

Slides – Support Vector Machine

Elements of Statistical Learning

Professors: Aleixo, S.; Geraldes, C.; Pinto, I.

Master in Applied Mathematics for Industry

DM – ISEL

These slides are mostly based in the book
“The Elements of Statistical Learning”, Hastie, T., Tibshirani, R. and Friedman, J., Springer, 2017.

Support Vector Machine

- In machine learning, **support-vector machines (SVM)** are **supervised learning models with associated learning algorithms** that **analyze data for classification and regression analysis**.
- **SVM**
 - **an approach for classification**, developed between 1992 and 1997 by Vapnik with colleagues among which Boser, Guyon and Cortes;
 - **one of the most robust prediction methods**, being based on statistical learning frameworks;
 - have been shown to **perform well in a variety of settings**, and are often considered **one of the best “out of the box” classifiers**.
- Given a set of training examples, each marked as belonging to one of two categories, an **SVM training algorithm** builds a model that assigns new examples to one category or the other, making it a **non-probabilistic binary linear classifier**.

Support Vector Machine

- **SVM** is a generalization of a simple and intuitive classifier called the **maximal margin classifier (MMC)**.
- This **MMC** classifier unfortunately cannot be applied to most data sets, since it **requires that the classes be separable by a linear boundary**.
- The **support vector classifier (SVC)** is an extension of the maximal margin classifier, **MMC**, that can be applied in a broader range of cases.
- The **support vector machine (SVM)** is a further extension of the support vector classifier in order to accommodate non-linear class boundaries.
- SVM can be applied for two (binary classification) or more classes.

Support Vector Machine

- SVM maps training examples to points in space so as to maximise the width of the gap between the two categories.
- New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.
- In addition to performing linear classification, **SVM can efficiently perform a non-linear classification** using what is called **the kernel trick**, implicitly mapping their inputs into high-dimensional feature spaces.

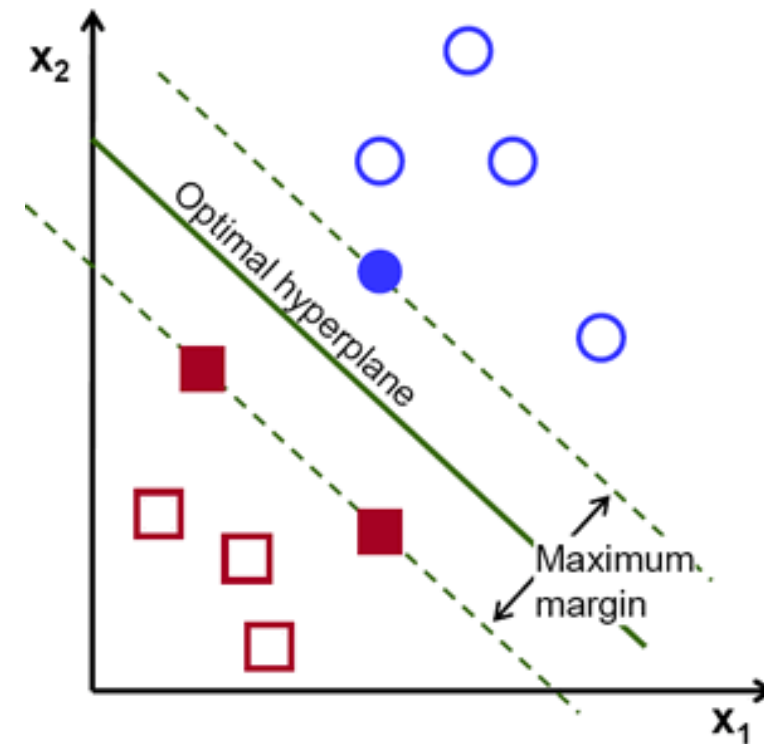
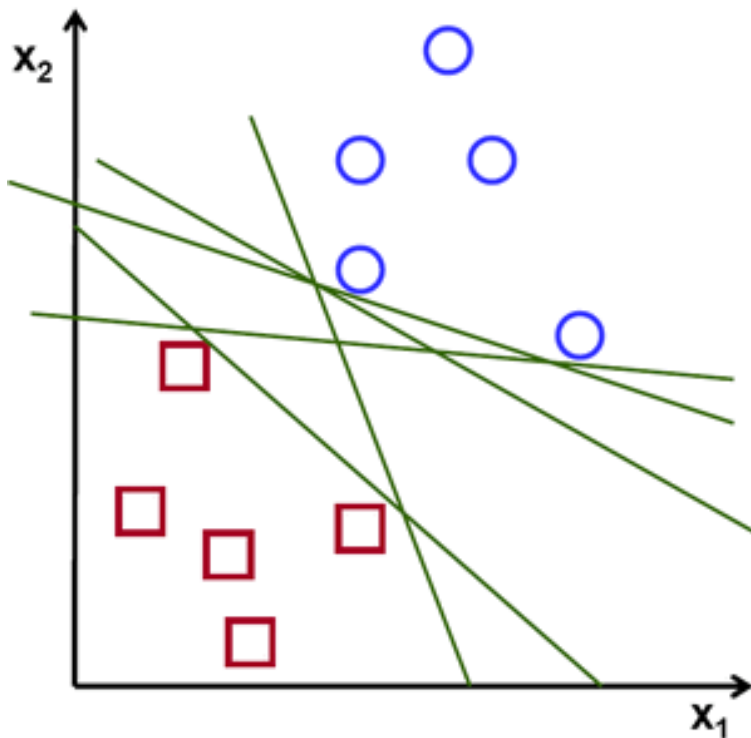
Support Vector Machine

- It is advisable to look at linear regression and logistic regression algorithms before moving on to support vector machine.
- **SVM**
 - is another **simple algorithm that every machine learning expert should know.**
 - is highly preferred by many as it **produces significant accuracy with less computation power.**
 - **can be used for both regression and classification tasks,** but it is widely used in classification objectives.

What is Support Vector Machine?

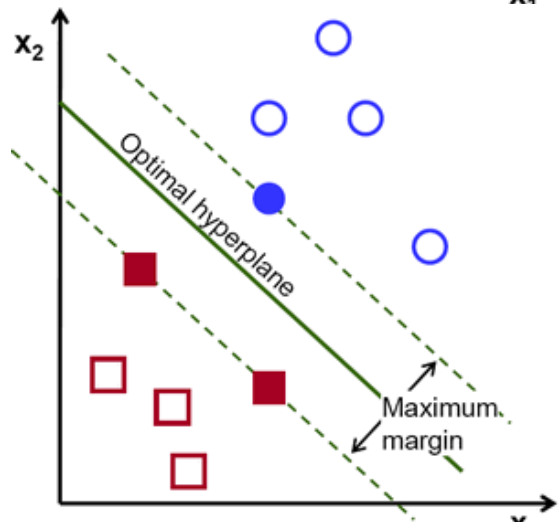
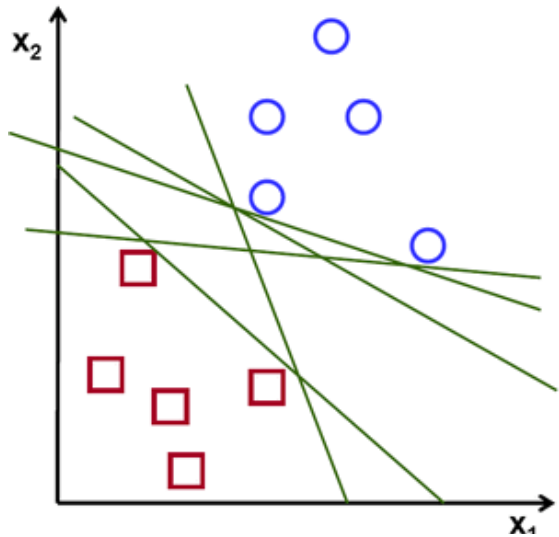
Objective of the support vector machine algorithm

➡ to find a hyperplane in an n -dimensional space (n - the number of features) that distinctly classifies the data points.



Possible hyperplanes

What is Support Vector Machine?

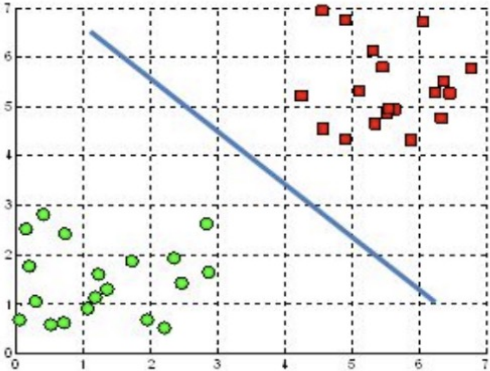


Possible hyperplanes

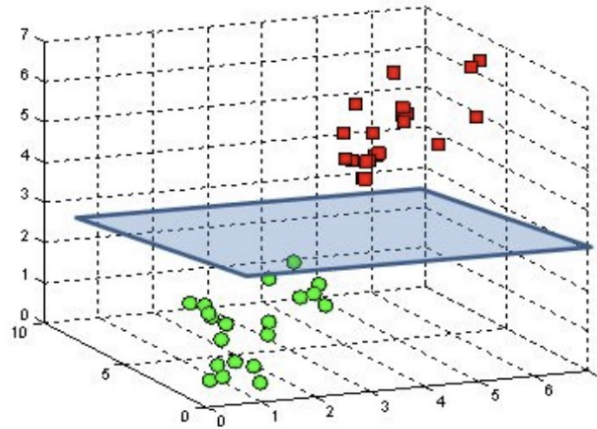
- To separate the two classes of data points, there are many possible hyperplanes that could be chosen.
- **Objective:** to find a plane that has the **maximum margin**, i.e the maximum distance between data points of both classes.
- Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Hyperplanes and Support Vectors

A hyperplane in \mathbb{R}^2 is a line



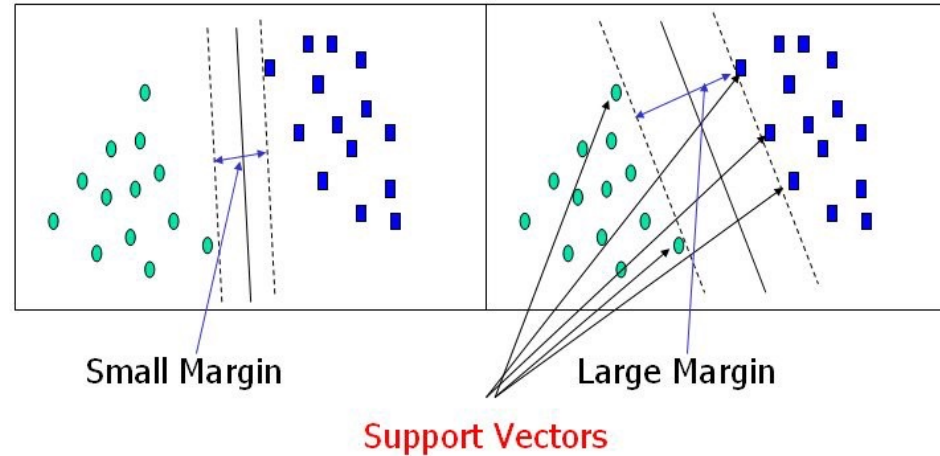
A hyperplane in \mathbb{R}^3 is a plane



Hyperplanes in 2D and 3D feature space

- **Hyperplanes** are decision boundaries that help classify the data points.
- **Data points** falling on either side of the hyperplane can be attributed to different classes.
- Also, the **dimension of the hyperplane depends upon the number of features**.
- If the **number of input features is 2**, then the **hyperplane** is just a **line**.
- If the **number of input features is 3**, then the **hyperplane** becomes a two-dimensional **plane**.
- It becomes **difficult to imagine** when the number of features exceeds 3.

Hyperplanes and Support Vectors



- **Support vectors** are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane.
- Using these support vectors, we maximize the margin of the classifier.
- Deleting the support vectors will change the position of the hyperplane.
- These are the **points that help us build the SVM**.

Hyperplanes and Support Vectors

- In a p -dimensional space, a **hyperplane** is a flat affine subspace of dimension $p-1$.

(The word affine indicates that the subspace need not pass through the origin)

- In **two dimensions**, a hyperplane is a flat one-dimensional subspace, that is, a line.
- In **three dimensions**, a hyperplane is a flat two-dimensional subspace, that is, a plane.
- In $p > 3$ dimensions, it can be hard to visualize a hyperplane, but the notion of a $(p-1)$ -dimensional flat subspace still applies.

Hyperplanes and Support Vectors

Mathematical definition of a hyperplane

- A **hyperplane in two dimensions** is defined by the equation

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

for parameters β_0 , β_1 and β_2 .

Any $X = (X_1, X_2)^T$ for which the previous equation holds is a point on the hyperplane.

- A **p -dimensional hyperplane** is defined by the equation

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p = 0$$

If a point $X = (X_1, X_2, \dots, X_p)^T$ in p -dimensional space (i.e. a vector of length p) satisfies the previous equation, then X lies on the hyperplane.

Hyperplanes and Support Vectors

- If $X = (X_1, X_2, \dots, X_p)^T$ verifies

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p > 0$$

Then X lies to one side of the hyperplane.

- On the other hand, if

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p < 0,$$

X lies on the other side of the hyperplane.

- So we can think of the hyperplane as dividing p-dimensional space into two halves. One can easily **determine on which side of the hyperplane a point lies** by simply **calculating the sign of**

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$$

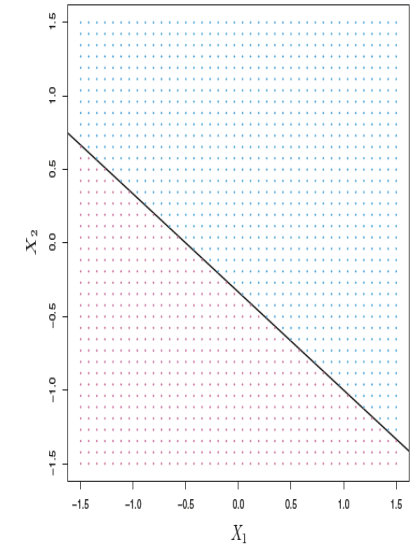


FIGURE ... The hyperplane $1 + 2X_1 + 3X_2 = 0$ is shown. The blue region is the set of points for which $1 + 2X_1 + 3X_2 > 0$, and the purple region is the set of points for which $1 + 2X_1 + 3X_2 < 0$.

Classification Using a Separating Hyperplane

- Suppose that we have:

- a $n \times p$ data matrix X that consists of n training observations in p -dimensional space

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

and **these observations fall into two classes**, that is,

$$y_1, \dots, y_n \in \{-1, 1\}$$

where -1 represents one class and 1 the other class.

- a test observation, a p -vector of observed features $x^* = (x_1^*, \dots, x_p^*)^T$.

Classification Using a Separating Hyperplane

Goal:

- to develop a classifier based on the training data that will correctly classify the test observation using its feature measurements.

Approaches for this task:

- logistic regression
- linear discriminant analysis
- classification trees
- new approach that is based upon the concept of a separating hyperplane.

Classification Using a Separating Hyperplane

- Suppose that it is possible to construct a hyperplane that separates the hyperplane training observations perfectly according to their class labels.

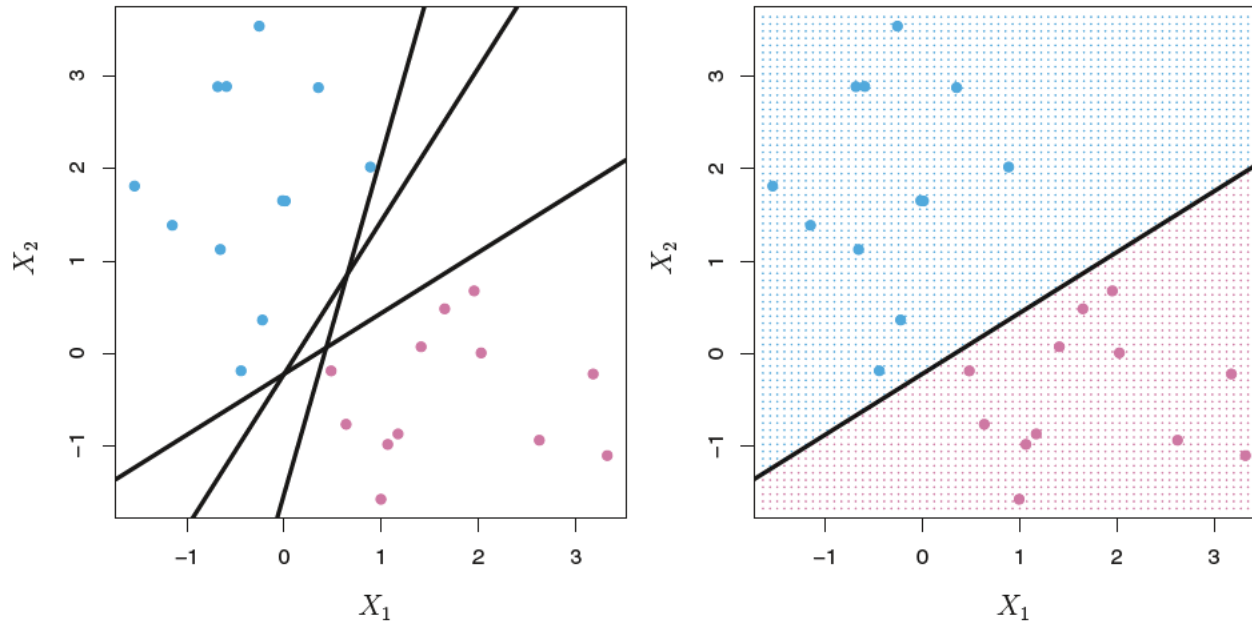


FIGURE Left: There are two classes of observations, shown in blue and in purple, each of which has measurements on two variables. Three separating hyperplanes, out of many possible, are shown in black. Right: A separating hyperplane is shown in black. The blue and purple grid indicates the decision rule made by a classifier based on this separating hyperplane: a test observation that falls in the blue portion of the grid will be assigned to the blue class, and a test observation that falls into the purple portion of the grid will be assigned to the purple class.

- Examples of three such separating hyperplanes are shown in the left-hand panel of Figure.
- We can label the observations from the **blue class** as $y_i = 1$ and those from the **purple class** as $y_i = -1$.
- Then a **separating hyperplane has the property** that
$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} > 0 \text{ if } y_i = 1,$$
and
$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} < 0 \text{ if } y_i = -1.$$
- Equivalently, a **separating hyperplane has the property** that
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) > 0$$
for all $i = 1, \dots, n$.

Classification Using a Separating Hyperplane

- If a separating hyperplane exists, we can use it to construct a very **natural classifier**:

a test observation is assigned a class depending on which side of the hyperplane it is located.

Classification Using a Separating Hyperplane

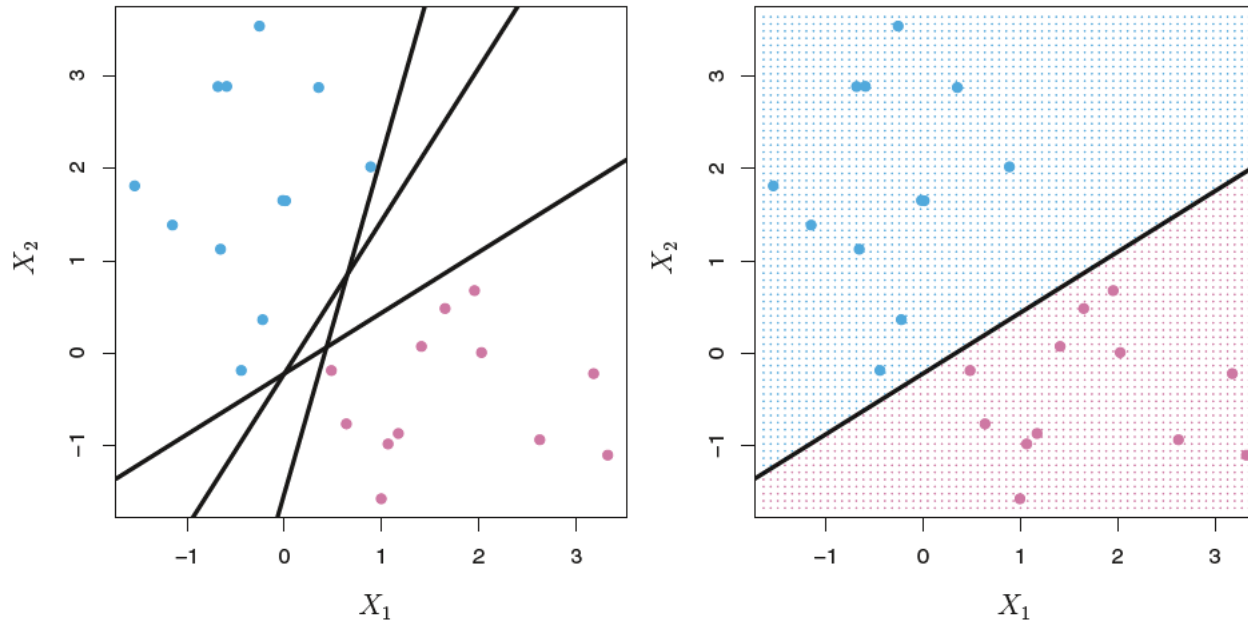


FIGURE Left: There are two classes of observations, shown in blue and in purple, each of which has measurements on two variables. Three separating hyperplanes, out of many possible, are shown in black. Right: A separating hyperplane is shown in black. The blue and purple grid indicates the decision rule made by a classifier based on this separating hyperplane: a test observation that falls in the blue portion of the grid will be assigned to the blue class, and a test observation that falls into the purple portion of the grid will be assigned to the purple class.

- The right-hand panel of Figure shows an example of such a classifier.

That is, we **classify the test observation x^* based on the sign of**

$$f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \dots + \beta_p x_p^*$$

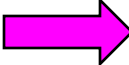
- If $f(x^*)$ is positive, then we assign the test observation to class 1, and if $f(x^*)$ is negative, then we assign it to class -1 .
- We can also make use of the magnitude of $f(x^*)$.
 - If $f(x^*)$ is far from zero, then x^* lies far from the hyperplane, and so we can be confident about our class assignment for x^* .
 - If $f(x^*)$ is close to zero, then x^* is located near the hyperplane, and so we are less certain about the class assignment for x^* .

The Maximal Margin Classifier (MMC)

- In general, if our data can be perfectly separated using a hyperplane, then there will in fact **exist an infinite number of separating hyperplanes**.
- This is because a given separating hyperplane can usually be shifted a tiny bit up or down, or rotated, without coming into contact with any of the observations.
- Three possible separating hyperplanes are shown in the left-hand panel of the previous Figure.
- **In order to construct a classifier based upon a separating hyperplane, we must have a reasonable way to decide which of the infinite possible separating hyperplanes to use.**
- A natural choice is the **maximal margin hyperplane** (also known as the optimal separating hyperplane), which is the **separating hyperplane is farthest from the training observations**.

The Maximal Margin Classifier (MMC)

Maximal margin hyperplane

- We can compute the (perpendicular) distance from each training observation to a given separating hyperplane; the smallest such distance is the **minimal distance from the observations to the hyperplane**, and is known as the **margin**.
- The **maximal margin hyperplane** is the **separating hyperplane for which the margin is largest**, that is, it is the hyperplane that has the farthest minimum distance to the training observations.
- We can then classify a test observation based on which side of the maximal margin hyperplane it lies
 **maximal margin classifier**.
- We hope that a classifier that has a **large margin on the training data will also have a large margin on the test data**, and hence **will classify the test observations correctly**.
- Although the maximal margin classifier is **often successful**, **it can also lead to overfitting when p is large**.
- If $\beta_0, \beta_1, \dots, \beta_p$ are the coefficients of the maximal margin hyperplane, then the **maximal margin classifier classifies the test observation x^* based on the sign of**

$$f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \dots + \beta_p x_p^*.$$

The Maximal Margin Classifier (MMC)

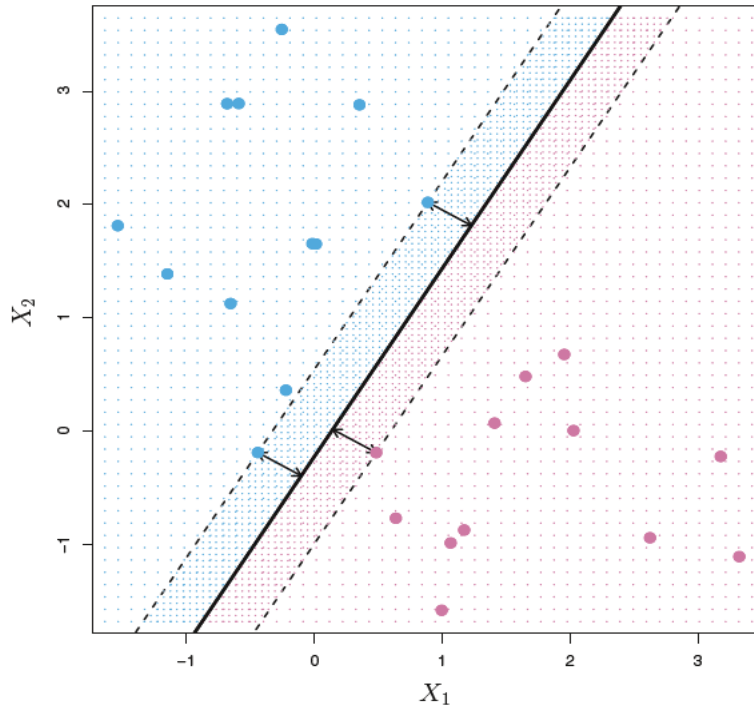


FIGURE 1 There are two classes of observations, shown in blue and in purple. The maximal margin hyperplane is shown as a solid line. The margin is the distance from the solid line to either of the dashed lines. The two blue points and the purple point that lie on the dashed lines are the support vectors, and the distance from those points to the margin is indicated by arrows. The purple and blue grid indicates the decision rule made by a classifier based on this separating hyperplane.

- Three training observations are **equidistant from the maximal margin hyperplane** and lie along the dashed lines indicating the width of the margin.
- These three observations are known as **support vectors**, since they are **vectors in p-dimensional space** (in Figure $p = 2$) and they “**support**” the maximal margin hyperplane in the sense that if these points were moved slightly then the maximal margin hyperplane would move as well.
- The maximal margin hyperplane depends directly on the support vectors, but not on the other observations: a movement to any of the other observations would not affect the separating hyperplane, provided that the observation’s movement does not cause it to cross the boundary set by the margin.

- The fact that the maximal margin hyperplane depends directly on only a small subset of the observations is an important **property**.

The Support Vector Classifier

- Using **linear methods for classification** it is possible to use a technique for constructing an optimal separating hyperplane between two perfectly **separated classes**.
- One can generalize this to the **nonseparable case**, where the **classes may not be separable by a linear boundary**.
- The **support vector classifier** can be used for both cases: **separable** and **nonseparable**.
- The **support vector classifier for separable case** coincides with **maximal margin classifier**. Let's see the problem now in matrix form.

The Support Vector Classifier

- Training data

n pairs $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$.

- Hyperplane

$\{x : f(x) = x^T \beta + \beta_0 = 0\}$ (*), where β is a unit vector: $\|\beta\| = 1$.

- A **classification rule** induced by $f(x)$

$$G(x) = \text{sign}[x^T \beta + \beta_0].$$

- $f(x)$ in (*) gives the **signed distance** from a point x to the **hyperplane** $f(x) = x^T \beta + \beta_0 = 0$.

The Support Vector Classifier - separable case (=MMC)

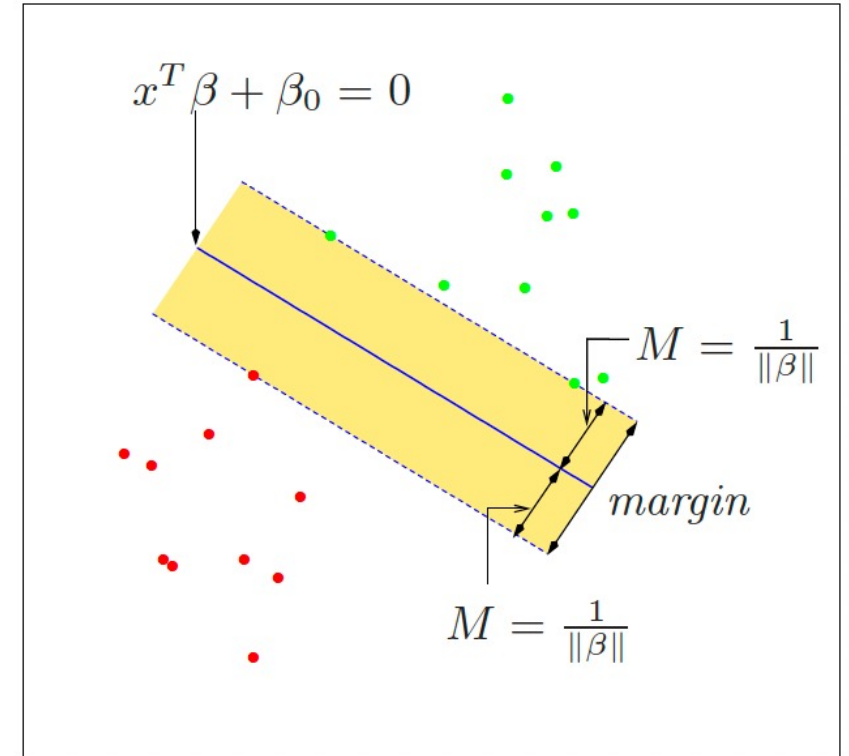
- **Task:** construct the maximal margin hyperplane based on a set of n training observations $x_1, \dots, x_n \in R^p$ and associated class labels $y_1, \dots, y_n \in \{-1, 1\}$.
- Since the **classes are separable**, we can find a function $f(x) = x^T \beta + \beta_0$ with $y_i f(x_i) > 0 \ \forall i$.
- Hence we are able to find the hyperplane that creates the **biggest margin between the training points for class 1 and -1**, the **maximal margin hyperplane** (see Figure)
- This **maximal margin hyperplane** is the **solution to the optimization problem:**

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

subject to $y_i(x_i^T \beta + \beta_0) \geq M, \ i = 1, \dots, n$

This constraint guarantees that each observation will be on the correct side of the hyperplane, provided that M is positive

- The band in Fig. is M units away from the hyperplane on either side, and hence $2M$ units wide. It is called the **margin**.



The 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 Support Vector Classifier - separable case (=MMC)

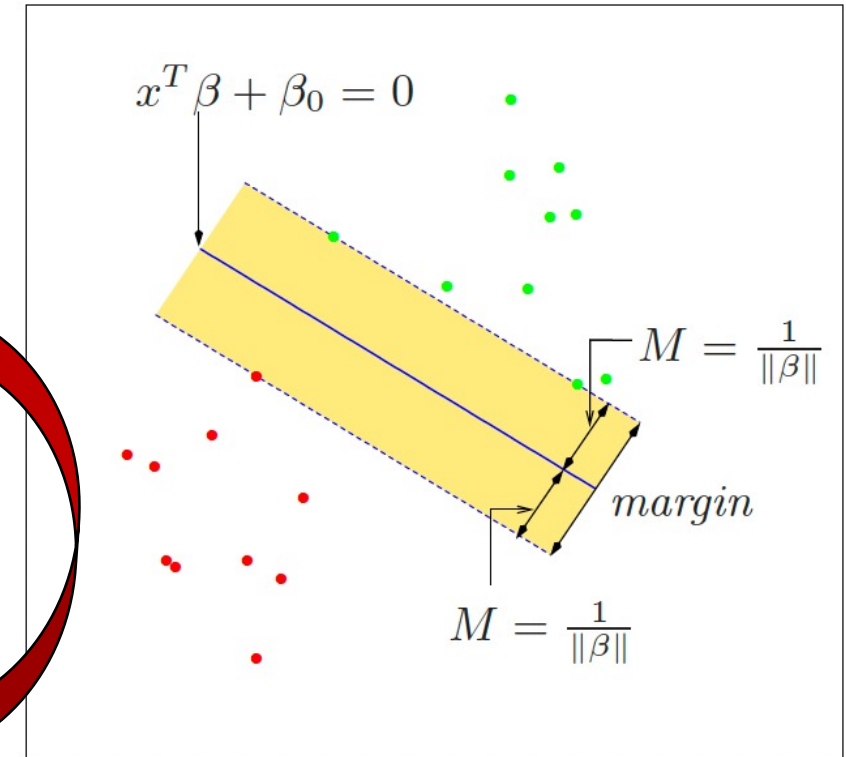
- The previous problem can be more conveniently rephrased as:

$$\begin{aligned} & \min_{\beta, \beta_0} \|\beta\| \\ & \text{subject to } y_i(x_i^T \beta + \beta_0) \geq 1, \quad i = 1, \dots, N, \end{aligned}$$

where we dropped the norm constraint on β .

Note that $M = 1/\|\beta\|$.

- The above expression is the usual way of writing the **support vector criterion for separated data**.
- This is a **convex optimization problem** (quadratic criterion, linear inequality constraints)

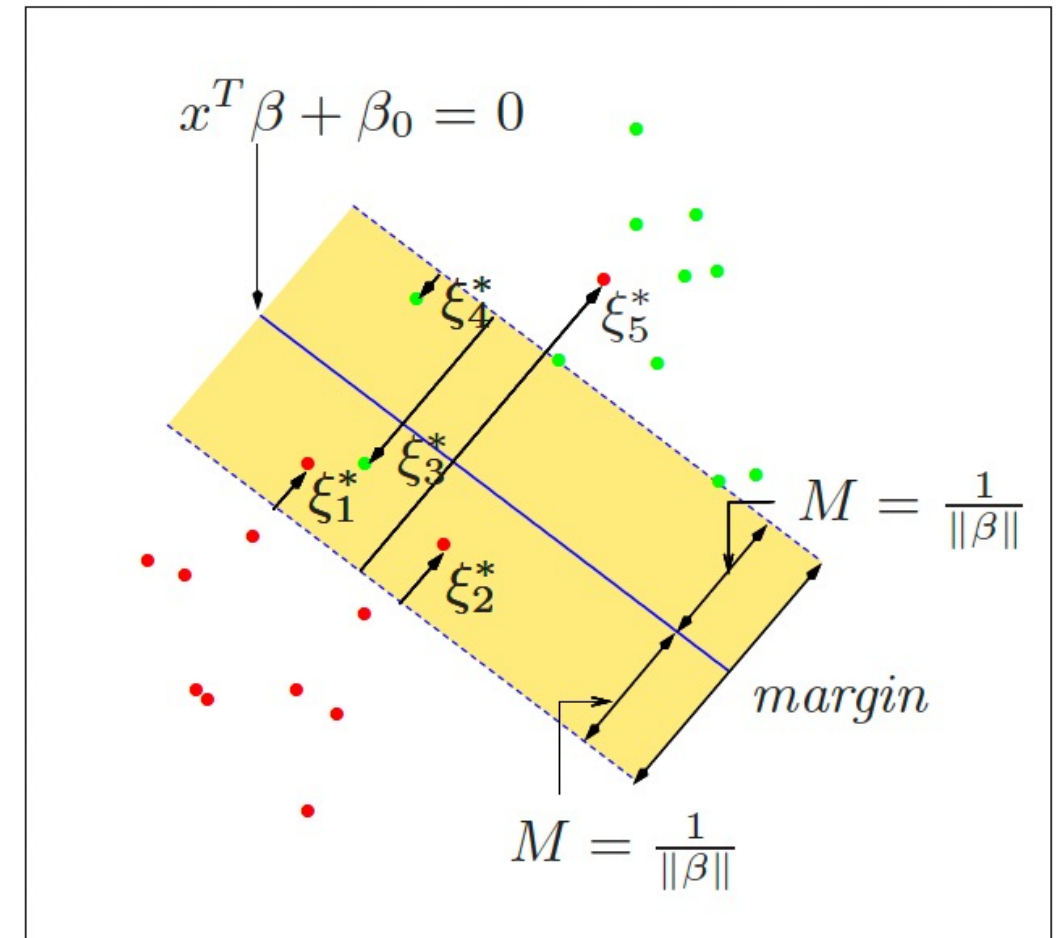


The panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width $2M = 2/\|\beta\|$.

The Support Vector Classifier – nonseparable (overlap) case

- Suppose now that **the classes overlap in feature space**.
- One way to deal with the overlap is to still maximize M , but allow for some points to be on the wrong side of the margin.
- Define the slack variables $\xi = (\xi_1, \xi_2, \dots, \xi_N)$.
- There are **two natural ways to modify the constraint** in the **optimization problema** for the separable case:

$$\begin{aligned} & \max_{\beta, \beta_0, \|\beta\|=1} M \\ & \text{subject to } y_i(x_i^T \beta + \beta_0) \geq M, \quad i = 1, \dots, n \end{aligned}$$



The Support Vector Classifier – nonseparable (overlap) case

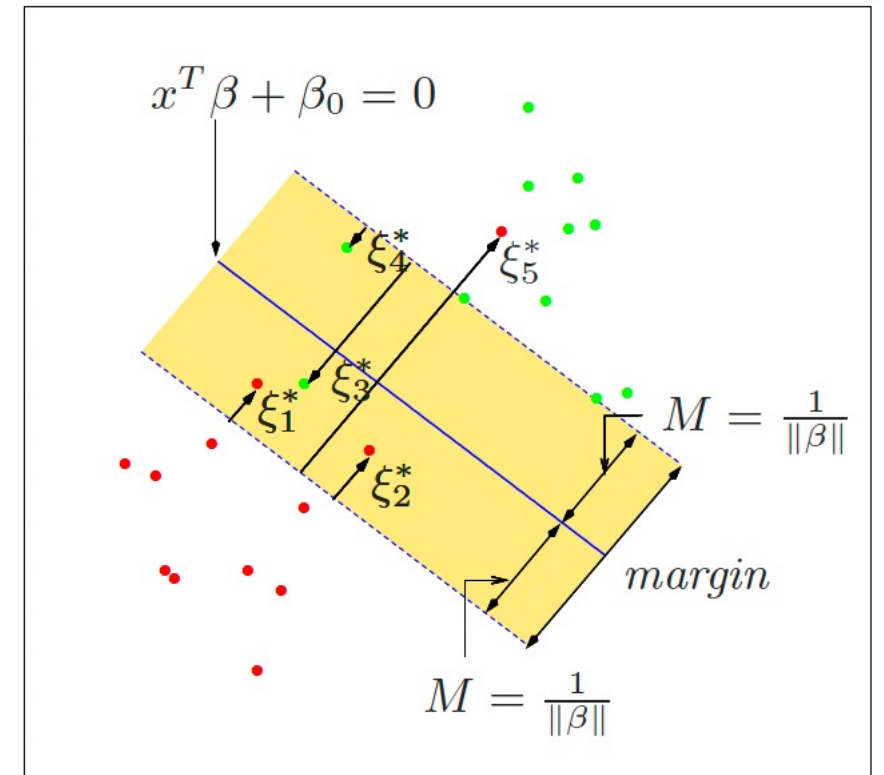
- These **two ways** are

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

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

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

- The two choices lead to different solutions.
 - The **first** seems more natural, since it measures overlap in actual distance from the margin.
 - The **second** choice measures the overlap in relative distance, which changes with the width of the margin M .
 - However, the **first results in a nonconvex optimization problem**, while the **second is convex**, thus leads to the “**standard**” **support vector classifier**, which we use from here on.



The 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 $\sum \xi_i \leq \text{constant}$. Hence $\sum \xi_j$ is the total distance of points on the wrong side of their margin.

The Support Vector Classifier – nonseparable (overlap) case

Idea of the formulation:

- The **value ξ_i in the constraint $y_i(x_i^T \beta + \beta_0) \geq M(1 - \xi_i)$** is the proportional amount by which the prediction $f(x_i) = x_i^T \beta + \beta_0$ is on the wrong side of its margin.
- Hence by **bounding the sum $\sum \xi_i$** , we bound the total proportional amount by which predictions fall on the wrong side of their margin.
- **Misclassifications occur when $\xi_i > 1$** , so bounding $\sum \xi_i$ at a value K say, bounds the total number of training misclassifications at K .

Usual definition for the support vector classifier for the nonseparable case:

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

dropping the norm constraint on β and defining $M = 1/\|\beta\|$.

Computing the Support Vector Classifier

- The problem stated above is quadratic with linear inequality constraints, hence it is a **convex optimization problem**.
- We describe a **quadratic programming solution using Lagrange multipliers**.
- Computationally it is convenient to re-express the problem stated above, for the nonseparable case, in the equivalent form

$$\min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^n \xi_i \quad (**)$$

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

where the “cost” parameter C replaces the constant; the **separable case** corresponds to $C = \infty$.

Computing the Support Vector Classifier

- We minimize the **Lagrange (primal) function** given by

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

with regard to β , β_0 and ξ_i .

- Setting the respective derivatives to zero, we obtain

$$\beta = \sum_{i=1}^n \alpha_i y_i x_i, \quad (1)$$

$$0 = \sum_{i=1}^n \alpha_i y_i, \quad (2)$$

$$\alpha_i = C - \mu_i, \quad \forall i, \quad (3)$$

as well as the positivity constraints $\alpha_i, \mu_i, \xi_i \geq 0 \quad \forall i$.

Computing the Support Vector Classifier

- By substituting (1), (2) and (3) into the **Lagrange (primal) function** we obtain the **Lagrangian (Wolfe) dual objective function**

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

which gives, for any feasible point, a lower bound on the objective function (**)
(slide 28):

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

- We want to

$Max L_D$ subject to: $0 \leq \alpha_i \leq C$

$$\sum_{i=1}^n \alpha_i y_i = 0.$$

Computing the Support Vector Classifier

- The **Karush–Kuhn–Tucker conditions** include the constraints:

$$\beta = \sum_{i=1}^n \alpha_i y_i x_i, \quad (1)$$

$$0 = \sum_{i=1}^n \alpha_i y_i, \quad (2)$$

$$\alpha_i = C - \mu_i, \quad \forall i, \quad (3)$$

$$\alpha_i [y_i (x_i^T \beta + \beta_0) - (1 - \xi_i)] = 0, \quad (4)$$

$$\mu_i \xi_i = 0, \quad (5)$$

$$y_i (x_i^T \beta + \beta_0) - (1 - \xi_i) \geq 0, \quad (6)$$

Computing the Support Vector Classifier

- The **Lagrangian (Wolfe) dual objective function LD** together with the **Karush–Kuhn–Tucker conditions** uniquely characterize the solution to the primal and dual problem.

- From (1), the **solution for β** has the form
$$\hat{\beta} = \sum_{i=1}^n \hat{\alpha}_i y_i x_i,$$

with nonzero coefficients $\hat{\alpha}_i$ only for those **observations i for which the constraints in (6) are exactly met** (due to (4)).

- These observations are called the **support vectors**, since $\hat{\beta}$ is represented in terms of them alone.
- Among these support points, some will lie on the edge of the margin ($\hat{\xi}_i = 0$), and hence from (5) and (2) will be characterized by $0 < \hat{\alpha}_i < C$; the remainder ($\hat{\xi}_i > 0$) have $\hat{\alpha}_i = C$.
- From (4) we can see that any of these margin points ($0 < \hat{\alpha}_i$, $\hat{\xi}_i = 0$) can be used to solve for β_0 , and we typically use an average of all the solutions for numerical stability.

Computing the Support Vector Classifier

- Maximizing the dual

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

is a simpler convex quadratic programming problem than the primal

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

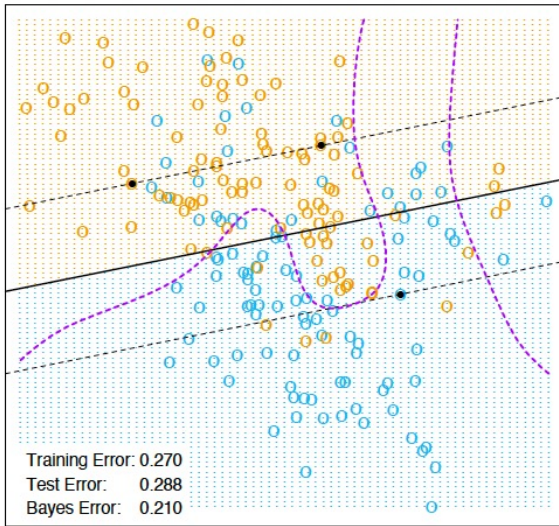
and can be solved with standard techniques (Murray et al., 1981, for example).

- Given the solutions $\hat{\beta}_0$ and $\hat{\beta}$, the **decision function** can be written as

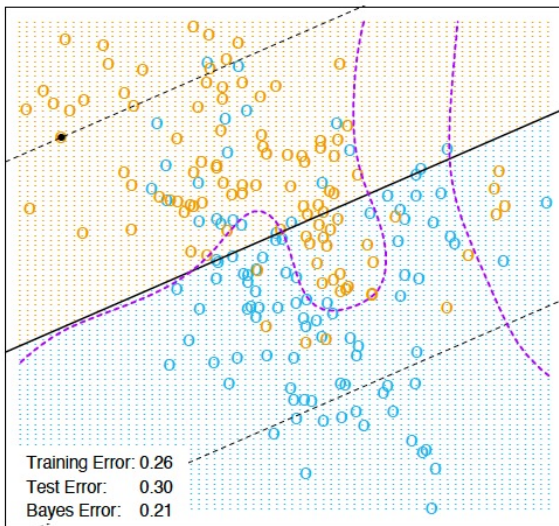
$$\begin{aligned} \hat{G}(x) &= \text{sign}[\hat{f}(x)] \\ &= \text{sign}[x^T \hat{\beta} + \hat{\beta}_0]. \end{aligned}$$

- The **tuning parameter** of this procedure is the **cost parameter** C .

Mixture Example



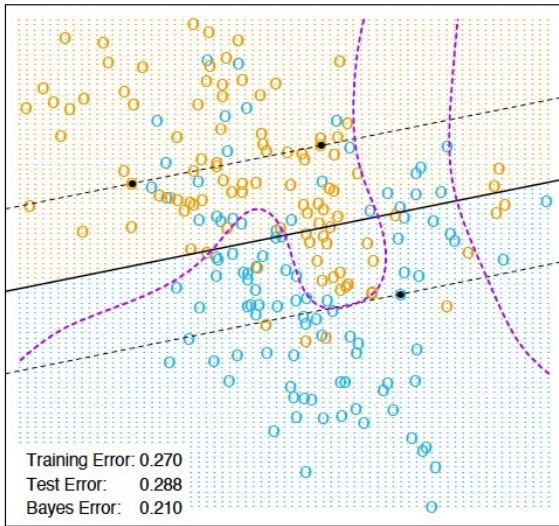
$C = 10000$



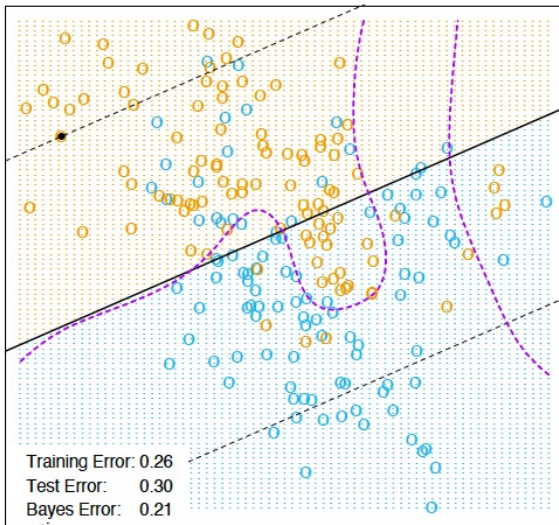
$C = 0.01$

- 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.

Mixture Example



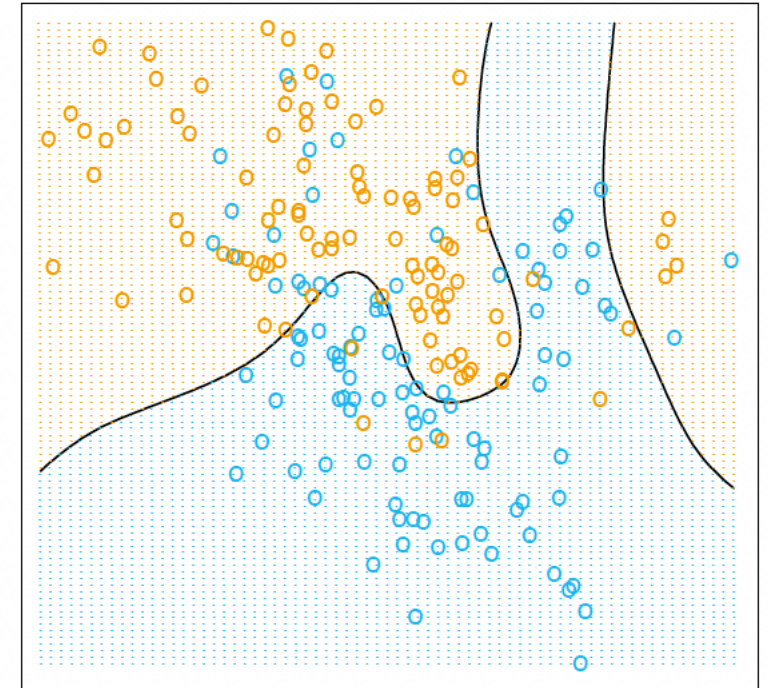
$C = 10000$



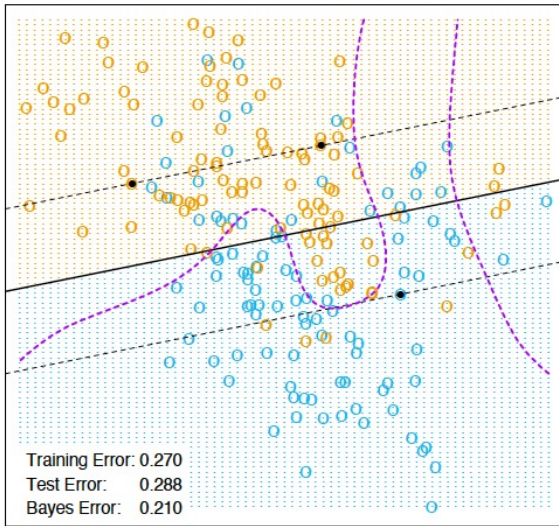
$C = 0.01$

- The Fig. on the left side shows the support vector boundary for the mixture example of Fig. on the right, with two overlapping classes, for two different values of the cost parameter C .
- The classifiers are rather similar in their performance. Points on the wrong side of the boundary are support vectors.
- In addition, points on the correct side of the boundary but close to it (in the margin), are also support vectors.

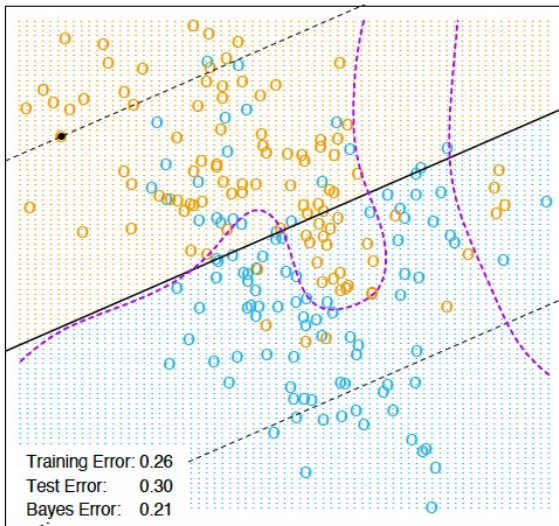
Bayes Optimal Classifier



Mixture Example



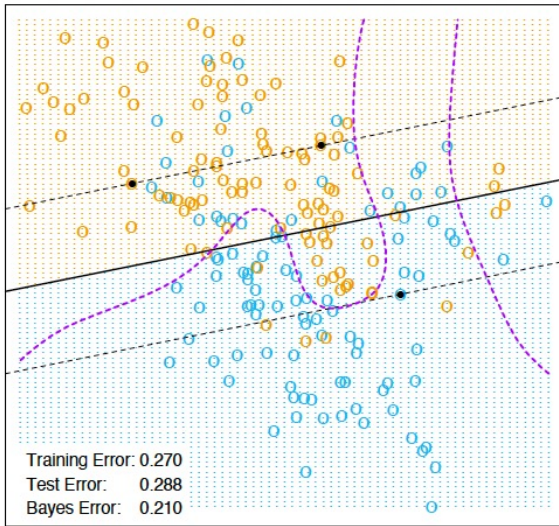
$C = 10000$



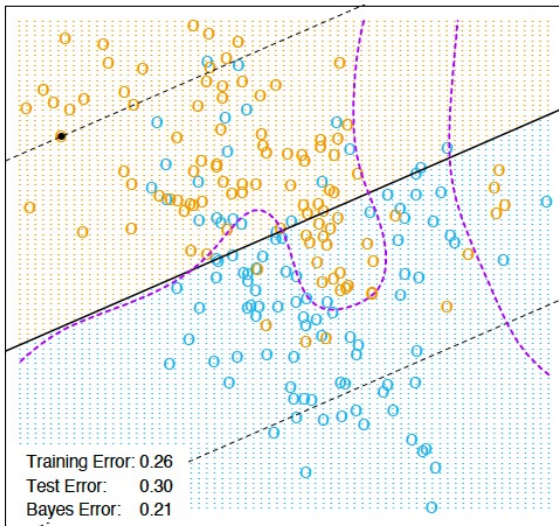
$C = 0.01$

- The margin is larger for $C = 0.01$ than it is for $C = 10,000$.
- Hence larger values of C focus attention more on (correctly classified) points near the decision boundary, while smaller values involve data further away.
- Either way, misclassified points are given weight, no matter how far away.
- In this example, the procedure is not very sensitive to choices of C , because of the rigidity of a linear boundary.

Mixture Example



$C = 10000$



$C = 0.01$

- The optimal value for C can be estimated by cross-validation.
- Interestingly, the leave-one-out cross-validation error can be bounded above by the proportion of support points in the data.
- The reason is that leaving out an observation that is not a support vector will not change the solution.
- Hence these observations, being classified correctly by the original boundary, will be classified correctly in the cross-validation process.
- However this bound tends to be too high, and not generally useful for choosing C (62% and 85%, respectively, in our examples).

Support Vector Machines and Kernels

- The **support vector classifier** described before finds linear boundaries in the input feature space.
- So far, we've only used linear decision boundaries.
- **Such a classifier is likely too restrictive to be useful in practice**, especially when compared to other algorithms that can adapt to nonlinear relationships.
- Fortunately, we can use a simple trick, called the **kernel trick**, to overcome this.
- As with other linear methods, the **procedure can be made more flexible by enlarging the feature space using basis expansions such as polynomials or splines.**
- Generally **linear boundaries in the enlarged space achieve better training-class separation, and translate to nonlinear boundaries in the original space.**
- We are going to show an example with a couple illustrations in 2-D/3-D feature space to drive home the key idea.

Support Vector Machines and Kernels

Example:

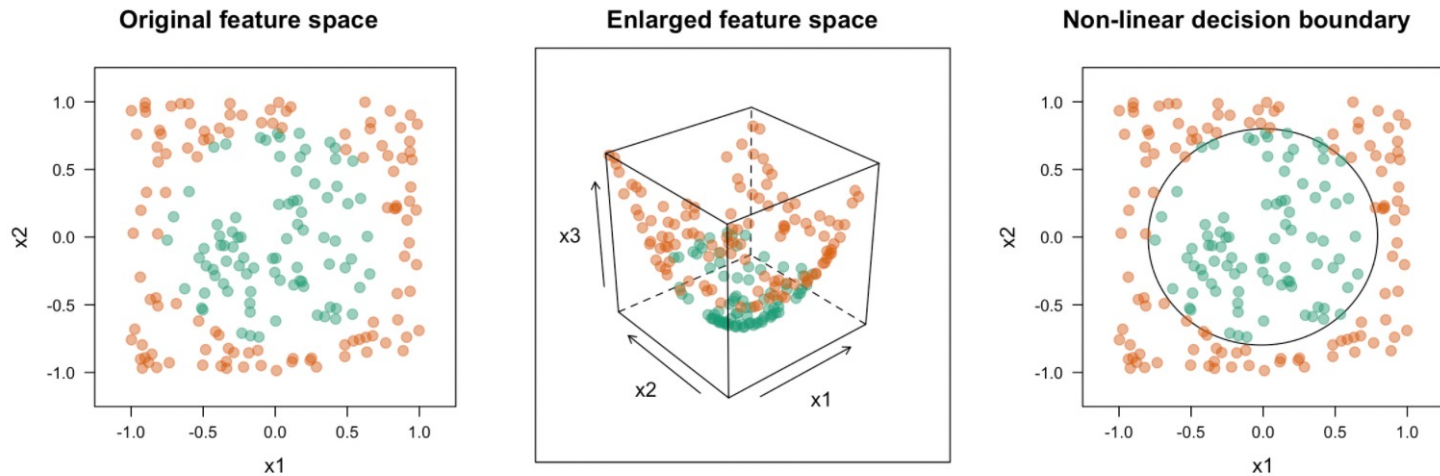


Figure : Simulated nested circle data. *Left:* The two classes in the original (2-D) feature space. *Middle:* The two classes in the enlarged (3-D) feature space. *Right:* The decision boundary from the HMC in the enlarged feature space projected back into the original feature space.

- Consider the circle data on the left side of Fig.
- This is a **binary classification problem**.
- The **first class forms a circle** in the middle of a square, the **remaining points form the second class**.
- **Although these two classes do not overlap** (although they appear to overlap slightly due to the size of the plotted points), **they are not perfectly separable by a hyperplane** (i.e., a straight line).

Support Vector Machines and Kernels

Example:

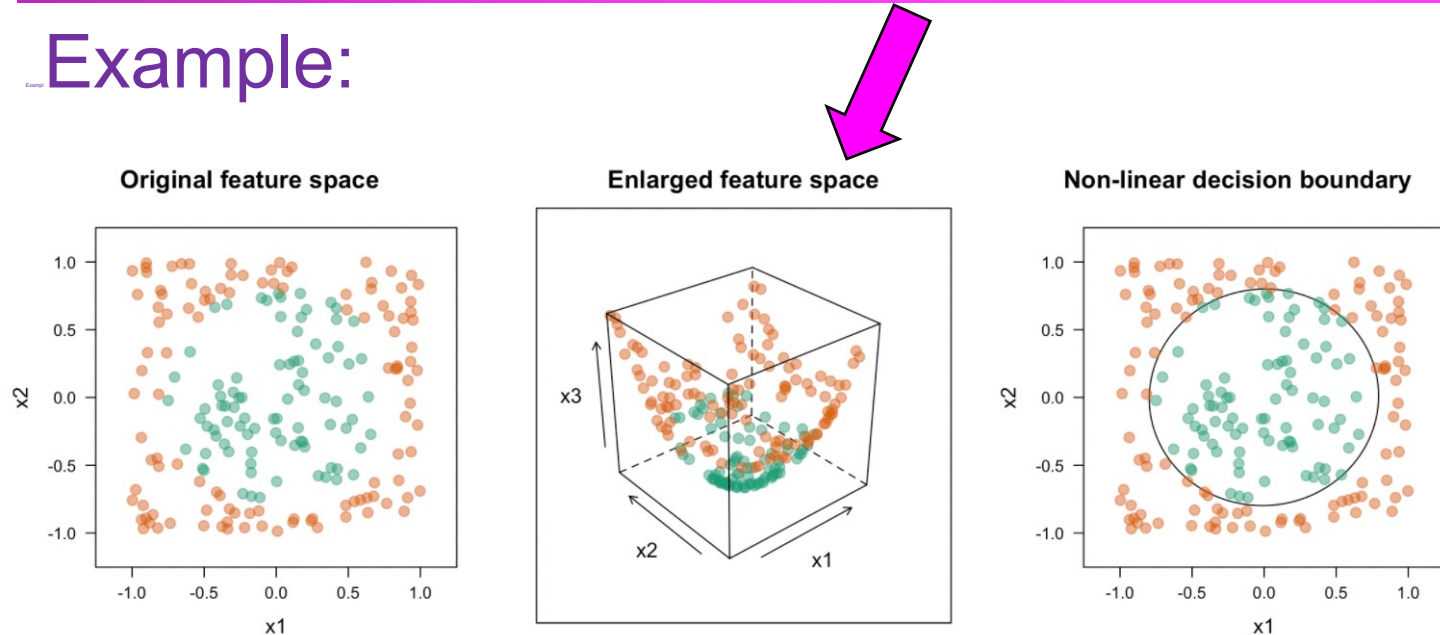


Figure : Simulated nested circle data. *Left:* The two classes in the original (2-D) feature space. *Middle:* The two classes in the enlarged (3-D) feature space. *Right:* The decision boundary from the HMC in the enlarged feature space projected back into the original feature space.

- However, we can **enlarge the feature space by adding a third feature**, say $X_3 = X_1^2 + X_2^2$ - this is akin to using the polynomial kernel function with $d=2$.
- The data are plotted in the enlarged feature space in the middle of Fig.
- **In this new three dimensional feature space, the two classes are perfectly separable by a hyperplane** (i.e., a flat plane); though it is hard to see (see the middle of Fig.), the green points form the tip of the hyperboloid in 3-D feature space (i.e., X_3 is smaller for all the green points leaving a small gap between the two classes).

Support Vector Machines and Kernels

Example:

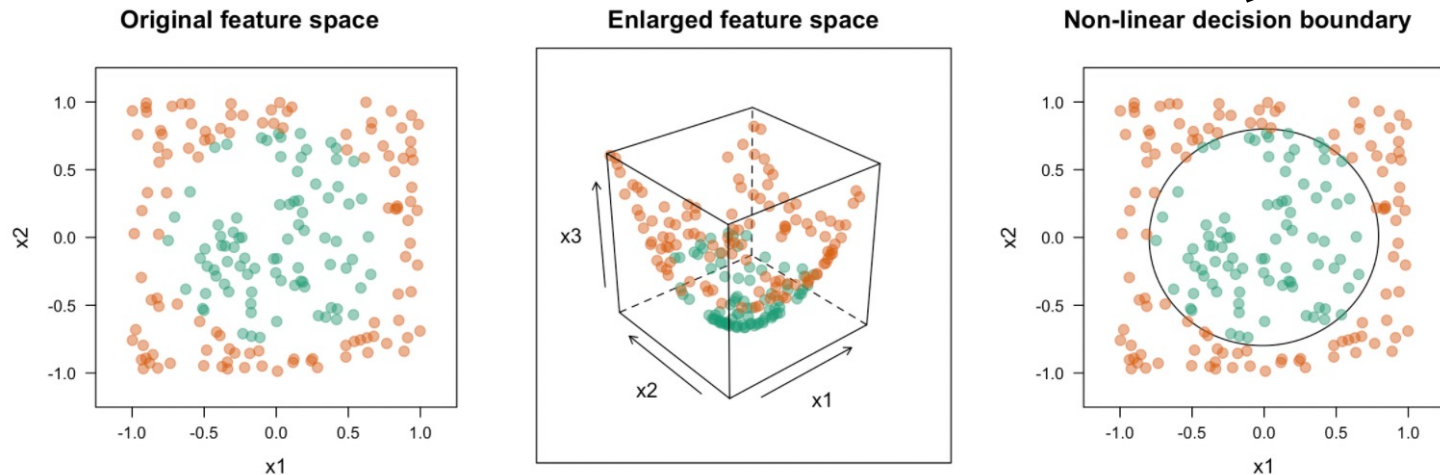


Figure : Simulated nested circle data. *Left:* The two classes in the original (2-D) feature space. *Middle:* The two classes in the enlarged (3-D) feature space. *Right:* The decision boundary from the HMC in the enlarged feature space projected back into the original feature space.

- The resulting decision boundary is then projected back onto the original feature space resulting in a non-linear decision boundary which perfectly separates the original data (see the right side of Fig.)

- In essence, **SVM use the kernel trick to enlarge the feature space using basis functions** (e.g., like polynomial regression).
- **In this enlarged** (kernel-induced) **feature space, a hyperplane can often separate the two classes.**
- **The resulting decision boundary, which is linear in the enlarged feature space, will be nonlinear when transformed back onto the original feature space.**

Support Vector Machines and Kernels

- Once the basis functions $h_m(x)$, $m = 1, \dots, M$ are selected, the procedure is the same as before.
- We fit the **support vector classifier** using input features $h(x_i) = (h_1(x_i), h_2(x_i), \dots, h_M(x_i))$, $i = 1, \dots, n$, and produce the (nonlinear) function $\hat{f}(x) = h(x)^T \hat{\beta} + \hat{\beta}_0$.
- The **classifier** is $\hat{G}(x) = \text{sign}(\hat{f}(x))$ as before.

Support Vector Machines and Kernels

- The **support vector machine classifier** is an **extension of this idea**, where the **dimension of the enlarged space is allowed to get very large, infinite in some cases**.
- It might seem that
 - the computations would become prohibitive;
 - with sufficient basis functions, the data would be separable, and overfitting would occur.
- It is explained
 - how the SVM technology deals with these issues;
 - the SVM classifier is solving a function-fitting problem using a particular criterion and form of regularization, and is part of a much bigger class of problems that includes the smoothing splines.

Computing the SVM for Classification

- The optimization problem, i.e., the **Lagrange (primal) function**

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

can be represented and well as its solution in a special way that only involves the input features via inner products.

- This is done directly for the transformed feature vectors $h(x_i)$.
- For particular choices of h , these inner products can be computed very cheaply.
- The **Lagrange dual function** is

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

Computing the SVM for Classification

- The **Lagrange dual function**

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

has the form

$$L_D = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{i'=1}^n \alpha_i \alpha_{i'} y_i y_{i'} \langle h(x_i), h(x_{i'}) \rangle. \quad (\text{a})$$

- From $\beta = \sum_{i=1}^n \alpha_i y_i x_i$, the solution function $f(x)$ can be written

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

- As before, given α_i, β_0 can be determined by solving $y_i f(x_i) = 1$ in (b) for any (or all) x_i for which $0 < \alpha_i < C$.

Computing the SVM for Classification

- So both (a) and (b) involve $h(x)$ only through inner products.
- In fact, we need not specify the transformation $h(x)$ at all, but require only knowledge of the kernel function that computes inner products in the transformed space

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

K should be a symmetric positive (semi-) definite function

- **Three popular choices for K** in the SVM literature

*d*th-Degree polynomial: $K(x, x') = (1 + \langle x, x' \rangle)^d,$

Radial basis: $K(x, x') = \exp(-\gamma \|x - x'\|^2),$

Neural network: $K(x, x') = \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2).$

Computing the SVM for Classification

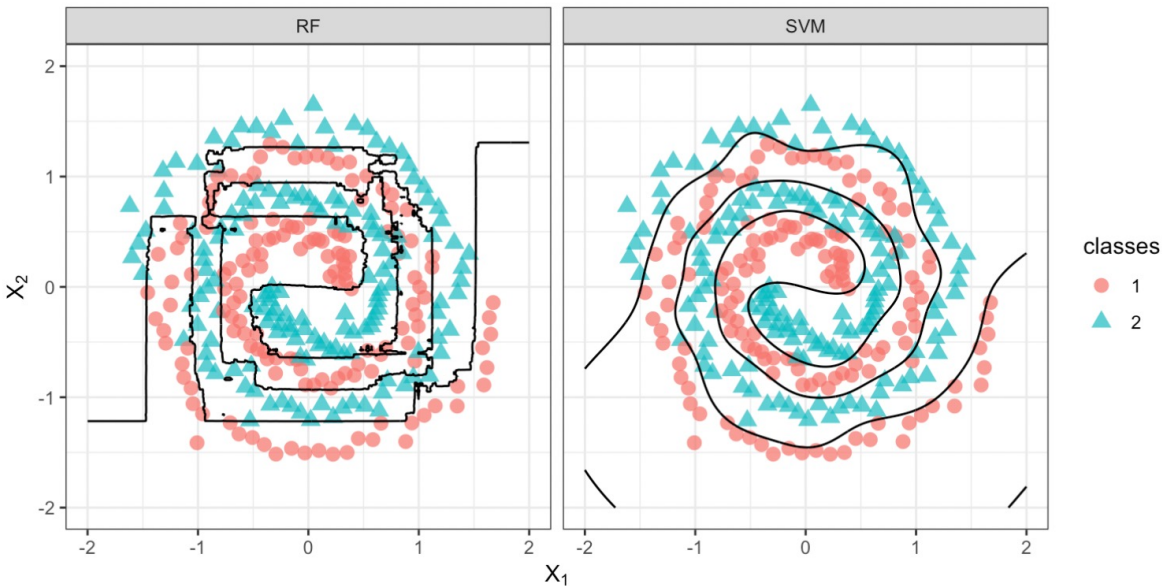


Figure 1: Two spirals benchmark problem. *Left:* Decision boundary from a random forest. *Right:* Decision boundary from an SVM with radial basis kernel.

- Through the use of various kernel functions, SVM are extremely flexible and capable of estimating complex nonlinear decision boundaries.
- For example, the right side of Fig. demonstrates the flexibility of an SVM using a radial basis kernel applied to the two spirals benchmark problem.
- As a reference, the left side of Fig. shows the decision boundary from a default random forest fit using the **ranger** package.
- The random forest decision boundary, while flexible, has trouble capturing smooth decision boundaries (like a spiral).
- The SVM with a radial basis kernel, on the other hand, does a great job (and in this case is more accurate).
- The radial basis kernel is extremely flexible and as a rule of thumb, we generally start with this kernel when fitting SVM in practice.

Computing the SVM for Classification

Example:

- Suppose a feature space with two inputs X_1 and X_2 . Then considering two points $X=(X_1, X_2)$ and $X'=(X_1', X_2')$, the kernel function as a polynomial kernel of degree 2 over this points is

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

- Then, considering

$$h(X) = (h_1(X), h_2(X), h_3(X), h_4(X), h_5(X), h_6(X)) \quad (M = 6),$$

being

$$h_1(X) = 1, h_2(X) = \sqrt{2}X_1, h_3(X) = \sqrt{2}X_2, h_4(X) = X_1^2, h_5(X) = X_2^2, h_6(X) = \sqrt{2} X_1 X_2,$$

then

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

Computing the SVM for Classification

Example:

- In fact

$$\begin{aligned} K(X, X') &= \langle h(X), h(X') \rangle = \\ &= \langle (h_1(X), h_2(X), h_3(X), h_4(X), h_5(X), h_6(X)), (h_1(X'), h_2(X'), h_3(X'), h_4(X'), h_5(X'), h_6(X')) \rangle = \\ &= h_1(X) h_1(X') + h_2(X) h_2(X') + h_3(X) h_3(X') + h_4(X) h_4(X') + h_5(X) h_5(X') + h_6(X) h_6(X') = \\ &= 1 \times 1 + \sqrt{2} X_1 \sqrt{2} X_1' + \sqrt{2} X_2 \sqrt{2} X_2' + X_1^2 X_1'^2 + X_2^2 X_2'^2 + \sqrt{2} X_1 X_2 \sqrt{2} X_1' X_2' = \\ &= 1 + 2 X_1 X_1' + 2 X_2 X_2' + X_1^2 X_1'^2 + X_2^2 X_2'^2 + 2 X_1 X_2 X_1' X_2' \\ &= (1 + \langle X, X' \rangle)^2 \end{aligned}$$

- From $f(x) = \sum_{i=1}^n \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0$ the **solution** can be written

$$\hat{f}(x) = \sum_{i=1}^n \hat{\alpha}_i y_i \langle h(x), h(x_i) \rangle + \hat{\beta}_0 = \sum_{i=1}^n \hat{\alpha}_i y_i K(x, x_i) + \hat{\beta}_0.$$

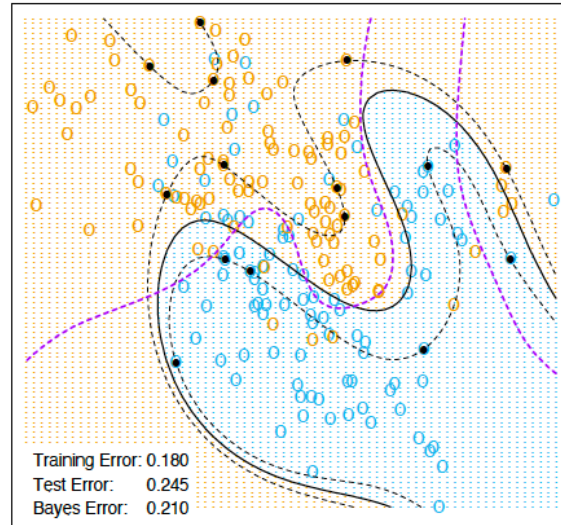
- The **classifier** is $\hat{G}(x) = \text{sign}(\hat{f}(x))$

Computing the SVM for Classification

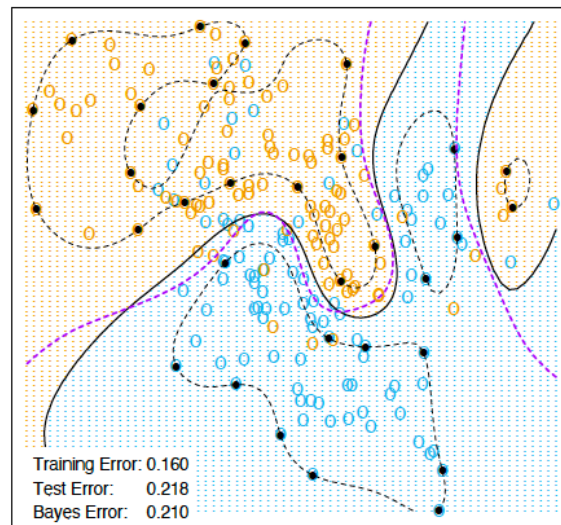
Figure Two nonlinear SVM for the mixture data.

- 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.

SVM - Degree-4 Polynomial in Feature Space



SVM - Radial Kernel in Feature Space

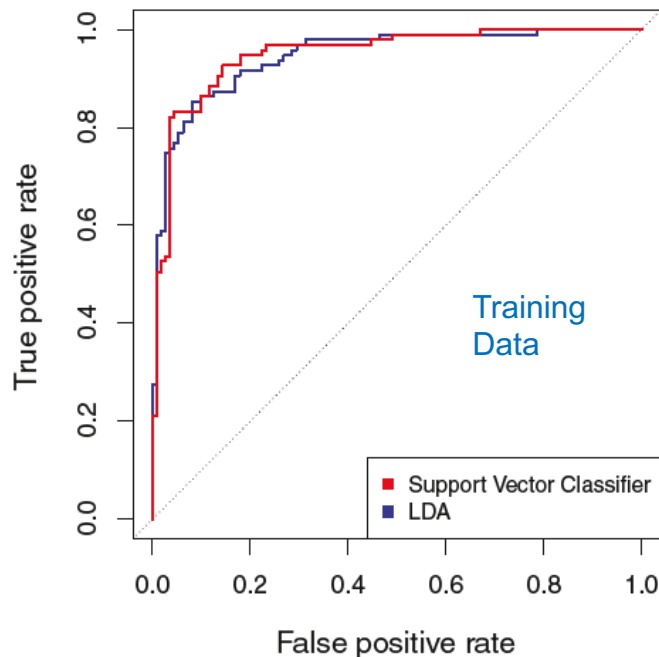


- The role of the parameter C is clearer in an enlarged feature space, since perfect separation is often achievable there.
- A **large value of C** will discourage any positive ξ_i , and lead to an overfit wiggly boundary in the original feature space; a **small value of C** will encourage a small value of $\|\beta\|$, which in turn causes $f(x)$ and hence the boundary to be smoother.
- Fig. show two nonlinear support vector machines applied to the mixture example.
- The regularization parameter was chosen in both cases to achieve good test error.
- **The radial basis kernel produces a boundary quite similar to the Bayes optimal boundary for this example** (purple dashed line);

SVM - An Application to the Heart Disease Data

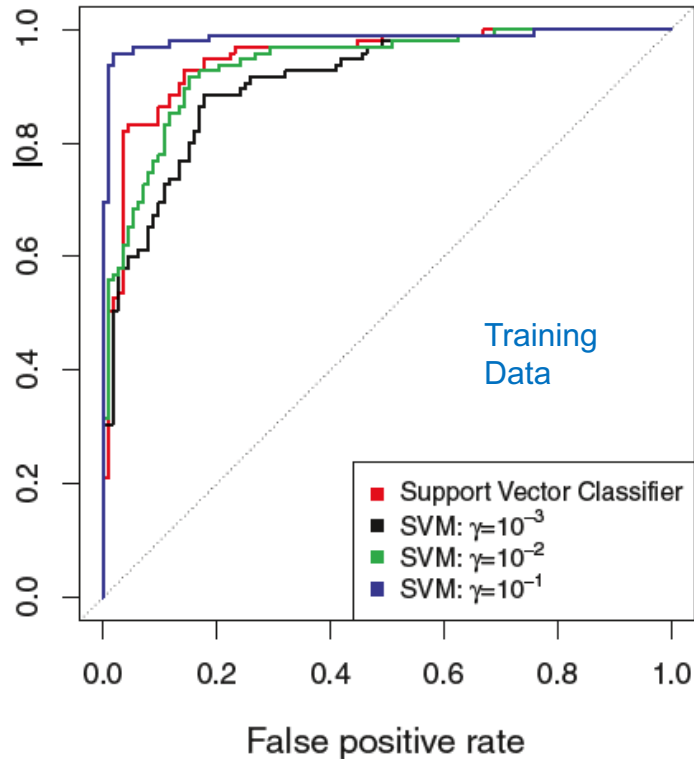
- **Goal:** investigate how an SVM compares to LDA on this data.
- The **data** consist of 297 subjects, which we randomly split into 207 training and 90 test observations.
- First, fit LDA and the SVM classifier to the training data.

The support vector classifier is equivalent to a SVM using a polynomial kernel of degree $d = 1$.



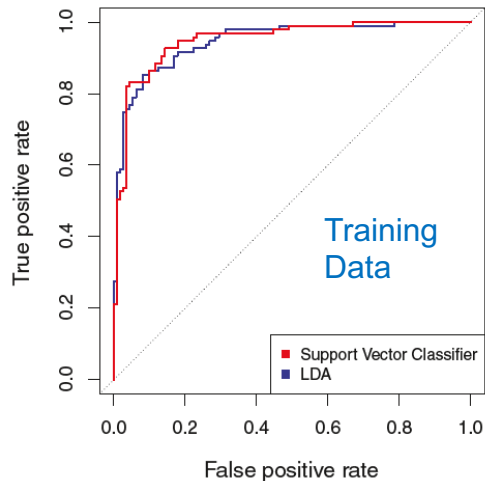
- Fig. displays ROC curves for the training set predictions for both LDA and the SVM classifier.
- Both classifiers compute scores of the form $\hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \dots + \hat{\beta}_p X_p$ for each observation.
- For any given cutoff t , we classify observations into the heart disease or no heart disease categories depending on whether $\hat{f}(x) < t$ or $\hat{f}(x) \geq t$.
- The ROC curve is obtained by forming these predictions and computing the false positive and true positive rates for a range of values of t .
- An optimal classifier will hug the top left corner of the ROC plot.
- In this instance, **LDA and the SVM classifier both perform well**, though there is a suggestion that the **SVM classifier may be slightly superior**.

SVM - An Application to the Heart Disease Data

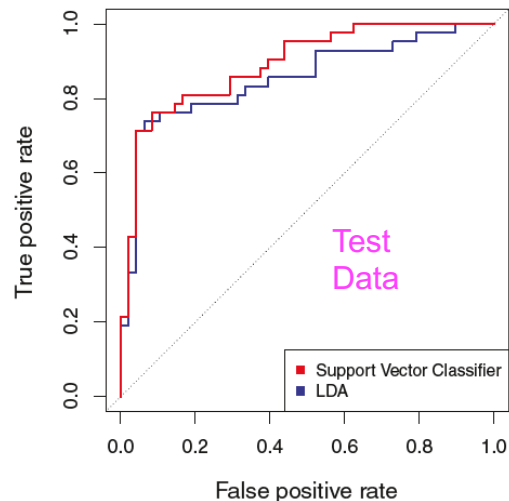


- Fig. displays **ROC curves for SVM using a radial kernel, with various values of γ .**
- **As γ increases and the fit becomes more non-linear, the ROC curves improve.**
- Using $\gamma = 10^{-1}$ appears to give an almost perfect ROC curve.
- **However, these curves represent training error rates, which can be misleading in terms of performance on new test data.**

SVM - An Application to the Heart Disease Data

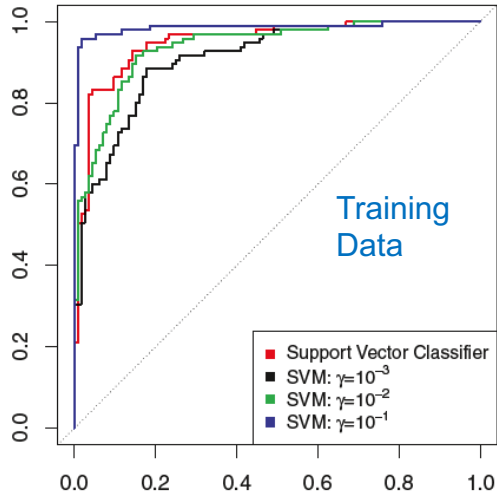


- ROC curves computed on the 90 test observations have some differences from the training ROC curves, computed on the 207 training observations.

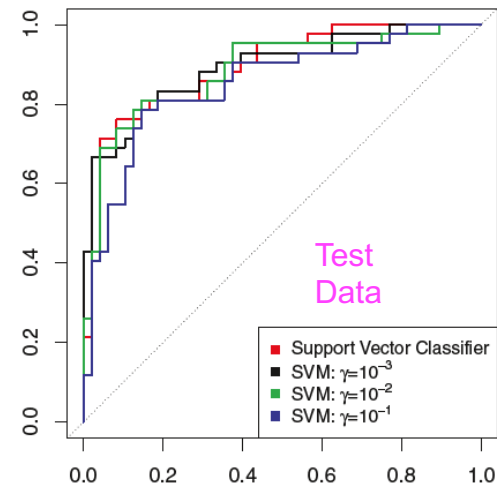


- Observing test ROC curves, the **support vector classifier appears to have a small advantage over LDA** (although these differences are not statistically significant).

SVM - An Application to the Heart Disease Data



False positive rate



False positive rate

- The SVM using $\gamma = 10^{-1}$, which showed the **best results** on the **training data**, produces the **worst estimates** on the **test data**.
- This is once again evidence that **while a more flexible method will often produce lower training error rates, this does not necessarily lead to improved performance on test data**.
- In **test data**, the SVM with $\gamma = 10^{-2}$ and $\gamma = 10^{-3}$ perform comparably to the support vector classifier, and all three outperform the SVM with $\gamma = 10^{-1}$.

SVM with More than Two Classes ($K > 2$)

- So far, our discussion has been limited to the case of **binary classification**: that is, classification in the two-class setting.
- How can we extend SVM to the more general case where we have some arbitrary number of classes?
- The support vector machine can be extended to multiclass problems, **essentially by solving many two-class problems**.
- Though a **number of proposals for extending SVM to the K-class case** have been made, the two **most popular approaches** are:
 - **one-versus-one**;
 - **one-versus-all**.

SVM with More than Two Classes - **One-versus-one**

- Suppose that we would like to perform classification using SVM, and there are $K > 2$ classes.
- A **one-versus-one** or **all-pairs** approach constructs $\binom{K}{2}$ SVM, each of which compares a pair of classes.
- For example, one such SVM might compare the k^{th} class, coded as $+1$, to the k'^{th} class, coded as -1 .
- We classify a test observation using each of the $\binom{K}{2}$ classifiers, and we count the number of times that the test observation is assigned to each of the K classes.
- The **final classification** is performed by **assigning the test observation to the class to which it was most frequently assigned in these $\binom{K}{2}$ pairwise classifications.**
- So in this approach, a classifier is built for each pair of classes, and the final classifier is the one that dominates the most.

SVM with More than Two Classes - One-versus-all

- The **one-versus-all** approach is an alternative procedure for applying SVM in the case of $K > 2$ classes.
- We fit K SVM, each time comparing one of the K classes to the remaining $K-1$ classes.
- Let $\beta_{0k}, \beta_{1k}, \dots, \beta_{pk}$ denote the parameters that result from fitting an SVM comparing the k^{th} class (coded as $+1$) to the others (coded as -1).
- Let \mathbf{x}^* denote a test observation.
- The classification is performed by assign the observation to the class for which $\beta_{0k} + \beta_{1k}x_1^* + \beta_{2k}x_2^* + \dots + \beta_{pk}x_p^*$ is largest, as this amounts to a high level of confidence that the test observation belongs to the k^{th} class rather than to any of the other classes.