

# Chapter 7

## Sparse Kernel Machines

Vladimir Nedović

Intelligent Systems Lab Amsterdam (ISLA)  
University of Amsterdam



20-04-2007

# Introduction

- Kernel function  $k(x_n, x_m)$  must be evaluated for all pairs of training points
- Advantageous (computationally) to look at kernel algorithms that have *sparse* solutions
  - new predictions depend only on kernel function evaluated at a few training points
  - possible solution: *support vector machines (SVM)*
    - determination of parameters is a convex optimization problem => any solution = global optimum
    - but, SVM does not provide posterior probabilities

# Maximum Margin Classifiers (1)

- Setup:
  - $y(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) + b$ ,  $\Phi(\mathbf{x}) =$  feature-space transformation
  - $N$  input data vectors  $x_1, \dots, x_N$
  - $N$  target vectors  $t_1, \dots, t_N$   $t_n \in \{-1, 1\}$
  - new samples classified according to sign of  $y(\mathbf{x})$
  
  - assume linearly separable data for the moment
    - $y(\mathbf{x}) > 0$  for points of class +1 (i.e.  $t_n = +1$ )
    - $y(\mathbf{x}) < 0$  for points having  $t_n = -1$
    - therefore,  $t_n y(\mathbf{x}_n) > 0$  for all *training* points

# Maximum Margin Classifiers (2)

- There may be many solutions to separate the data

SVM tries to minimize the generalization error by introducing the concept of a *margin*

i.e. smallest distance between the decision boundary and *any* of the samples

- Decision boundary is chosen to be that for which the margin is maximized
  - - decision boundary thus only dependent on points closest to it
    - similar concept to maximizing the between-class variance?

# Maximum Margin Classifiers (3)

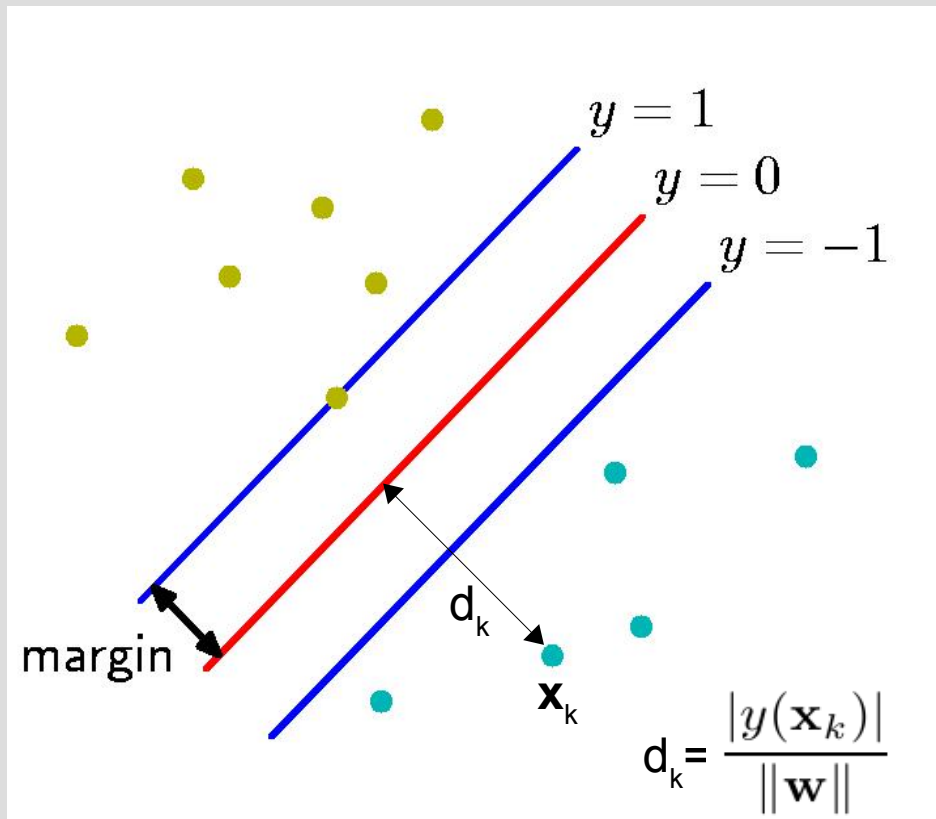


Figure 7.1a

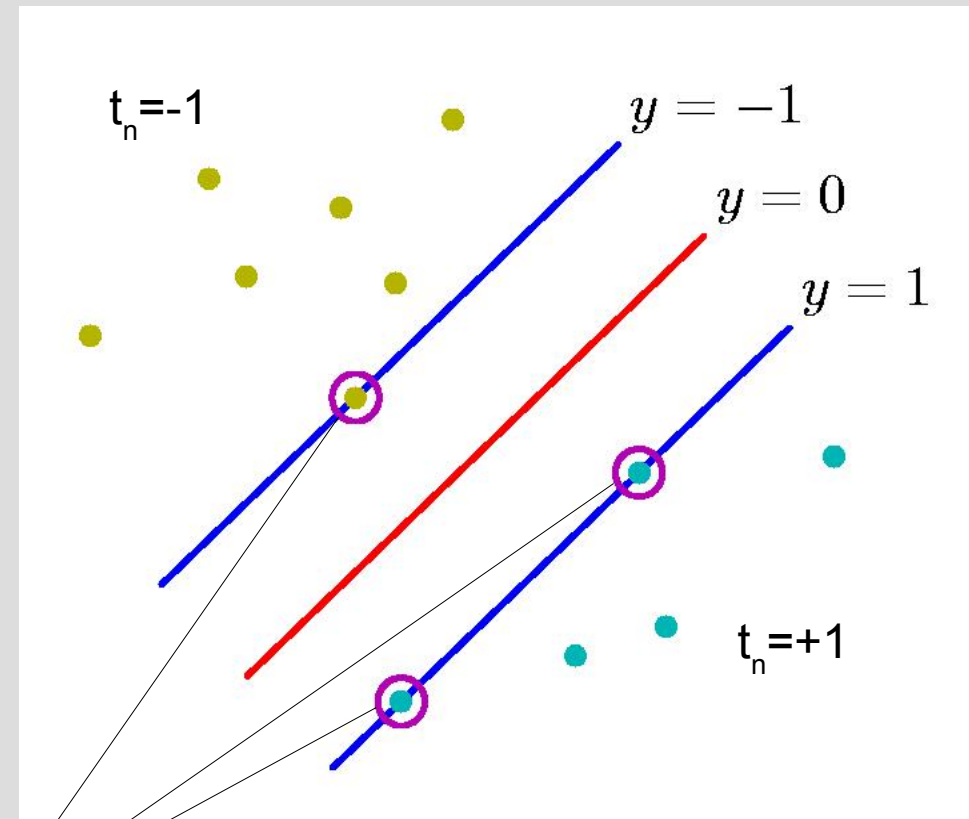


Figure 7.1b

support vectors

# Maximum Margin Classifiers (4)

- If we consider only points that are correctly classified, then we have  $t_n y(\mathbf{x}_n) > 0$  for all  $n$
- The distance of  $\mathbf{x}_n$  to the decision surface is then

$$\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|} \quad (7.2)$$

- Optimize  $\mathbf{w}$  and  $b$  to maximize this distance  $\Rightarrow$  solve

$$\arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n [t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)] \right\} \quad (7.3)$$

- Quite hard to solve  $\Rightarrow$  need to get around it

# Maximum Margin Classifiers (5)

- Rescaling  $\mathbf{w}$  and  $b$  with the same factor in  $\frac{t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}$  does not change the distance
- So for points closest to the decision surface (i.e. support vectors), we can set

$$t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) = 1 \quad (7.4)$$

- Such that for all points we have

$$t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \dots, N \quad (7.5)$$

- That is the canonical representation of the decision hyperplane
  - for points with  $=$ , constraint is active
  - for points with  $>$ , constraint is inactive

# Maximum Margin Classifiers (6)

- Optimization of  $\arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n [t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)] \right\}$  then reduces to maximizing  $\|\mathbf{w}\|^{-1}$ , equiv. to minimizing  $\|\mathbf{w}\|^2$ :

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad (7.6)$$

- (square and  $\frac{1}{2}$  included for later convenience)

- Minimize a quadratic function, subject to a set of linear inequality constraints => Lagrange:

$$L(\mathbf{w}, b, \mathbf{a}) = \underbrace{\frac{1}{2} \|\mathbf{w}\|^2}_{(7.6)} - \sum_{n=1}^N a_n \underbrace{\left\{ t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) - 1 \right\}}_{\text{constraint 7.5}} \quad (7.7)$$

one Lagrange multiplier for each  $t_n$

# Maximum Margin Classifiers (7)

- Eliminating  $\mathbf{w}$  and  $b$  from the equation (through derivatives), leads to the *dual representation* of the max. margin problem (in terms of *kernel function*, without explicitly formulating  $\Phi(\mathbf{x})$ ):

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \quad (7.10)$$

– the constraints are  $a_n \geq 0$  and  $\sum_{n=1}^N a_n t_n = 0$ , kernel is  $k(x, x') = \phi(x)^T \phi(x')$

- In going from (7.7) to (7.10), optimization over  $M$  variables (bases) was replaced by that over  $N$  variables (not sure how)
  - since usually  $M < N$ , this seems to be a disadvantage
- But problem reformulated in terms of kernel function  $\Rightarrow$  max. margin classifier can be applied to feature spaces whose dimensionality exceeds  $N$

# Maximum Margin Classifiers (8)

- Equation (7.10) also explains why the kernel function should be positive definite
  - Lagrangian function (7.10) then has a lower bound
- Test samples are classified by evaluating the sign of  $y(x) = \mathbf{w}^T \phi(x) + b$ 
  - in terms of parameters  $a_n$  and the kernel function, we have

$$(7.8) \quad \mathbf{w} = \sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n) \Rightarrow y(\mathbf{x}) = \sum_{n=1}^N a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b \quad (7.13)$$

- in that case, following 3 conditions hold:  $a_n \geq 0$  (7.14)

$$t_n y(\mathbf{x}_n) - 1 \geq 0 \quad (7.15)$$

$$a_n \{t_n y(\mathbf{x}_n) - 1\} = 0 \quad (7.16)$$

- thus, for every point, either  $a_n = 0$  (those do not make a difference in the sum) or  $t_n y(\mathbf{x}_n) = 1$  (those are support vectors)

# Maximum Margin Classifiers (9)

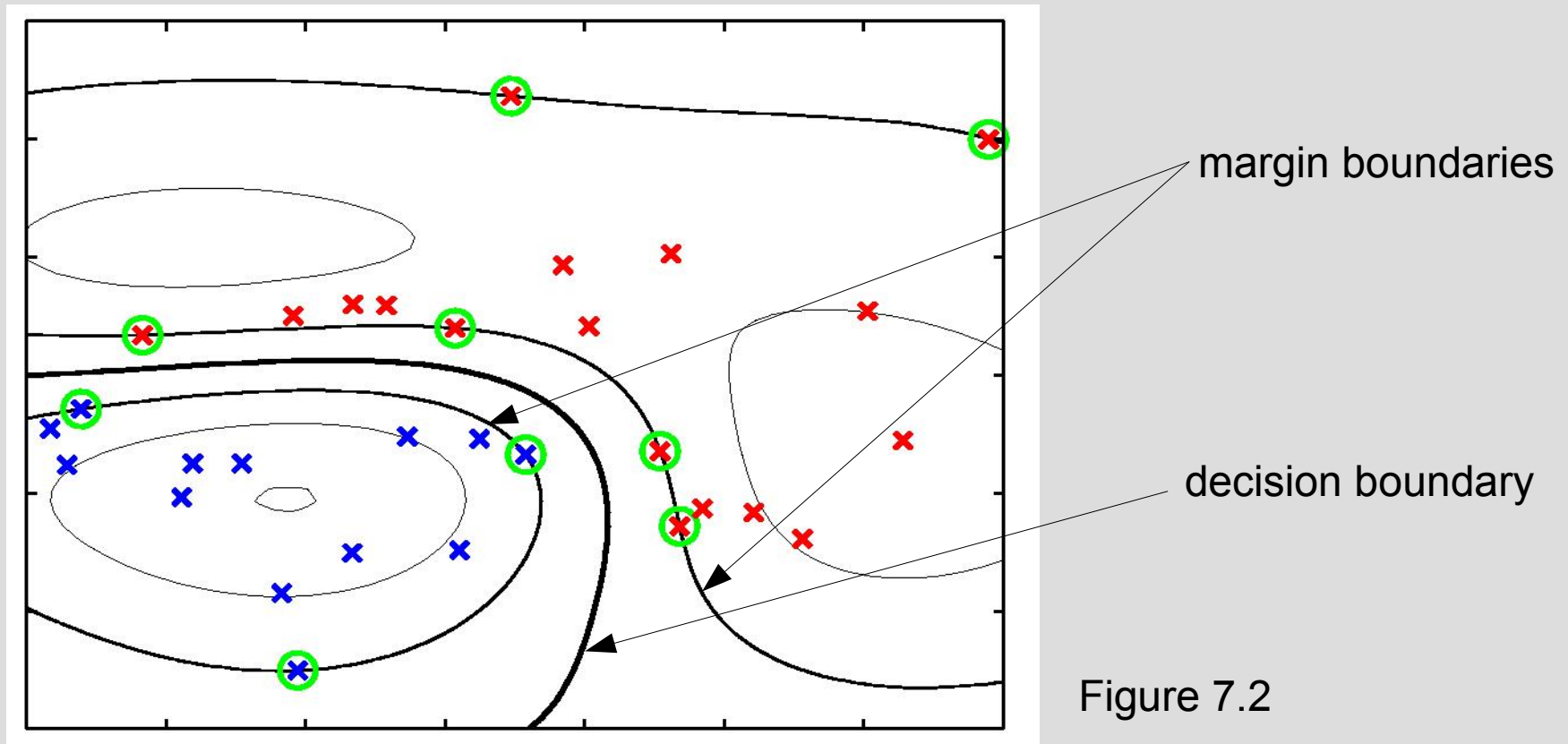
- Since only the support vectors (i.e. points that lie on the margin boundary) matter, a big proportion of the data can be discarded after the training phase => very practical
- The threshold  $b$  is determined using only those points:

$$t_n \left( \sum_{m \in S} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b \right) = 1 \quad (7.17)$$

set of  
support  
vectors

$$b = \frac{1}{N_S} \sum_{n \in S} \left( t_n - \sum_{m \in S} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right) \quad (7.18)$$

# Maximum Margin Classifiers (10)



- Although data not linearly separable in the original data space  $\mathbf{x}$ , it is in the non-linear feature space (defined implicitly by the kernel!)  
=> the training data points are separable in the original data space

# Maximum Margin Classifiers (11)

## 7.1.1 Overlapping class distributions

- So far, we have assumed linearly separable classes (in space  $\Phi(x)$ )
- The error function (7.19) punishes misclassifications infinitely
  - need to soften this by allowing mistakes
- Introduce slack variables as penalty that linearly increases with distance of a point from its correct margin boundary

- $\xi_n = 0$  for points on or inside the correct margin boundary
- $\xi_n = |t_n - y(\mathbf{x}_n)|$  for other points
- (7.5) becomes

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n, \quad n = 1, \dots, N \quad (7.20)$$

(7.5), sufficient for correct classifications

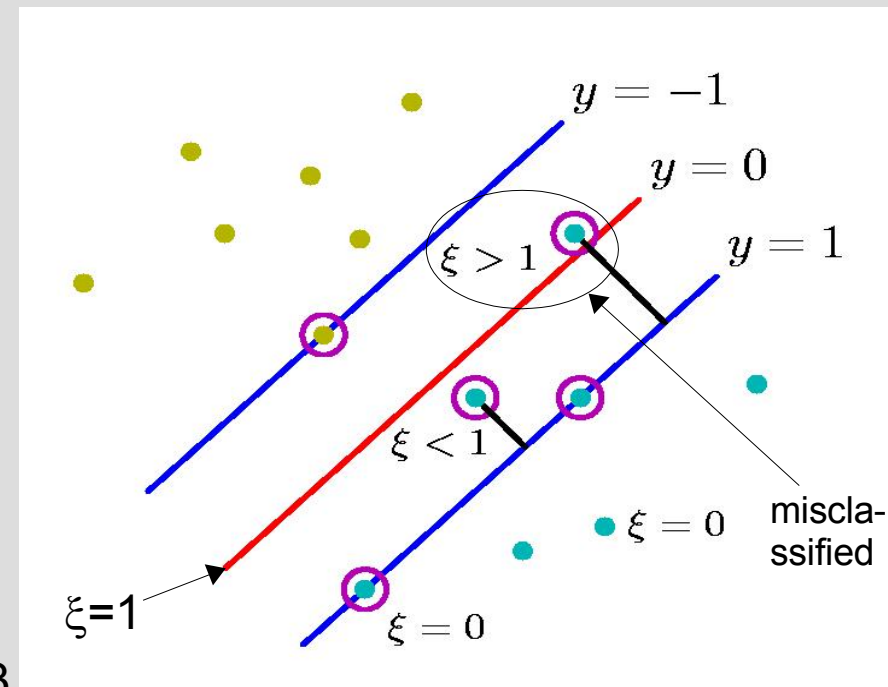


Figure 7.3

# Maximum Margin Classifiers (12)

## 7.1.1 Overlapping class distributions

- So now we need to optimize for  $\|\mathbf{w}\|$  and for  $\xi$
- The Lagrangian is

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n - \sum_{n=1}^N a_n \{t_n y(\mathbf{x}_n) - 1 + \xi_n\} - \sum_{n=1}^N \mu_n \xi_n \quad (7.22)$$

- And the constraints are  $a_n \geq 0$ ,  $\mu_n \geq 0$ ,  $\xi_n \geq 0$ ,  $\mu_n \xi_n = 0$

$$t_n y(\mathbf{x}_n) - 1 + \xi_n \geq 0 \quad (7.24)$$

$$a_n (t_n y(\mathbf{x}_n) - 1 + \xi_n) = 0 \quad (7.25)$$

- Using derivatives and optimizing leads to the dual representation:

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \quad (7.32)$$

- This is identical to the formula for the separable case, with slightly different constraints:  $0 \leq a_n \leq C$  and  $\sum_{n=1}^N a_n t_n = 0$

box constraint

# Maximum Margin Classifiers (13)

## 7.1.1 Overlapping class distributions

- Even though the constraints are different, they lead to the same optimization problem, and the classifications are made using (7.13):

$$y(\mathbf{x}) = \sum_{n=1}^N a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

- From (7.24) and (7.25) we see that a subset of data may have  $a_n = 0$  and they do not contribute to the model; the remaining points are the support vectors – for those,  $a_n > 0$  and  $t_n y(\mathbf{x}_n) = 1 - \xi_n$

Note that now, support vectors contain more than just points on the margin:

- (1)  $a_n < C \Rightarrow \xi_n = 0$  and  $t_n y(\mathbf{x}_n) = 1$ ; these lie on the margin
- (2)  $a_n = C$  points lie inside the margin;  $\xi_n \leq 1$  are correctly classified, whereas  $\xi_n > 1$  are misclassified

# Maximum Margin Classifiers (14)

## 7.1.1 Overlapping class distributions

- The threshold  $b$  is now determined via the support vectors

$$b = \frac{1}{N_{\mathcal{M}}} \sum_{n \in \mathcal{M}} \left( t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right) \quad (7.37)$$

subset of support vectors that lie on the margin

set of all support vectors

- Bishop says (bottom, p.334) that both parameters  $\mathbf{a}$  and  $b$  are determined using the whole data set, not only the support vectors
  - I don't see that in case of  $b$  from the formulas
- Many algorithms have been proposed to solve the quadratic optimization problem: chunking, decomposition methods, sequential minimal optimization (SMO – extreme chunking)

# Maximum Margin Classifiers (15)

## 7.1.1 Overlapping class distributions

- Although expressed in terms of kernel function, SVMs do not avoid the curse of dimensionality – it is still defined by that of the feature mapping  $\Phi(\mathbf{x})$
- SVMs only provide classification results, do not output posterior probabilities
  - Platt proposed fitting a logistic sigmoid to the outputs of the SVM; the conditional *pdf* is of the form

$$p(t = 1|\mathbf{x}) = \sigma(Ay(\mathbf{x} + B)) \quad (7.43)$$

- values for A and B are found by minimizing the cross-entropy error function defined by a training set, consisting of pairs of values  $y(\mathbf{x}_n)$  and  $t_n$
- this is equivalent to assuming that the output of SVM represents the log-odds of  $\mathbf{x}$  belonging to class  $t=1$ ; SVMs not designed for this => they can give poor approximation to posteriors

# Maximum Margin Classifiers (16)

## 7.1.2 Relation to logistic regression

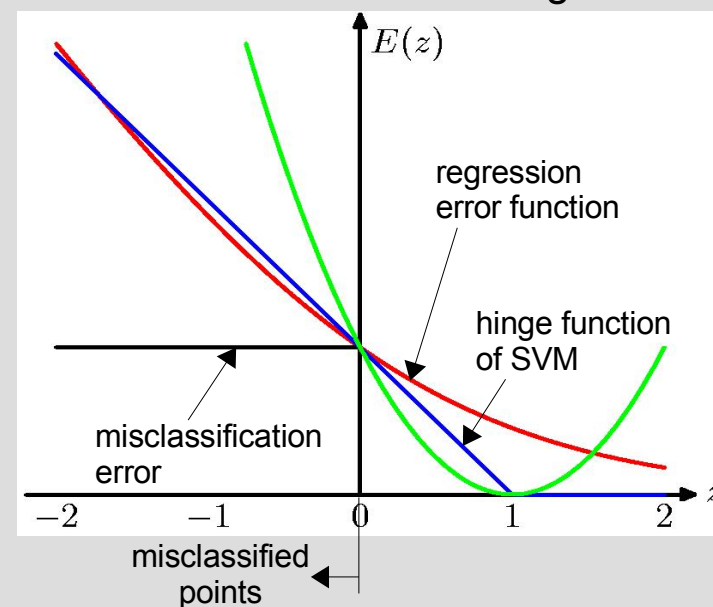
- Rewind a bit: for data points on the correct side of the margin (i.e.  $y_n t_n \geq 1$ ) we have  $\xi_n = 0$ , whereas for the remaining ones we have  $\xi_n = 1 - y_n t_n$ 
  - $0 \leq y_n t_n \leq 1$ ;  $\xi_n \leq 1$ ; correctly classified points
  - $y_n t_n < 0$ ;  $\xi_n > 1$ ; misclassified points
- Now the optimization function (7.21) can be written as

$$\sum_{n=1}^N E_{SV}(y_n t_n) + \lambda \|\mathbf{w}\|^2 \quad (7.44)$$

- $\lambda$  is related to  $C$ ,  $E_{SV}$  is the hinge function that is an approx. of the misclass. error

$$E_{SV}(y_n t_n) = [1 - y_n t_n]_+ \quad (7.45)$$

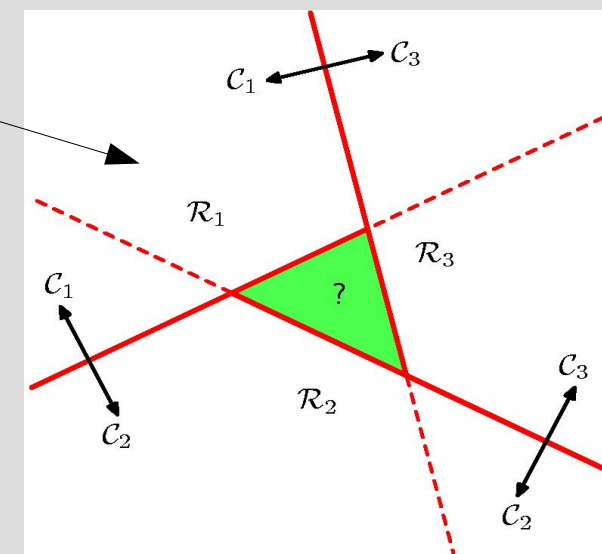
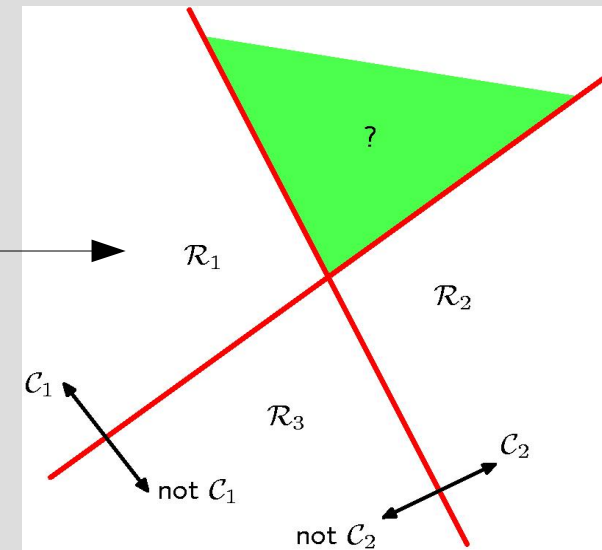
Figure 7.5



# Maximum Margin Classifiers (17)

## 7.1.3 Multi-class SVM

- SVM is fundamentally a two-class classifier
- In order to use it for more than 2 classes:
  - *one-vs-all* approach ( $K$  separate SVMs); however
    - resolved using  $y(\mathbf{x}) = \max_k y_k(\mathbf{x})$ , although this heuristic approach suffers from some drawbacks
  - another approach: *one-vs-one* ( $K(K-1)/2$  different SVMs + voting); however



# Maximum Margin Classifiers (18)

## 7.1.4 SVMs for regression

- Extend SVM to regression problems, but preserve sparseness
- To account for sparseness, the MSE function from regression is replaced by an  $\epsilon$ -sensitive error function
  - an example of such an error is

$$E_\epsilon(y(\mathbf{x}) - t) = \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \epsilon \\ |y(\mathbf{x}) - t| - \epsilon, & \text{otherwise} \end{cases}$$

0 for correct classifications

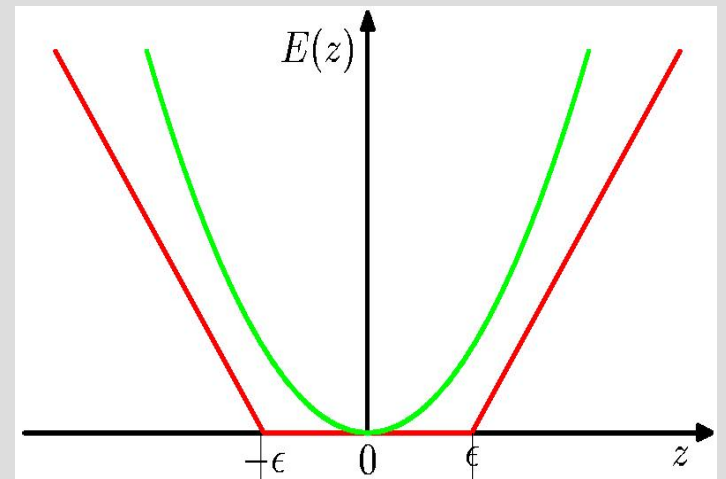


Figure 7.6

- Need to optimize

$$C \sum_{n=1}^N E_\epsilon(y(\mathbf{x}_n) - t_n) + \frac{1}{2} \|\mathbf{w}\|^2 \quad (7.52)$$

- Introduce slack variables  $\xi_n \geq 0$  and  $\hat{\xi}_n \geq 0$ , where strictly greater corresponds to

$$\xi_n > 0 : t_n > y(\mathbf{x}_n) + \epsilon$$

$$\hat{\xi}_n > 0 : t_n < y(\mathbf{x}_n) - \epsilon$$

# Maximum Margin Classifiers (19)

## 7.1.4 SVMs for regression

- The condition for a target point to lie inside the insensitive region (the  $\epsilon$ -tube) is  $y_n - \epsilon \leq t_n \leq y_n + \epsilon$
- Introducing slack variables allows points to lie outside the tube
  - conditions are

$$t_n \leq y(\mathbf{x}_n) + \epsilon + \xi_n \quad (7.53)$$

$$t_n \geq y(\mathbf{x}_n) - \epsilon - \hat{\xi}_n \quad (7.54)$$

- The error function for SV regression is then

$$C \sum_{n=1}^N (\xi_n + \hat{\xi}_n) + \frac{1}{2} \|\mathbf{w}\|^2 \quad (7.55)$$

- To be minimized with Lagrangian:

$$(7.56) \quad L = C \sum_{n=1}^N (\xi_n + \hat{\xi}_n) + \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N (\mu_n \xi_n + \hat{\mu}_n \hat{\xi}_n) - \sum_{n=1}^N a_n (\epsilon + \xi_n + y_n - t_n) - \sum_{n=1}^N \hat{a}_n (\epsilon + \hat{\xi}_n + y_n - t_n)$$

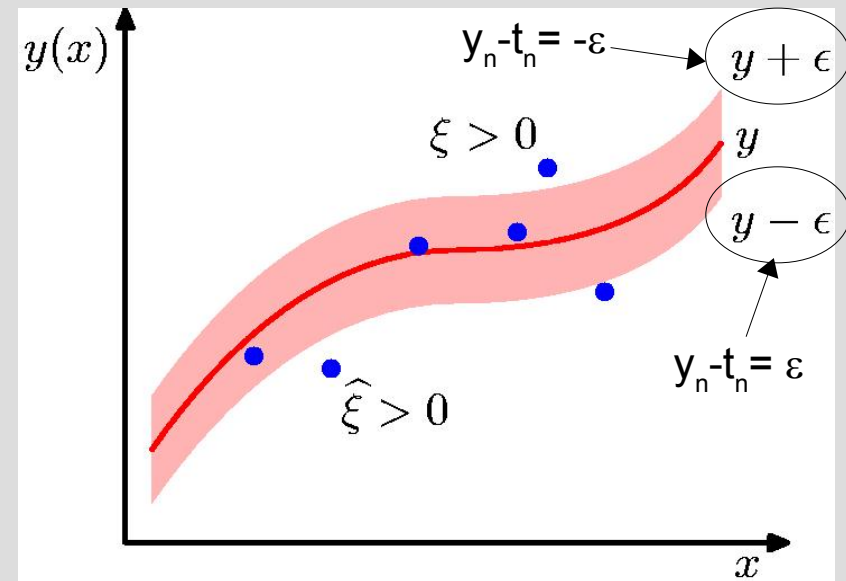


Figure 7.7

# Maximum Margin Classifiers (20)

## 7.1.4 SVMs for regression

- Through derivatives, we get the expression for  $\mathbf{w}$  to be

$$\mathbf{w} = \sum_{n=1}^N (a_n - \hat{a}_n) \phi(\mathbf{x}_n) \quad (7.57)$$

- Therefore, the predictions for new inputs can be made using

$$y(\mathbf{x}) = \sum_{n=1}^N (a_n - \hat{a}_n) k(\mathbf{x}, \mathbf{x}_n) + b \quad (7.64)$$

- the conditions are  $a_n(\epsilon + \xi_n + y_n - t_n) = 0$  (7.65)

$$\hat{a}_n(\epsilon + \hat{\xi}_n + y_n - t_n) = 0 \quad (7.66)$$

$$(C - a_n)\xi_n = 0 \quad (7.67)$$

$$(C - \hat{a}_n)\hat{\xi}_n = 0 \quad (7.68)$$

- so for both (7.65) and (7.66), either  $a_n = 0$  (no contribution from such points) or the term  $(\epsilon + \xi_n + y_n - t_n)$  is 0

# Maximum Margin Classifiers (21)

## 7.1.4 SVMs for regression

- The term  $(\epsilon + \xi_n + y_n - t_n)$  can only be zero in two cases:
  - (1) for points that lie on the upper boundary of the tube,  $\xi_n = 0$  and  $y_n - t_n = -\epsilon$
  - (2) for points that lie above the upper boundary of the tube,  $\xi_n > 0$  and  $y_n - t_n = -\xi_n - \epsilon$
- Similar is true for points on or below the lower boundary
- Therefore, support vectors in this case are those points that either lie on the boundary of the tube or outside of it
  - i.e. points far away from the desired curve define it the best
  - again a sparse solution
- Parameter  $b$  is then found from

$$b = t_n - \epsilon - \sum_{m=1}^N (a_m - \hat{a}_m) k(\mathbf{x}_n, \mathbf{x}_m) \quad (7.69)$$

# Maximum Margin Classifiers (22)

## 7.1.4 SVMs for regression & 7.1.5 Computational learning theory

- SVM for regression: support vectors are points on or outside the  $\varepsilon$ -tube
- computational/statistical learning theory: based on the *probably approximately correct (PAC)* framework
  - PAC tries to understand how large a data set needs to be in order to give good generalization
  - bounds very strict, since they do not consider stat. regularities of data (i.e. that many points belong to the same class)
  - therefore, not many practical applications of PAC

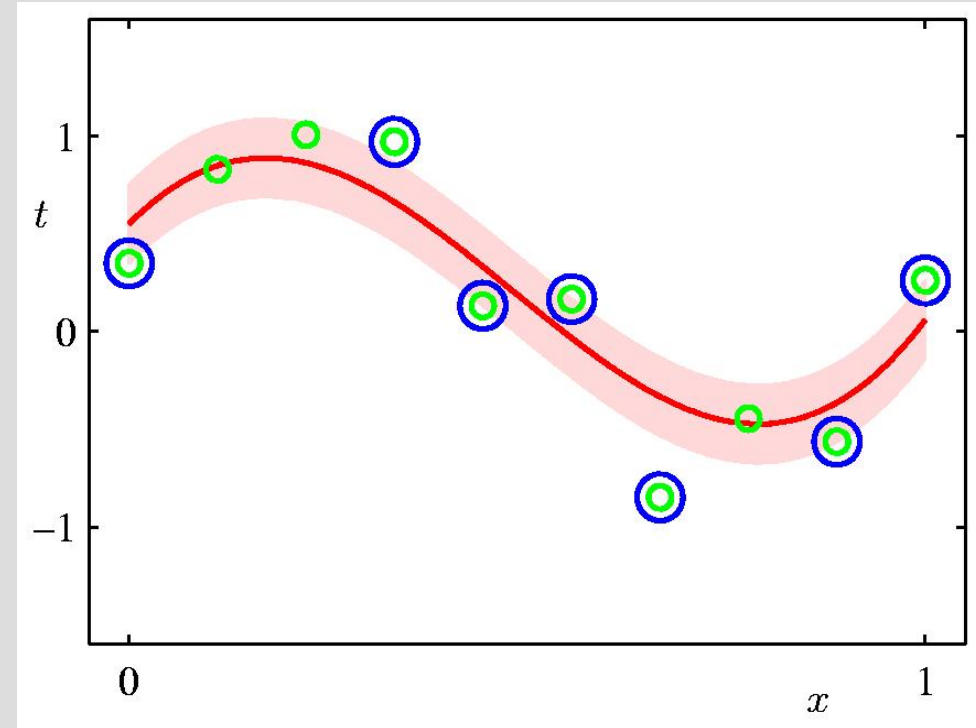


Figure 7.8

# Relevance Vector Machines (1)

- Why SVMs are bad:
  - outputs are not probabilities
  - extension to  $K > 2$  classes problematic
  - there is a complexity parameter that must be found using cross-validation
  - kernel functions are centered on data points (don't know why that is bad) and must be positive definite
- Why RVMs are good:
  - it is a *Bayesian* sparse kernel method
  - shares advantages of SVM while avoiding its limitations (but it suffices to say 'Bayes')
  - it results in sparser models and faster performance on test data (generalization not hurt)
- For some reason, here we do regression first, then classification

# Relevance Vector Machines (2)

## 7.2.1 RVM for regression

- Linear model of the same form as those in Ch.3, but with a modified prior
  - the new prior results in sparse solutions
  - the model generally defines a cond. *pdf* for target  $t$ , given  $\mathbf{x}$ :

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}), \beta^{-1}) \quad (7.76), \text{ same as (3.08)}$$

- variance is given only by noise through  $\beta$ , whereas the mean is

$$y(\mathbf{x}) = \sum_{i=1}^M w_i \phi_i(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) \quad (7.77)$$

- RVM mimics the structure of the SVM
  - basis functions given by kernels, one per training point
  - the general expression (7.77) is then

$$y(\mathbf{x}) = \sum_{n=1}^N w_n k(\mathbf{x}, \mathbf{x}_n) + b \quad (7.78), \text{ same form as (7.64) of SVMs}$$

# Relevance Vector Machines (3)

## 7.2.1 RVM for regression

- However, we turn back to (7.77), since the basis functions considered here need not necessarily be positive-definite kernels
- If we have  $N$  observations of  $\mathbf{x}$ , and put them as rows in matrix  $\mathbf{X}$ , then the likelihood function for target values  $\mathbf{t}$  (of all  $N$   $\mathbf{x}$ 's) is

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^N p(t_n|\mathbf{x}_n, \mathbf{w}, \beta^{-1}) \quad (7.79)$$

- We also define a prior distribution over the parameter  $\mathbf{w}$ 
  - instead of shared inverse variance, now one hyper-parameter per weight  $w_i$

$$p(\mathbf{w}|\alpha) = \prod_{i=1}^M \mathcal{N}(w_i|0, \alpha_i^{-1}) \quad (7.80)$$

- (many  $\alpha_i$  will go to Inf., such that their corresponding  $w_i$  have post. *pdfs* concentrated at 0 => their basis functions do not contribute to the model at all)

# Relevance Vector Machines (4)

## 7.2.1 RVM for regression

- The posterior for the weights is also a Gaussian (Ch.3), of the form

$$p(\mathbf{w}|\mathbf{t}, \mathbf{X}, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\mathbf{m}, \Sigma) \quad (7.81)$$

- where mean and covariance are

$$\mathbf{m} = \beta \Sigma \Phi^T \mathbf{t} \quad \Sigma = (\mathbf{A} + \beta \Phi^T \Phi)^{-1} \quad (7.82), (7.83)$$

- $\Phi$  is the  $N \times M$  design matrix, and  $\mathbf{A}$  is a diagonal matrix with  $\alpha_i$  as elements
- $\alpha$  and  $\beta$  are determined using type-2 MLE (evidence approximation or *automatic relevance determination, ARD*) - maximize the marginal likelihood, obtained by summing the weight parameters:

$$p(\mathbf{t}|\mathbf{X}, \alpha, \beta) = \int p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) p(\mathbf{w}|\alpha) d\mathbf{w} \quad (7.84)$$

- choose initial values for  $\alpha$  and  $\beta$ , then calculate  $\mathbf{m}$  and  $\Sigma$ ; then re-estimate  $\alpha$  and  $\beta$ , re-calculate  $\mathbf{m}$  and  $\Sigma$  etc. until convergence.

# Relevance Vector Machines (5)

## 7.2.1 RVM for regression

- During the ARD optimization procedure, many  $\alpha_i$  become very large (i.e. infinite), such that their weights have *pdfs* with both mean and variance 0
  - => associated bases removed from the model, do not contribute in making new predictions
  - points with non-zero weights are called *relevance vectors*
- Now we can evaluate the predictive *pdf* over  $t$  for new points:

$$p(\mathbf{t}|\mathbf{x}, \mathbf{X}, \mathbf{t}, \alpha^*, \beta^*) = \int p(t|\mathbf{x}, \mathbf{w}, \beta^*)p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \alpha^*, \beta^*)d\mathbf{w}$$
$$= \mathcal{N}(t|\underbrace{\mathbf{m}^T \phi(\mathbf{x})}_{\text{same as (7.77), but with a posterior mean } \mathbf{m}}, \sigma^2(\mathbf{x}))$$

(7.90), same form as (7.76)

- variance is given by

$$\sigma^2(\mathbf{x}) = (\beta^*)^{-1} + \phi(\mathbf{x})^T \Sigma \phi(\mathbf{x}) \quad (7.91)$$

# Relevance Vector Machines (6)

## 7.2.1 RVM for regression

- Figure 7.9 shows the same data as in Figure 7.8 and the same kernel functions
  - number of RVs much smaller than that of SVs
  - models typically an order of magnitude more compact than corresponding SVM

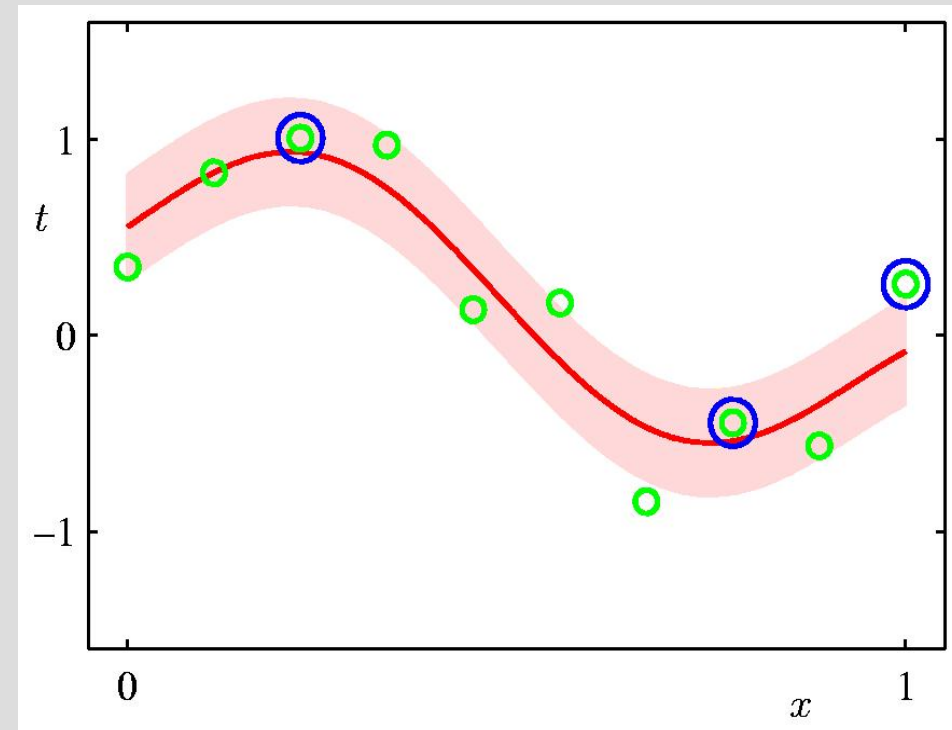


Figure 7.9

- the main disadvantage of RVMs is that non-convex function must be optimized during training and that training times can be longer

# Relevance Vector Machines (7)

## 7.2.2 Analysis of sparsity

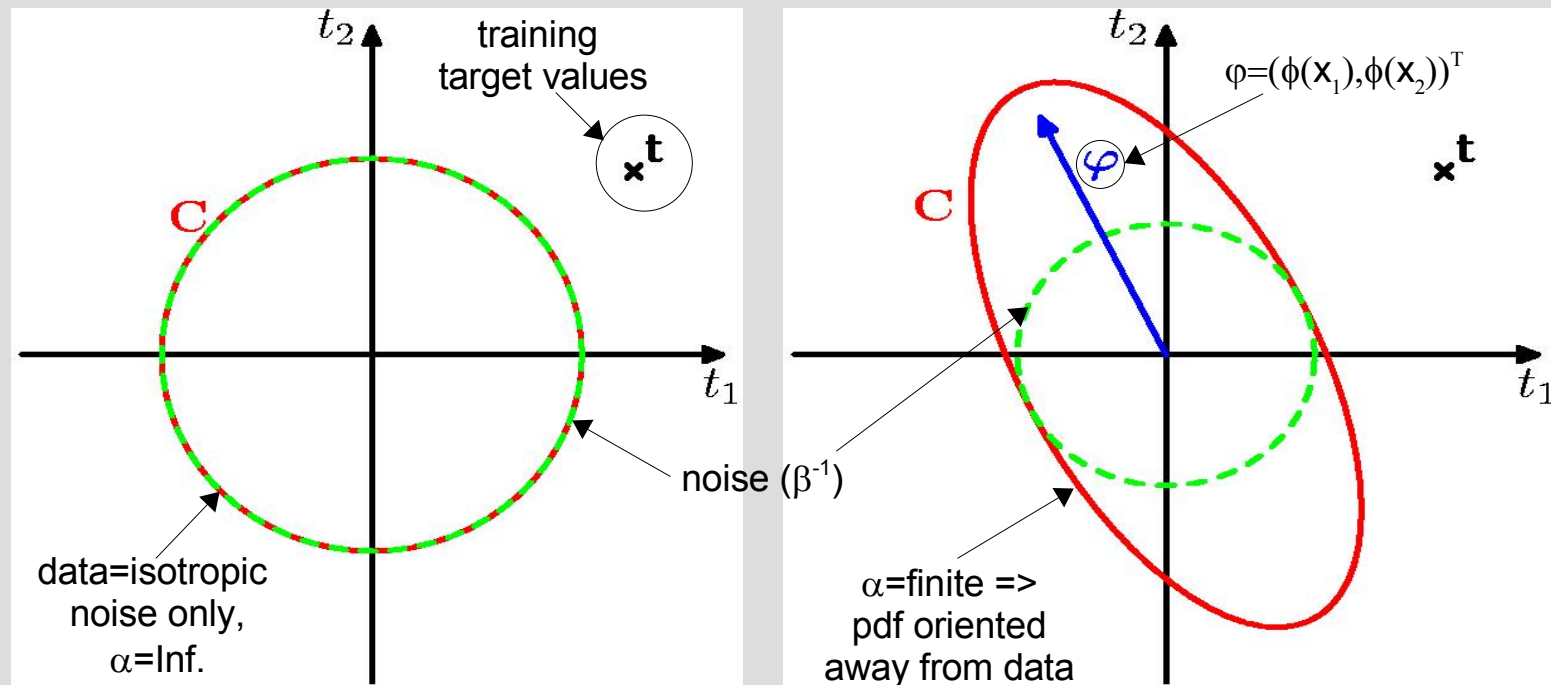


Figure 7.10

- (left): if there is a poor alignment between  $\varphi$  and training data  $t$ , then  $\alpha$  will be driven to infinity and the basis vector pruned
- (right): any finite value of  $\alpha$  causes the distribution to be oriented away from the data - reduces the probability at the target data vector => for the most probable solution, the basis vector is removed

# Relevance Vector Machines (8)

## 7.2.2 Analysis of sparsity

- Mathematical analysis of sparsity leads to a closed-form solution for  $\alpha_i$ , which can be optimized using an efficient algorithm
  - the algorithm uses a fixed set of basis vectors and cycles through them in turn to decide whether each vector should be included in the model or not
  - given in steps on pp. 352-353

# Relevance Vector Machines (9)

## 7.2.3 RVM for classification

- RVM framework can obviously be used for classification
  - apply the prior over weights (7.80) to a probabilistic linear classification model
- For two-class problems, we have binary target variables  $t$  from  $\{0, 1\}$ 
  - the model is a lin.comb. of basis functions transformed by a logistic sigmoid

$$y(\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T \phi(\mathbf{x})) \quad (7.108)$$

- however, we cannot integrate over the weights anymore; have to use Laplace approximation as in Section 4.5.1
- basically, we initialize the hyper-parameter vector  $\alpha$ , with which we build a Gaussian approximation to the post. *pdf* (i.e. to the marginal likelihood); maximizing this approximation leads to a new value for  $\alpha$ , and the process repeats until convergence

# Relevance Vector Machines (10)

## 7.2.3 RVM for classification

- The re-estimation formula for  $\alpha$  and the approximate marginal likelihood take the same form as in the regression case
  - therefore, the same analysis of sparsity can be applied
  - same fast algorithm can be designed

# Relevance Vector Machines (11)

## 7.2.3 RVM for classification

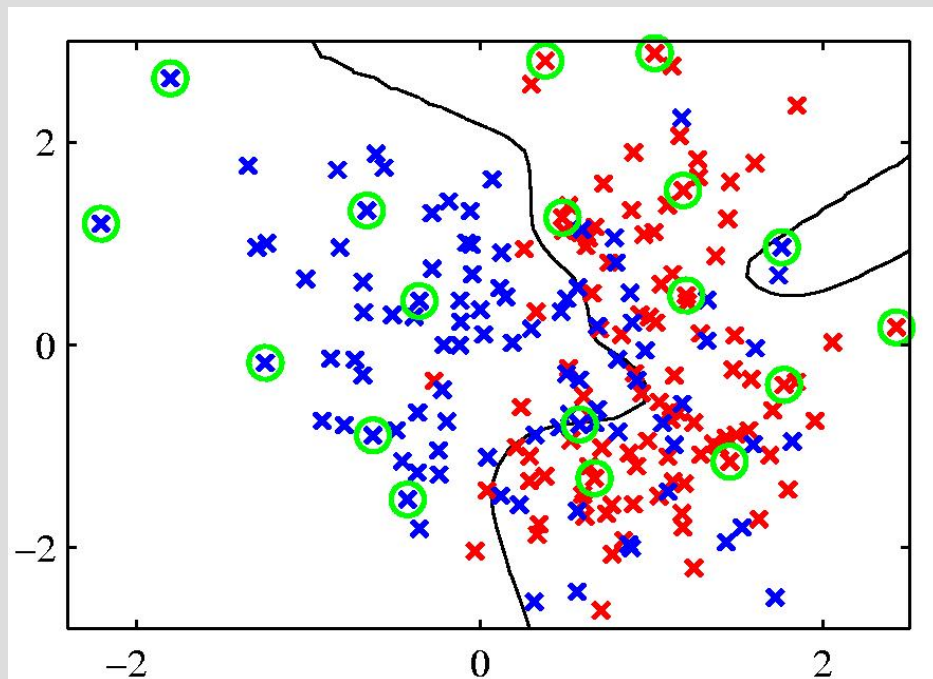


Figure 7.12 (RVM)

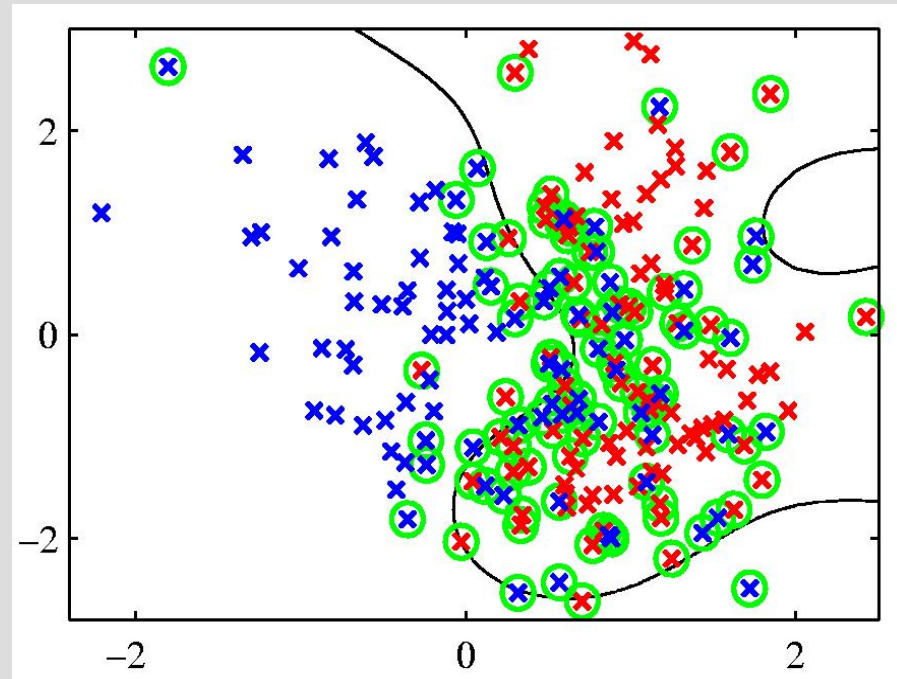


Figure 7.4 (SVM)

- contrary to SVs, RVs usually do not lie in the region of the decision boundary (remember Figure 7.10?) - basis function centered on a data point near the boundary will have a basis vector that is poorly aligned with training data
- also, RVs much sparser than SVs

# Relevance Vector Machines (12)

## 7.2.3 RVM for classification

- For  $K > 2$  classes, logistic regression (Section 4.3.4) is used
  - i.e.  $K$  linear models of the form

$$a_k = \mathbf{w}_k^T \mathbf{x} \quad (7.120)$$

- these are combined using a softmax function to provide outputs

$$y_k(\mathbf{x}) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \quad (7.121)$$

- the log-likelihood function is then

$$\ln p(\mathbf{T} | \mathbf{w}_1, \dots, \mathbf{w}_K) = \prod_{n=1}^N \prod_{k=1}^K y_{nk}^{t_{nk}} \quad (7.122)$$

- Laplace approximation is used again to optimize the hyper-parameters
  - more principled approach than the pairwise methods of SVM + provides probabilistic outputs
  - however, computational cost is higher