ● | Harder 1-dimensional dataset

Remember how permitting non-linear basis functions made linear regression so much nicer?  
Let's permit them here too

$$\mathbf{z}_k = (x_k, x_k^2)$$

Copyright © 2001, 2003, Andrew W. Moore

● ● ● | Common SVM basis functions

$\mathbf{z}_k =$  ( polynomial terms of  $\mathbf{x}_k$  of degree 1 to  $q$  )

$\mathbf{z}_k =$  ( radial basis functions of  $\mathbf{x}_k$  )

$$\mathbf{z}_k[l, j] = \varphi_j(\mathbf{x}_k) = \text{KernelFn}\left(\frac{\|\mathbf{x}_k - \mathbf{c}_j\|}{KW}\right)$$

$\mathbf{z}_k =$  ( sigmoid functions of  $\mathbf{x}_k$  )

This is sensible.  
Is that the end of the story?  
No...there's one more trick!

Copyright © 2001, 2003, Andrew W. Moore

● ● ● | Quadratic Basis Functions

$$\Phi(\mathbf{x}) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \sqrt{2}x_2x_3 \\ \vdots \\ \sqrt{2}x_2x_m \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{pmatrix}$$

- Constant Term
- Linear Terms
- Pure Quadratic Terms
- Quadratic Cross-Terms

Number of terms (assuming  $m$  input dimensions) =  $(m+2) \cdot \text{choose-2}$   
 $= (m+2)(m+1)/2$   
 $=$  (as near as makes no difference)  $m^2/2$

You may be wondering what those  $\sqrt{2}$ 's are doing.  
 •You should be happy that they do no harm  
 •You'll find out why they're there soon.

Copyright © 2001, 2003, Andrew W. Moore

● ● ● | QP with basis functions

Maximize  $\sum_{k=1}^R \alpha_k - \frac{1}{2} \sum_{k=1}^R \sum_{l=1}^R \alpha_k \alpha_l Q_{kl}$  where  $Q_{kl} = y_k y_l (\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_l))$

Subject to these constraints:  $0 \leq \alpha_k \leq C \quad \forall k \quad \sum_{k=1}^R \alpha_k y_k = 0$

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(\mathbf{x}_k)$$

$$b = y_K (1 - \varepsilon_K) - \mathbf{x}_K \cdot \mathbf{w}_K$$

where  $K = \arg \max \alpha_k$

Then classify with:

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \Phi(\mathbf{x}) - b)$$

Copyright © 2001, 2003, Andrew W. Moore

● ● ● | QP with basis functions

Maximize  $\sum_{k=1}^R \alpha_k - \frac{1}{2} \sum_{k=1}^R \sum_{l=1}^R \alpha_k \alpha_l Q_{kl}$  where  $Q_{kl} = y_k y_l (\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_l))$

Subject to these constraints:  $0 \leq \alpha_k \leq C$

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(\mathbf{x}_k)$$

$$b = y_K (1 - \varepsilon_K) - \mathbf{x}_K \cdot \mathbf{w}_K$$

where  $K = \arg \max \alpha_k$

We must do  $R^2/2$  dot products to get this matrix ready.  
 Each dot product requires  $m^2/2$  additions and multiplications  
 The whole thing costs  $R^2 m^2 / 4$ .  
 Yeeks!  
 ...or does it?

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \Phi(\mathbf{x}) - b)$$

Copyright © 2001, 2003, Andrew W. Moore

● ● ● | Quadratic Dot Products

$$\Phi(\mathbf{a}) \cdot \Phi(\mathbf{b}) = \begin{pmatrix} 1 \\ \sqrt{2}a_1 \\ \sqrt{2}a_2 \\ \vdots \\ \sqrt{2}a_m \\ a_1^2 \\ a_2^2 \\ \vdots \\ a_m^2 \\ \sqrt{2}a_1a_2 \\ \sqrt{2}a_1a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \sqrt{2}a_2a_3 \\ \vdots \\ \sqrt{2}a_2a_m \\ \vdots \\ \sqrt{2}a_{m-1}a_m \end{pmatrix} \cdot \begin{pmatrix} 1 \\ \sqrt{2}b_1 \\ \sqrt{2}b_2 \\ \vdots \\ \sqrt{2}b_m \\ b_1^2 \\ b_2^2 \\ \vdots \\ b_m^2 \\ \sqrt{2}b_1b_2 \\ \sqrt{2}b_1b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \sqrt{2}b_2b_3 \\ \vdots \\ \sqrt{2}b_2b_m \\ \vdots \\ \sqrt{2}b_{m-1}b_m \end{pmatrix}$$

$\sum_{i=1}^m 2a_i b_i$   
 $\sum_{i=1}^m a_i^2 b_i^2$   
 $\sum_{i=1}^m \sum_{j=i+1}^m 2a_i a_j b_i b_j$

Copyright © 2001, 2003, Andrew W. Moore

### Quadratic Dot Products

Just out of casual, innocent, interest, let's look at another function of  $a$  and  $b$ :

$$\begin{aligned}
 & (a \cdot b + 1)^2 \\
 &= (a \cdot b)^2 + 2a \cdot b + 1 \\
 &= \left( \sum_{i=1}^m a_i b_i \right)^2 + 2 \sum_{i=1}^m a_i b_i + 1 \\
 &= \sum_{i=1}^m a_i b_i a_i b_i + 2 \sum_{i=1}^m a_i b_i + 1 \\
 &= \sum_{i=1}^m (a_i b_i)^2 + 2 \sum_{i=1}^m a_i b_i a_i b_i + 2 \sum_{i=1}^m a_i b_i + 1
 \end{aligned}$$

$\Phi(a) \cdot \Phi(b) =$   
 $1 + 2 \sum_{i=1}^m a_i b_i + \sum_{i=1}^m a_i^2 b_i^2 + \sum_{i=1}^m \sum_{j=i+1}^m 2a_i a_j b_i b_j$

Copyright © 2001, 2003, Andrew W. Moore

### Quadratic Dot Products

Just out of casual, innocent, interest, let's look at another function of  $a$  and  $b$ :

$$\begin{aligned}
 & (a \cdot b + 1)^2 \\
 &= (a \cdot b)^2 + 2a \cdot b + 1 \\
 &= \left( \sum_{i=1}^m a_i b_i \right)^2 + 2 \sum_{i=1}^m a_i b_i + 1 \\
 &= \sum_{i=1}^m \sum_{j=1}^m a_i b_i a_j b_j + 2 \sum_{i=1}^m a_i b_i + 1 \\
 &= \sum_{i=1}^m (a_i b_i)^2 + 2 \sum_{i=1}^m \sum_{j=i+1}^m a_i b_i a_j b_j + 2 \sum_{i=1}^m a_i b_i + 1
 \end{aligned}$$

$\Phi(a) \cdot \Phi(b) =$   
 $1 + 2 \sum_{i=1}^m a_i b_i + \sum_{i=1}^m a_i^2 b_i^2 + \sum_{i=1}^m \sum_{j=i+1}^m 2a_i a_j b_i b_j$

They're the same!  
 And this is only  $O(m)$  to compute!

Copyright © 2001, 2003, Andrew W. Moore

### QP with Quadratic basis functions

Maximize  $\sum_{k=1}^R \alpha_k - \frac{1}{2} \sum_{k=1}^R \sum_{l=1}^R \alpha_k \alpha_l Q_{kl}$  where  $Q_{kl} = y_k y_l (\Phi(x_k) \cdot \Phi(x_l))$

Subject to these constraints:  $0 \leq \alpha_k \leq$

We must do  $R^2/2$  dot products to get this matrix ready.  
 Each dot product now only requires  $m$  additions and multiplications

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(x_k)$$

$$b = y_K (1 - \epsilon_K) - \mathbf{x}_K \cdot \mathbf{w}_K$$

where  $K = \arg \max_k \alpha_k$

Then classify with:  
 $f(x, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \Phi(x) - b)$

Copyright © 2001, 2003, Andrew W. Moore

### Higher Order Polynomials

Polynomial	$\Phi(x)$	Cost to build $Q_{ij}$ matrix traditionally	Cost if 100 inputs	$\Phi(a) \cdot \Phi(b)$	Cost to build $Q_{ij}$ matrix efficiently	Cost if 100 inputs
Quadratic	All $m^2/2$ terms up to degree 2	$m^2 R^2 / 4$	$2,500 R^2$	$(a \cdot b + 1)^2$	$m R^2 / 2$	$50 R^2$
Cubic	All $m^3/6$ terms up to degree 3	$m^3 R^2 / 12$	$83,000 R^2$	$(a \cdot b + 1)^3$	$m R^2 / 2$	$50 R^2$
Quartic	All $m^4/24$ terms up to degree 4	$m^4 R^2 / 48$	$1,960,000 R^2$	$(a \cdot b + 1)^4$	$m R^2 / 2$	$50 R^2$

Copyright © 2001, 2003, Andrew W. Moore

We must do  $R^2/2$  dot products to get this matrix ready.  
 In 100-d, each dot product now needs 103 operations instead of 75 million

But there are still worrying things lurking away. What are they?  
 constraints.

$$\forall k \quad \sum_{k=1}^R \alpha_k y_k = 0$$

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(x_k)$$

$$b = y_K (1 - \epsilon_K) - \mathbf{x}_K \cdot \mathbf{w}_K$$

where  $K = \arg \max_k \alpha_k$

Then classify with:  
 $f(x, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \Phi(x) - b)$

Copyright © 2001, 2003, Andrew W. Moore

We must do  $R^2/2$  dot products to get this matrix ready.  
 In 100-d, each dot product now needs 103 operations instead of 75 million

But there are still worrying things lurking away. What are they?  
 constraints.

$$\forall k \quad \sum_{k=1}^R \alpha_k y_k = 0$$

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(x_k)$$

$$b = y_K (1 - \epsilon_K) - \mathbf{x}_K \cdot \mathbf{w}_K$$

where  $K = \arg \max_k \alpha_k$

Then classify with:  
 $f(x, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \Phi(x) - b)$

- The fear of overfitting with this enormous number of terms
- The evaluation phase (doing a set of predictions on a test set) will be very expensive (why?)

Copyright © 2001, 2003, Andrew W. Moore

We must do  $R^2/2$  dot products to get this matrix ready.

In 100-d, each dot product now needs 103 operations instead of 75 million

But there are still worrying things lurking away. What are they?

constraints.

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(\mathbf{x}_k)$$

$$b = y_K (1 - \epsilon_K) - \mathbf{x}_K \cdot \mathbf{w}_K$$

where  $K = \arg \max_k \alpha_k$

Then classify with:

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \Phi(\mathbf{x}) - b)$$

**Kernel functions**  
 $Q_{kl} = y_l y_k (\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_l))$

The use of Maximum Margin **magically** makes this not a problem

$\forall k \sum_l \alpha_l y_l = 0$

•The fear of overfitting with this enormous number of terms

•The evaluation phase (doing a set of predictions on a test set) will be very expensive (why?)

Because each  $\mathbf{w} \cdot \Phi(\mathbf{x})$  (see below) needs 75 million operations. What can be done?

Copyright © 2001, 2003, Andrew W. Moore

We must do  $R^2/2$  dot products to get this matrix ready.

In 100-d, each dot product now needs 103 operations instead of 75 million

But there are still worrying things lurking away. What are they?

constraints.

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(\mathbf{x}_k)$$

$$\mathbf{w} \cdot \Phi(\mathbf{x}) = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}) = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k (\mathbf{x}_k \cdot \mathbf{x} + 1)^5$$

Only  $S$ m operations ( $S$ =#support vectors)

Then classify with:

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \Phi(\mathbf{x}) - b)$$

**Kernel functions**  
 $Q_{kl} = y_l y_k (\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_l))$

The use of Maximum Margin **magically** makes this not a problem

$\forall k \sum_l \alpha_l y_l = 0$

•The fear of overfitting with this enormous number of terms

•The evaluation phase (doing a set of predictions on a test set) will be very expensive (why?)

Because each  $\mathbf{w} \cdot \Phi(\mathbf{x})$  (see below) needs 75 million operations. What can be done?

Copyright © 2001, 2003, Andrew W. Moore

We must do  $R^2/2$  dot products to get this matrix ready.

In 100-d, each dot product now needs 103 operations instead of 75 million

But there are still worrying things lurking away. What are they?

constraints.

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(\mathbf{x}_k)$$

$$\mathbf{w} \cdot \Phi(\mathbf{x}) = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}) = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k (\mathbf{x}_k \cdot \mathbf{x} + 1)^5$$

Only  $S$ m operations ( $S$ =#support vectors)

Then classify with:

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \Phi(\mathbf{x}) - b)$$

**Kernel functions**  
 $Q_{kl} = y_l y_k (\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_l))$

The use of Maximum Margin **magically** makes this not a problem

$\forall k \sum_l \alpha_l y_l = 0$

•The fear of overfitting with this enormous number of terms

•The evaluation phase (doing a set of predictions on a test set) will be very expensive (why?)

Because each  $\mathbf{w} \cdot \Phi(\mathbf{x})$  (see below) needs 75 million operations. What can be done?

Copyright © 2001, 2003, Andrew W. Moore

● ● ● | **QP with Quintic basis functions**

Maximize  $\sum_{k=1}^R \alpha_k - \frac{1}{2} \sum_{k=1}^R \sum_{l=1}^R \alpha_k \alpha_l Q_{kl}$  w.r.t.  $\alpha_k$

Subject to these constraints:  $0 \leq \alpha_k \leq C$

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(\mathbf{x}_k)$$

$$\mathbf{w} \cdot \Phi(\mathbf{x}) = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}) = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k (\mathbf{x}_k \cdot \mathbf{x} + 1)^5$$

Only  $S$ m operations ( $S$ =#support vectors)

Then classify with:

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \Phi(\mathbf{x}) - b)$$

Why SVMs don't overfit as much as you'd think:

No matter what the basis function, there are really only up to  $R$  parameters:  $\alpha_1, \alpha_2, \dots, \alpha_R$ , and usually most are set to zero by the Maximum Margin.

Asking for small  $\mathbf{w} \cdot \mathbf{w}$  is like "weight decay" in Neural Nets and like Ridge Regression parameters in Linear regression and like the use of Priors in Bayesian Regression---all designed to smooth the function and reduce overfitting.

Copyright © 2001, 2003, Andrew W. Moore

● ● ● | **SVM Kernel Functions**

- $K(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + 1)^d$  is an example of an SVM Kernel Function
- Beyond polynomials there are other very high dimensional basis functions that can be made practical by finding the right Kernel Function
  - Radial-Basis-style Kernel Function:
 
$$K(\mathbf{a}, \mathbf{b}) = \exp\left(-\frac{(\mathbf{a} - \mathbf{b})^2}{2\sigma^2}\right)$$
  - Neural-net-style Kernel Function:
 
$$K(\mathbf{a}, \mathbf{b}) = \tanh(\kappa \mathbf{a} \cdot \mathbf{b} - \delta)$$

$\sigma, \kappa$  and  $\delta$  are magic parameters that must be chosen by a model selection method such as CV or VCSRM

Copyright © 2001, 2003, Andrew W. Moore

● ● ● | **What Makes a Kernel**

Mercer's theorem characterizes when a function  $f(\vec{x}, \vec{z})$  is a kernel

If  $K_1()$  and  $K_2()$  are kernels, then so are

- 1)  $K_1() + K_2()$
- 2)  $c * K_1()$  where  $c$  is constant
- 3)  $K_1() * K_2()$
- 4)  $f(\vec{x}) * f(\vec{z})$  where  $f()$  returns a real

...

## ● ● ● | Key SVM Ideas

- Maximize the **margin** between positive and negative examples (connects to PAC theory)
- Penalize errors in non-separable case
- Only the **support vectors** contribute to the solution
- Kernels map examples into a new, usually non-linear space
  - We implicitly do dot products in this new space (in the “dual” form of the SVM program)

## ● ● ● | SVM Performance

- Anecdotally they work very very well indeed.
- Example: They are currently the best-known classifier on a well-studied hand-written-character recognition benchmark
- Another Example: AWM knows several reliable people doing practical real-world work who claim that SVMs have saved them when their other favorite classifiers did poorly.
- There is a lot of excitement and religious fervor about SVMs and Kernel machines as of 2004.
- Despite this, some practitioners are a little skeptical.

Copyright © 2001, 2003, Andrew W. Moore

## ● ● ● | Doing multi-class classification

- SVMs can only handle two-class outputs (i.e. a categorical output variable with arity 2).
- What can be done?
- Answer: with output arity N, learn N SVM's
  - SVM 1 learns “Output==1” vs “Output != 1”
  - SVM 2 learns “Output==2” vs “Output != 2”
  - :
  - SVM N learns “Output==N” vs “Output != N”
- Then to predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

Copyright © 2001, 2003, Andrew W. Moore

## ● ● ● | SVM Implementations

- Sequential Minimal Optimization, SMO, efficient implementation of SVMs, Platt
  - in Weka
- SVM<sup>light</sup>
  - <http://svmlight.joachims.org/>
- LibSVM
  - <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

## ● ● ● | References

- Tutorial on VC-dimension and Support Vector Machines:
  - C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):955-974, 1998. <http://citeseer.nj.nec.com/burges98tutorial.html>
- The VC/SRM/SVM Bible:
  - Statistical Learning Theory by Vladimir Vapnik, Wiley-Interscience; 1998

Copyright © 2001, 2003, Andrew W. Moore