POLITECNICO DI MILANO

Machine Learning

Support Vector Machines

Prof. Matteo Matteucci

Discriminative vs. Generative Approaches

- ***Generative approach***: we derived the classifier from some generative hypothesis about the way data have been generated
  - Linearity of the log odds for posteriors (Logistic Regression)
  - Multivariate Gaussian given the class for the likelihood (LDA)

- ***Discriminative approach***: find the prescribed boundary (e.g., a linear separating boundary) able to reduce the classifier error
  - Define a discriminating function instead of making assumptions on the data distribution

From ESL

- *Example:* find the separating hyperplane which separates the data points in the best way (e.g., with the minimum error rate)

$$\{x : \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = 0\}$$

Perceptron

- A perceptron computes the value of a weighted sum and returns its sign (name dates back to ’50s literature on neural networks)

$$\{x : \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = 0\}$$

- Basically a perceptron is a linear classifier for which:
  - We do not assume any particular probabilistic model generating the data
  - We learn the parameters using some optimization technique so to minimize an error function (e.g., the error rate)
  - We cannot infer the role of the single variables in the model from the values of the weights
- Inference in discriminative models is quite complex and usually it is not the goal which is prediction.

An Example: Simulated Data

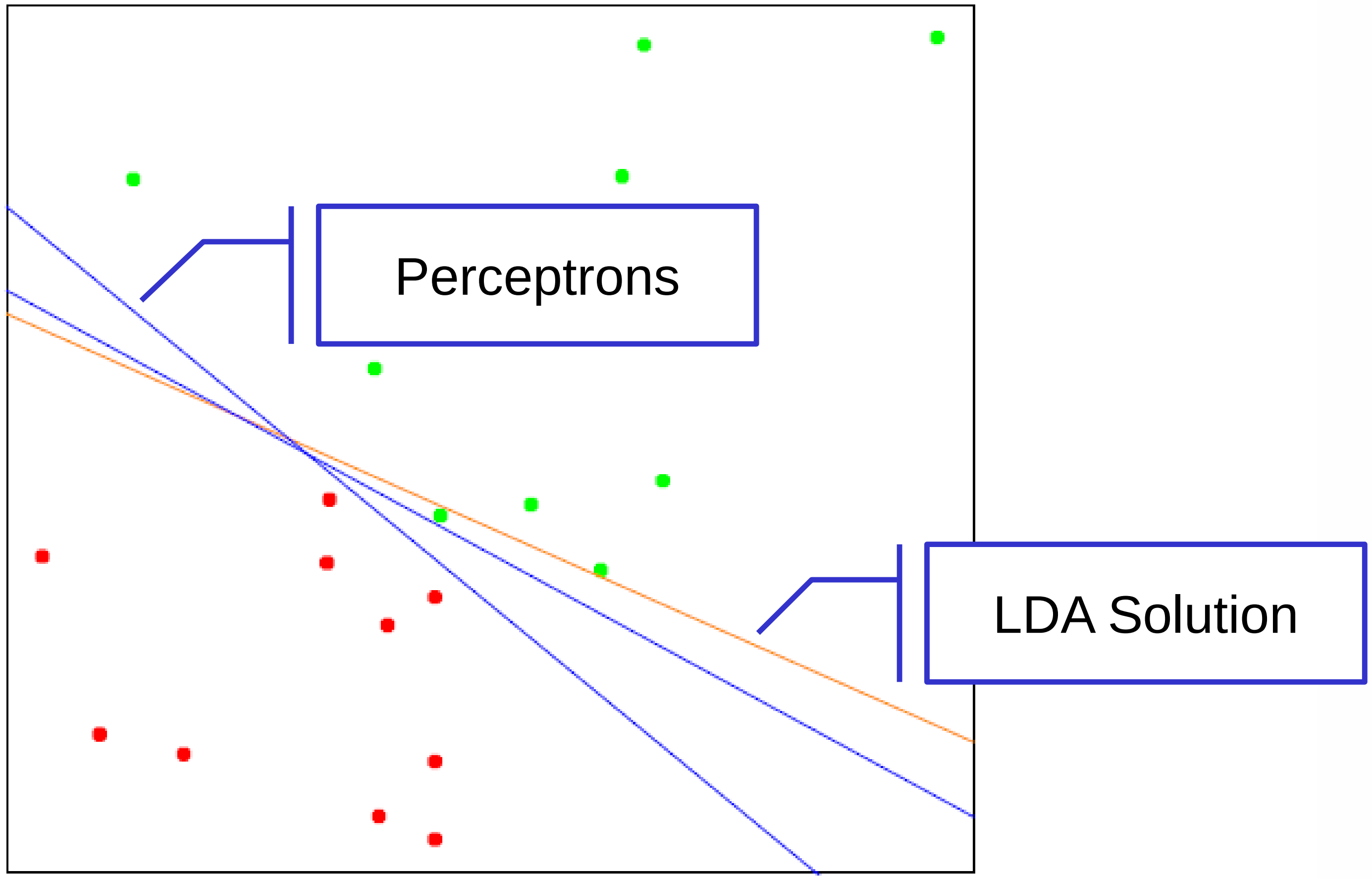
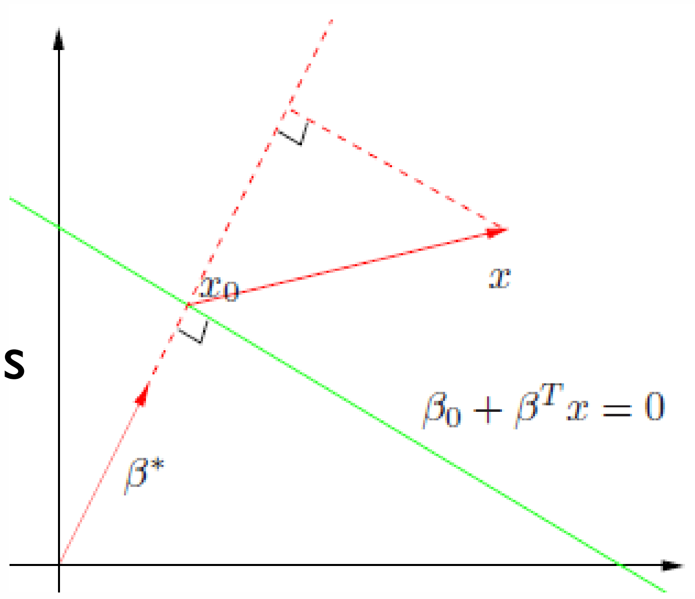


FIGURE 4.14. *A toy example with two classes separable by a hyperplane. The orange line is the least squares solution, which misclassifies one of the training points. Also shown are two blue separating hyperplanes found by the* perceptron learning algorithm *with different random starts.*

Hyperplanes Linear Algebra

- Let consider the hyperplane (affine set) ***L*** in $\mathbb{R}^2$

$$f(x) = \beta_0 + \beta^T x = 0$$



  - Any two points *x₁* and *x₂* on ***L*** have

$$\beta^T(x_1 - x_2) = 0$$

  - The vector normal to the surface ***L*** is

$$\beta^* = \beta / ||\beta||$$

  - For any point x_0 in L we have

$$\beta^T x_0 = -\beta_0$$

  - The signed distance of any point *x* to ***L*** is defined by

$$\begin{aligned} {\beta^*}^T(x - x_0) &= \frac{1}{||\beta||}(\beta^T x + \beta_0) \\ &= \frac{1}{||f'(x)||} f(x). \end{aligned}$$

f(x) proportional to the distance of x from the plane defined by *f(x)=0*

Perceptron Learning Algorithm (1/2)

- The error function for the perceptron learning is the distance of misclassified points from the decision boundary
  - The output is coded with +1/-1
  - If an output which should be +1 is misclassified

$$x_i^T \beta + \beta_0 < 0$$

  - For an output with -1 we have the opposite

- The goal becomes minimizing

Set of points misclassified

$$D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i (x_i^T \beta + \beta_0)$$

  - This is non negative and proportional to the distance of the misclassified points from

$$\beta^T x + \beta_0 = 0$$

Perceptron Learning Algorithm (2/2)

- Minimize by stochastic gradient descend the error function

$$D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i (x_i^T \beta + \beta_0)$$

  - The gradients with respect to the model parameters are

$$\partial \frac{D(\beta, \beta_0)}{\partial \beta} = -\sum_{i \in \mathcal{M}} y_i x_i,$$

$$\partial \frac{D(\beta, \beta_0)}{\partial \beta_0} = -\sum_{i \in \mathcal{M}} y_i.$$

  - Stochastic gradient descent applies for each misclassified point

$$\begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} \leftarrow \begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} + \rho \begin{pmatrix} y_i x_i \\ y_i \end{pmatrix}$$

Learning rate

- If data are linearly separable, it converges to a separating hyperplane in a finite number of steps

An Example: Simulated Data

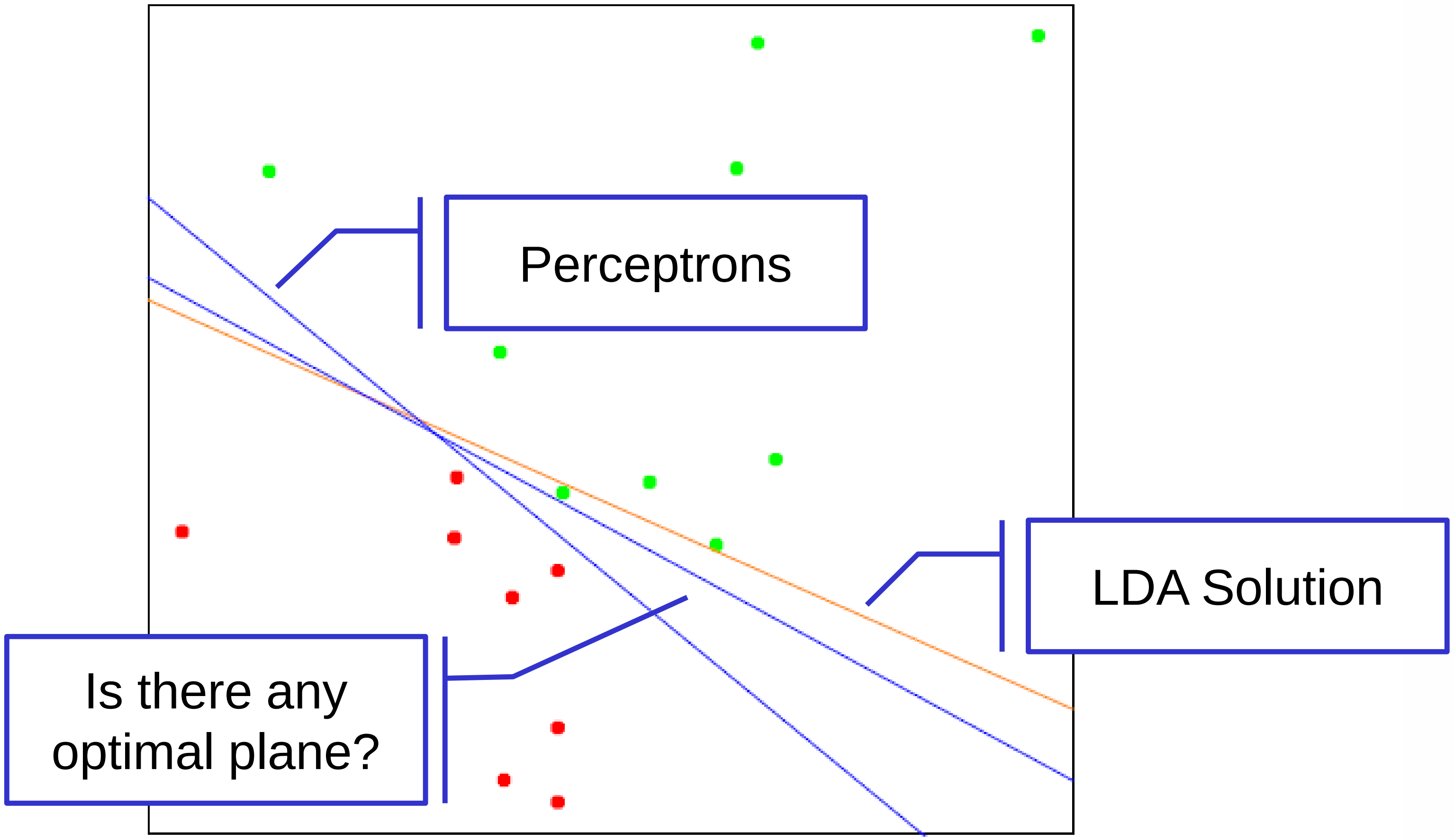


FIGURE 4.14. *A toy example with two classes separable by a hyperplane. The orange line is the least squares solution, which misclassifies one of the training points. Also shown are two blue separating hyperplanes found by the* perceptron learning algorithm *with different random starts.*

An Example: Simulated data

Maximum margin classifier …

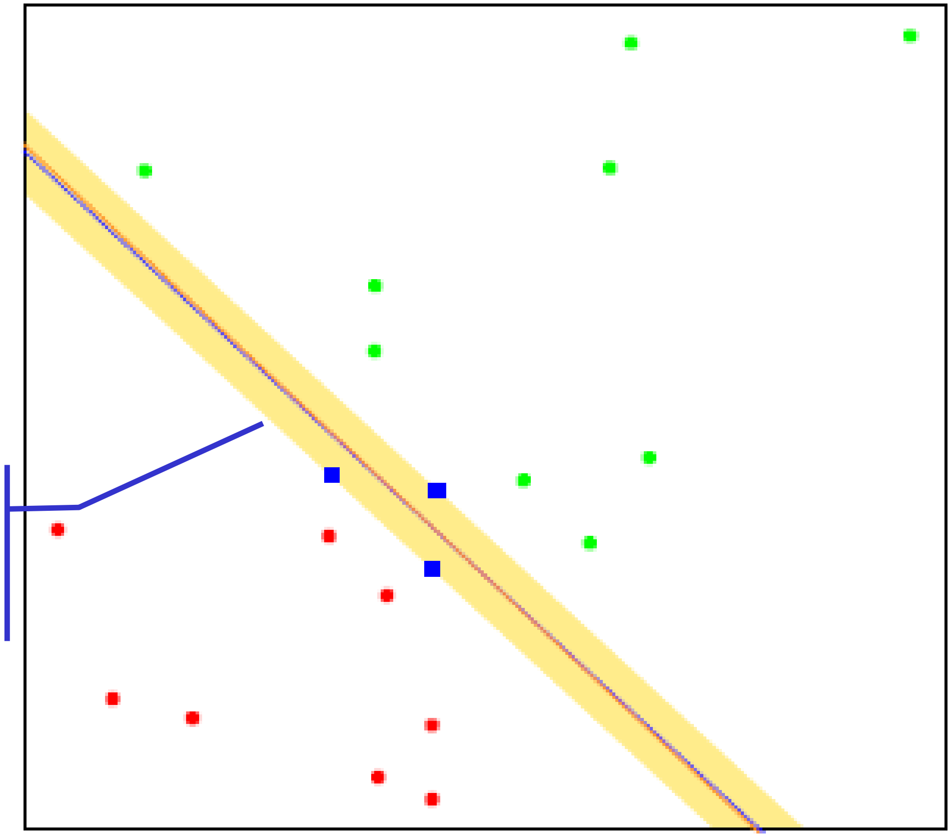


FIGURE 4.16. *The same data as in Figure 4.14. The shaded region delineates the maximum margin separating the two classes. There are three support points indicated, which lie on the boundary of the margin, and the optimal separating hyperplane (blue line) bisects the slab. Included in the figure is the boundary found using logistic regression (red line), which is very close to the optimal separating hyperplane (see Section 12.3.3).*

Separable Case Formulation (1)

- Maximize the margin M

From ESL

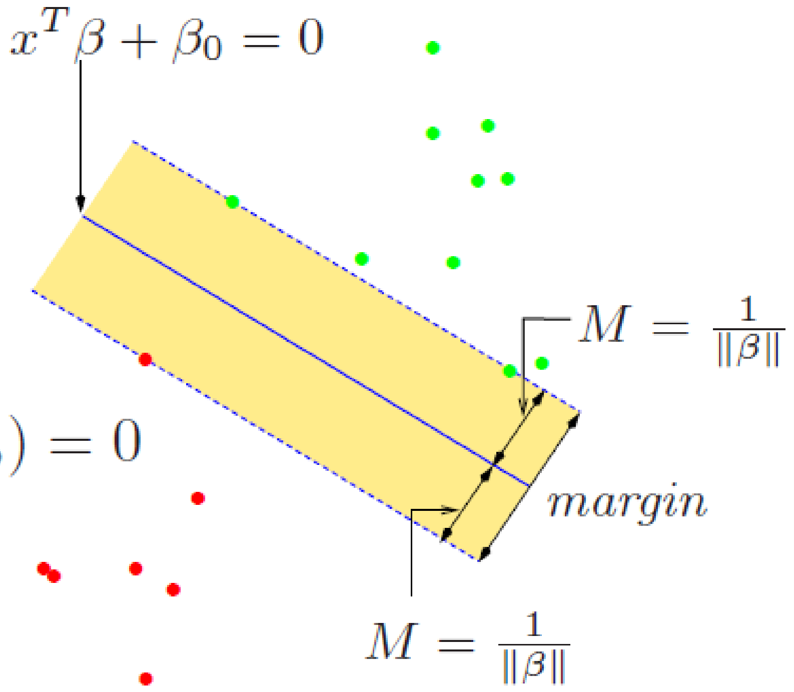
$$\max_{\beta,\beta_0,||\beta||=1} M$$

$$\text{subject to } y_i(x_i^T\beta+\beta_0) \geq M,\ i=1,\ldots,N$$

- The two constraints respectively
  - Select one of the possible hyper planes

$$k(\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\ldots+\beta_p x_{ip})=0$$

  - Each point is on the right side of the margin



Separable Case Formulation (2)

- We can remove the constraint on the parameter norm by changing the margin constraint

$$\frac{1}{||\beta||} y_i(x_i^T \beta + \beta_0) \geq M$$

  - which becomes

$$y_i(x_i^T \beta + \beta_0) \geq M||\beta||$$

- If we redefine $M = 1/||\beta||$ we obtain the equivalent problem

$$\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2$$
$$\text{subject to } y_i(x_i^T \beta + \beta_0) \geq 1,\ i = 1, \dots, N$$

- Which is a “simple” convex optimization problem (quadratic with linear constraints)

Separable Case Solution (1)

- We can solve the constrained quadratic problem by the use of Lagrange multipliers

$$L_P = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{N} \alpha_i [y_i(x_i^T \beta + \beta_0) - 1]$$

  - Setting the derivatives to zero we have

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i,$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i,$$

Can you derive it?

  - And by substitution the so-called Wolf dual

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k$$

$$\text{subject to} \qquad \alpha_i \geq 0.$$

Separable Case Solution (2)

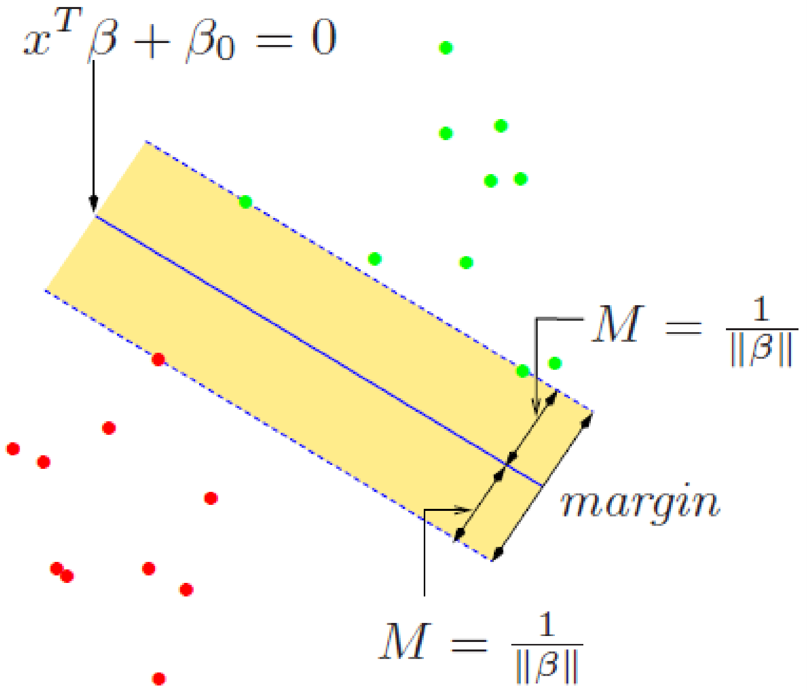
- The Karush-Kuhn-Tucker conditions must hold too

$$\alpha_i[y_i(x_i^T\beta + \beta_0) - 1] = 0 \; \forall i.$$

  - If $\alpha_i > 0$ then we have $y_i(x_i^T\beta + \beta_0) = 1$
  - If $y_i(x_i^T\beta+\beta_0) > 1$ this means $\alpha_i = 0$

- The final output comes from:

$$\hat{G}(x) = \operatorname{sign}\hat{f}(x)$$



Non Separable Case Formulation (1)

- Maximize the margin M

$$\max_{\beta,\beta_0,||\beta||=1} M$$

$$\text{subject to } y_i(x_i^T\beta+\beta_0)\geq M,\ i=1,\ldots,N$$

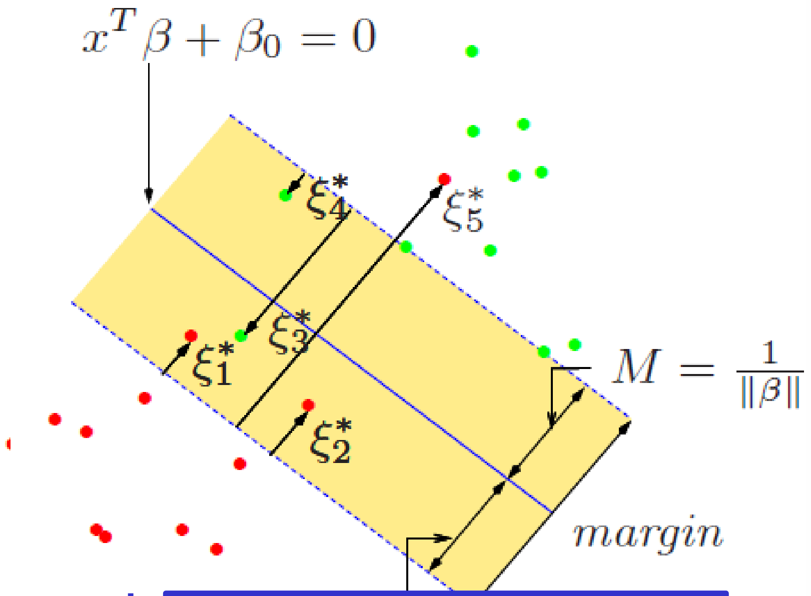
- To accommodate for "errors" we use some extra variables …

$$y_i(x_i^T\beta+\beta_0) \geq M-\xi_i,$$

or

$$y_i(x_i^T\beta+\beta_0) \geq M(1-\xi_i),$$

$$\forall i,\ \xi_i\geq 0,\ \sum_{i=1}^N \xi_i \leq \text{constant}$$



This gives a convex optimization problem

Non Separable Case Formulation (2)

- We can remove the constraint on the parameter norm obtaining

$$\min \|\beta\| \quad \text{subject to} \begin{cases} y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \ \forall i, \\ \xi_i \geq 0, \ \sum \xi_i \leq \text{constant}. \end{cases}$$

From ESL

- This can be rewritten as

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to} \quad \xi_i \geq 0, \ y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \ \forall i,$$

Used defined cost

Infinite corresponds to separable case …

- Solved by Lagrange multipliers as for the separable case …

Non Separable Case Solution (1)

- The primal Lagrange function is

$$L_P = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta+\beta_0)-(1-\xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i,$$

  - Setting the derivatives to zero gives

$$\begin{aligned}\beta &= \sum_{i=1}^{N}\alpha_i y_i x_i,\\ 0 &= \sum_{i=1}^{N}\alpha_i y_i,\\ \alpha_i &= C-\mu_i,\ \forall i,\end{aligned}$$

  - with $\alpha_i,\ \mu_i,\ \xi_i \geq 0\ \forall i$

- By substitution we obtain the Lagrangian (Wolf) dual function …

Non Separable Case Solution (2)

Can you derive it?

- The dual Lagrange function is

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} x_i^T x_{i'},$$

  - Subject to $0 \le \alpha_i \le C$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$
  - Having the Karush-Kuhn-Tucker conditions

$$\begin{aligned} \alpha_i[y_i(x_i^T\beta + \beta_0) - (1 - \xi_i)] &= 0, \\ \mu_i \xi_i &= 0, \\ y_i(x_i^T\beta + \beta_0) - (1 - \xi_i) &\ge 0, \end{aligned}$$

- Solving the optimization problem we have

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i,$$

  - computed using only the support vectors …

Non Separable Case Solution (3)

- The solution of the Dual optimization problem provides

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i,$$

- Karush-Kuhn-Tucker conditions imply

$$\begin{aligned} \alpha_i[y_i(x_i^T\beta + \beta_0) - (1 - \xi_i)] &= 0, \\ \mu_i \xi_i &= 0, \\ y_i(x_i^T\beta + \beta_0) - (1 - \xi_i) &\geq 0, \end{aligned}$$

  - $\hat{\alpha}_i$ is non zero only for support vectors
  - If $\hat{\xi}_i = 0$ the support is on the margin and $0 < \hat{\alpha}_i < C$
  - If $\hat{\xi}_i > 0$ we have $\hat{\alpha}_i = C$
  - Using the margin points $0 < \hat{\alpha}_i, \ \hat{\xi}_i = 0$ we can solve for β_0

Separable Data vs Non Separable Data

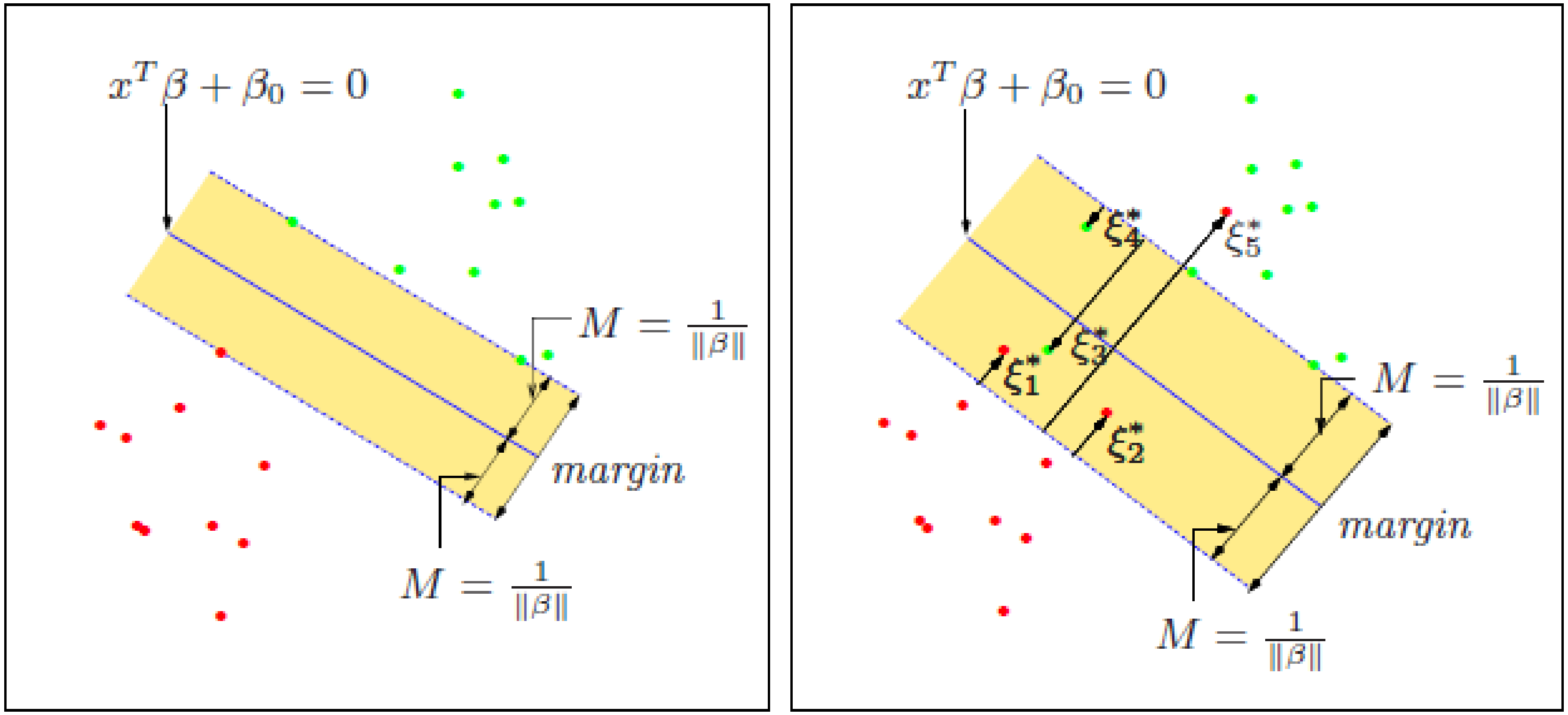
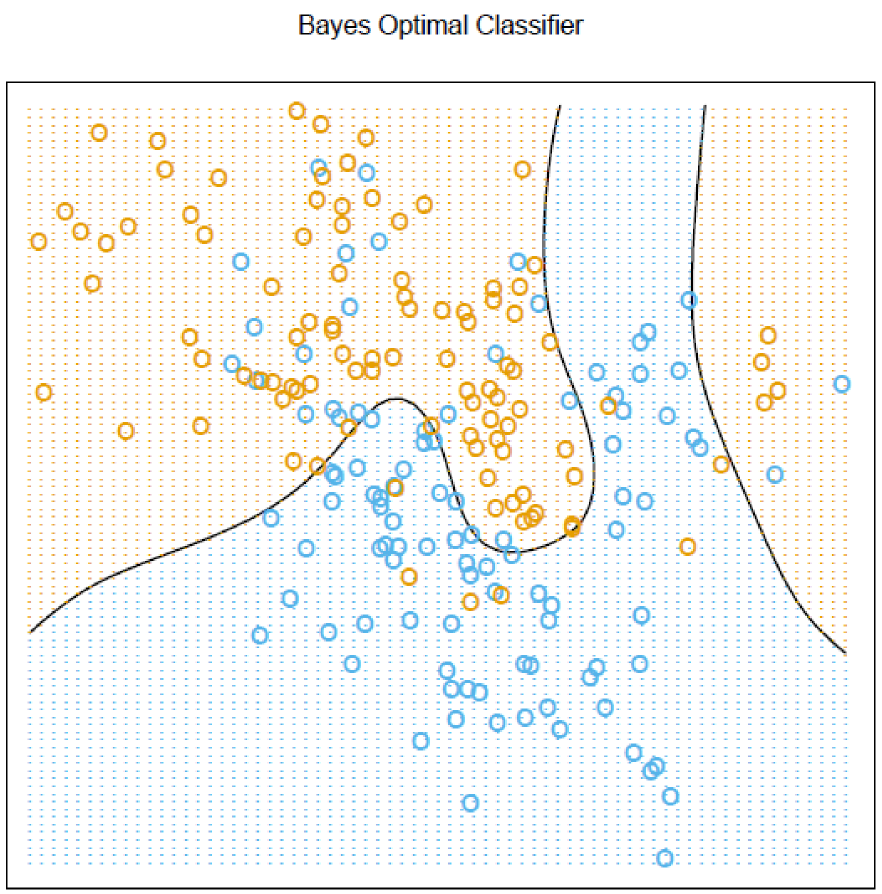


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width $2M = 2/\|\beta\|$. The right panel shows the nonseparable (overlap) case. The points labeled ξ_j^* are on the wrong side of their margin by an amount $\xi_j^* = M\xi_j$; points on the correct side have $\xi_j^* = 0$. The margin is maximized subject to a total budget $\sum \xi_i \leq$ constant. Hence $\sum \xi_j^*$ is the total distance of points on the wrong side of their margin.*

A Synthetic Example



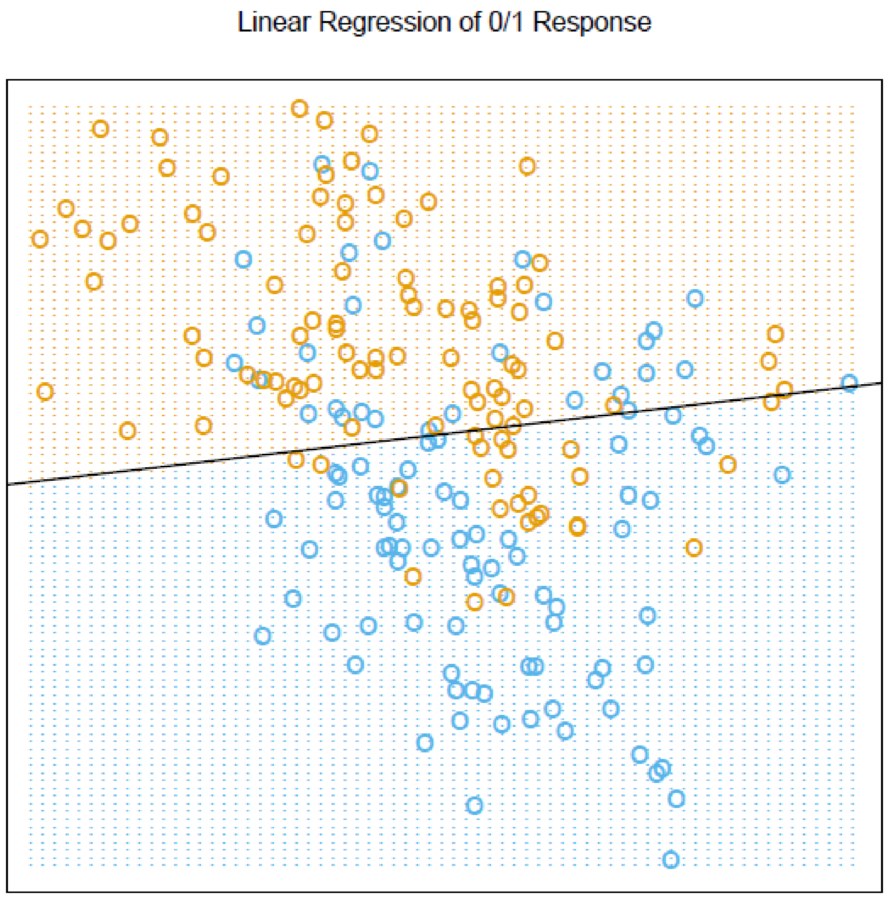
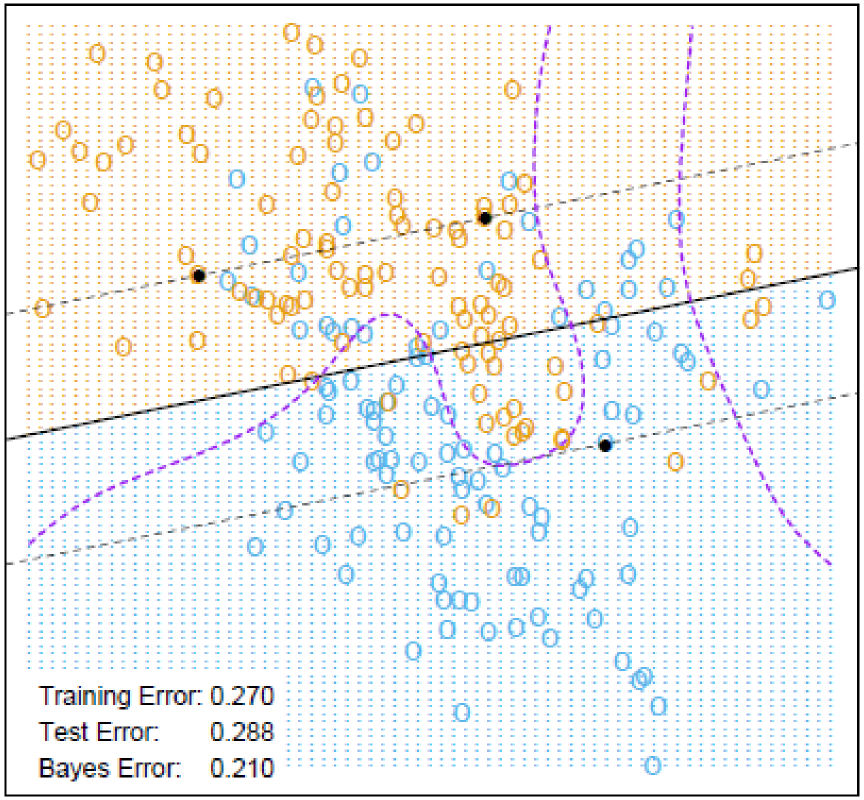
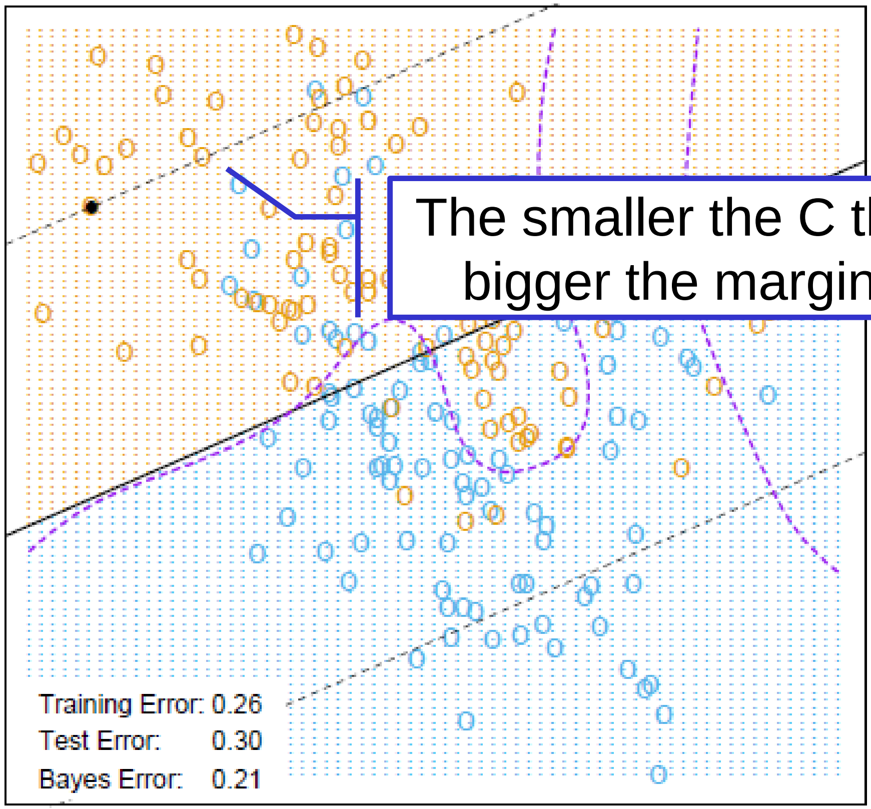


FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (*BLUE $= 0$, ORANGE $= 1$*), and then fit by linear regression. The line is the decision boundary defined by $x^T\hat{\beta} = 0.5$. The orange shaded region denotes that part of input space classified as* ORANGE*, while the blue region is classified as* BLUE.

A Synthetic Example



$C = 10000$



$C = 0.01$

The smaller the C the bigger the margin

FIGURE 12.2. *The linear support vector boundary for the mixture data example with two overlapping classes, for two different values of C. The broken lines indicate the margins, where $f(x) = \pm 1$. The support points ($\alpha_i > 0$) are all the points on the wrong side of their margin. The black solid dots are those support points falling exactly on the margin ($\xi_i = 0$, $\alpha_i > 0$). In the upper panel 62% of the observations are support points, while in the lower panel 85% are. The broken purple curve in the background is the Bayes decision boundary.*

Support Vector Machines (1)

- Learning the classifier involves only the scalar product of features

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} \underbrace{\langle h(x_i), h(x_{i'}) \rangle}_{\text{Scalar product}}$$

- We can compute this scalar product in a new feature space

$$h_m(x),\ m = 1, \ldots, M$$

$$h(x_i) = (h_1(x_i), h_2(x_i), \ldots, h_M(x_i)),\ i = 1, \ldots, N,$$

$$\hat{f}(x) = h(x)^T \hat{\beta} + \hat{\beta}_0$$

$$\hat{G}(x) = \text{sign}(\hat{f}(x))$$

- The feature space can grow up to infinity with SVM …

$$\begin{aligned} f(x) &= h(x)^T \beta + \beta_0 \\ &= \sum_{i=1}^{N} \alpha_i y_i \underbrace{\langle h(x), h(x_i) \rangle}_{\text{Scalar product}} + \beta_0 \end{aligned}$$

The Kernel Trick

- What we need for learning and prediction is the result of the scalar product only

$$\begin{aligned} f(x) &= h(x)^T\beta + \beta_0 \\ &= \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i)\rangle + \beta_0 \end{aligned}$$

- In some cases this scalar product can be written as a Kernel

$$K(x, x') = \langle h(x), h(x')\rangle$$

- Popular choices are

$$\begin{aligned} d\text{th-Degree polynomial: } K(x, x') &= (1 + \langle x, x'\rangle)^d, \\ \text{Radial basis: } K(x, x') &= \exp(-\gamma \|x - x'\|^2), \\ \text{Neural network: } K(x, x') &= \tanh(\kappa_1 \langle x, x'\rangle + \kappa_2). \end{aligned}$$

The Kernel Trick Example

- Consider an Input space with two variables and a polynomial kernel of degree 2

$$\begin{aligned} K(X, X') &= (1 + \langle X, X' \rangle)^2 \\ &= (1 + X_1 X_1' + X_2 X_2')^2 \\ &= 1 + 2X_1X_1' + 2X_2X_2' + (X_1X_1')^2 + (X_2X_2')^2 + 2X_1X_1'X_2X_2'. \end{aligned}$$

- It turns out the that M=6 and if we chose

$$\begin{aligned} h_1(X) &= 1 & h_4(X) &= X_1^2 \\ h_2(X) &= \sqrt{2}X_1 & h_5(X) &= X_2^2 \\ h_3(X) &= \sqrt{2}X_2 & h_6(X) &= \sqrt{2}X_1X_2 \end{aligned}$$

- We obtain

$$K(X, X') = \langle h(X), h(X') \rangle$$

A Synthetic Example (Non Linear)

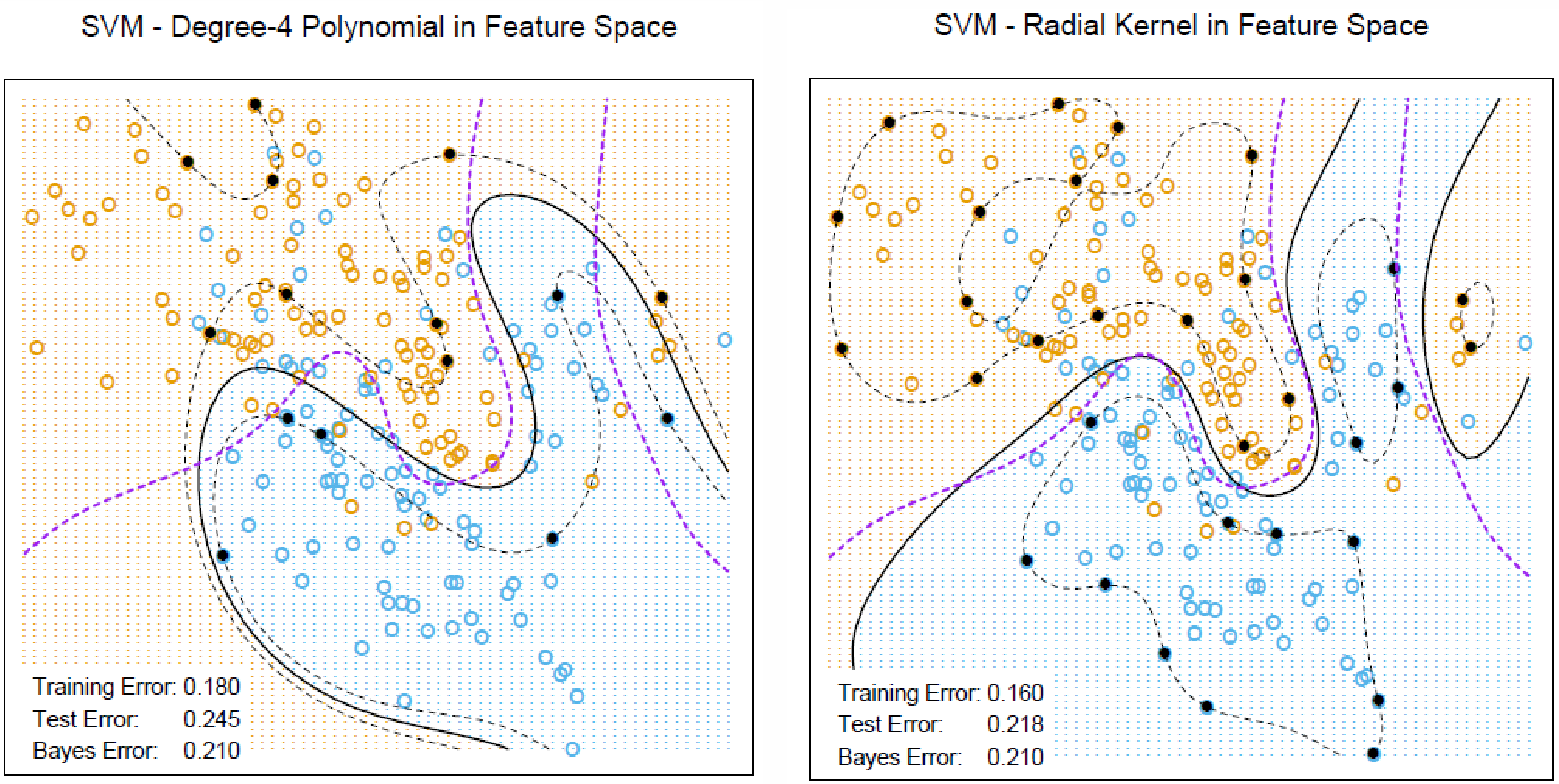


FIGURE 12.3. *Two nonlinear SVMs for the mixture data. The upper plot uses a 4th degree polynomial kernel, the lower a radial basis kernel (with $\gamma = 1$). In each case C was tuned to approximately achieve the best test error performance, and $C = 1$ worked well in both cases. The radial basis kernel performs the best (close to Bayes optimal), as might be expected given the data arise from mixtures of Gaussians. The broken purple curve in the background is the Bayes decision boundary.*