Chapter 9 Support Vector Machines

Here we approach the two-class classification problem in a direct way:

$$We\ try\ and\ find\ a\ plane\ that\ separates\ the\ classes\ in\ feature\ space.$$

If we can't , we get creative in two ways:

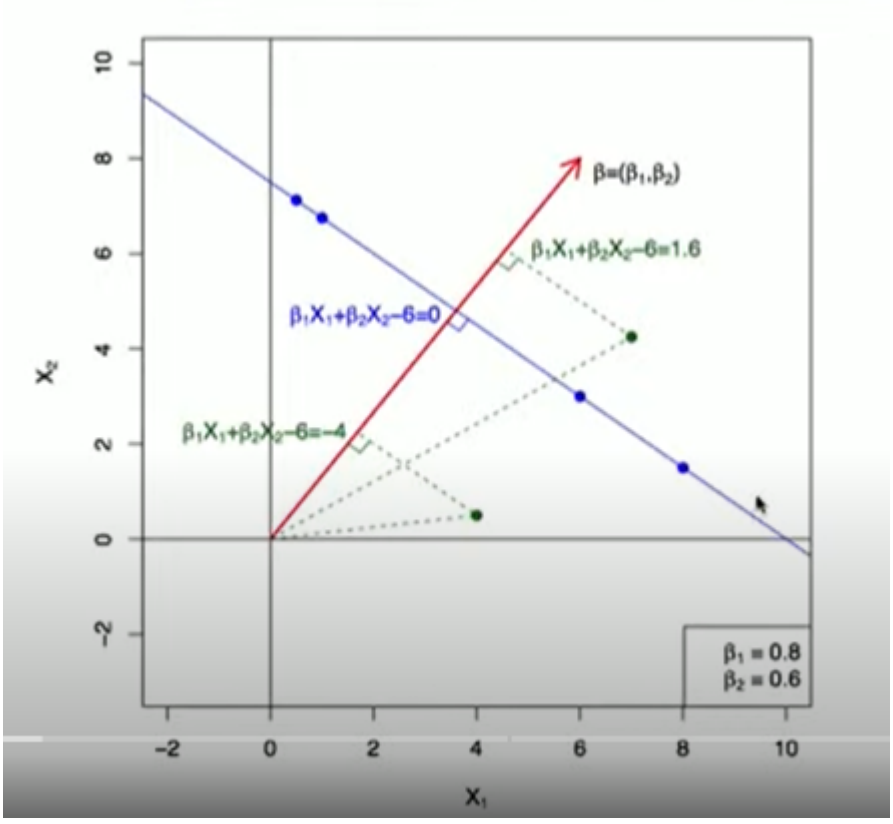
- We soften what we mean by "separates" and,
- We enrich and enlarge the feature space so that separation is possible.

$What\ is\ hyperplane?$

- A hyperplane in p dimensions is a flat affine subspace of dimension (p-1).
- In general the equation for a hyperplane has the form:

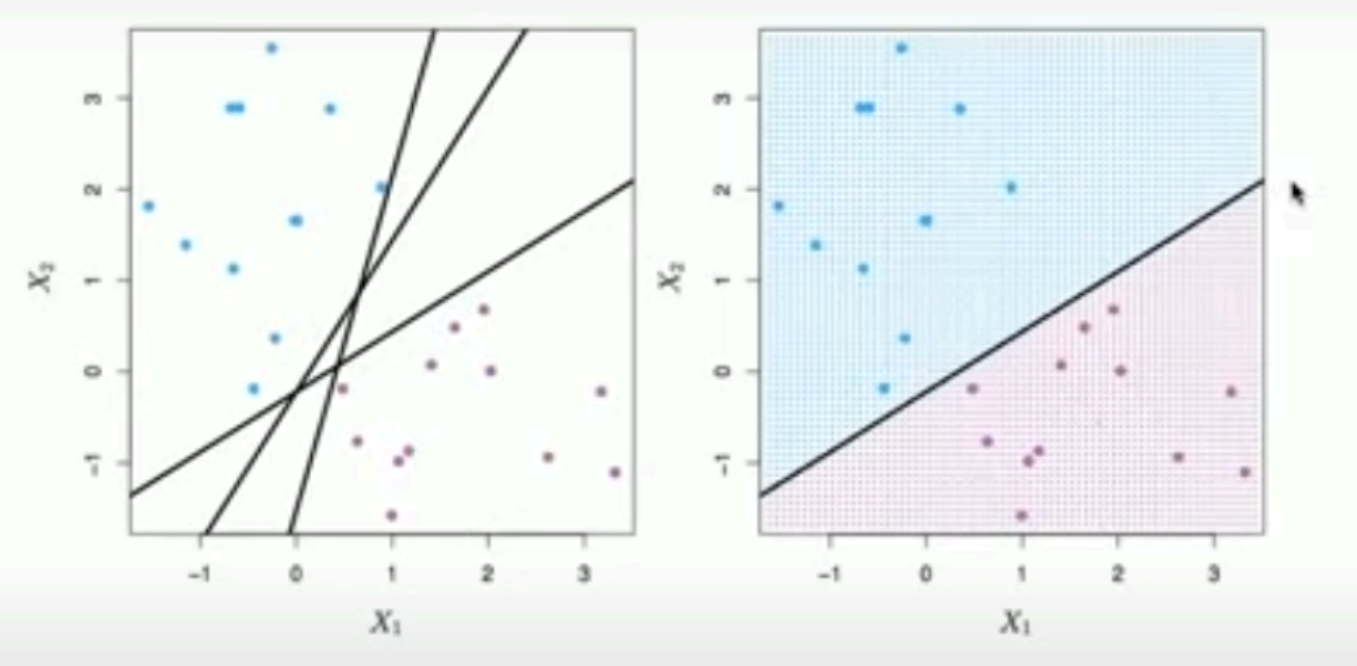
$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = 0$$

- In $p = 2$ dimensions a hyperplane is a line.
- If $\beta_0 = 0$, the hyperplane goes through the origin, otherwise not.
- The vector $\beta = (\beta_0, \beta_1, \ldots, \beta_p)$ is called the **normal vector** - it points in a direction orthogonal to the surface of a hyperplane.



- Above is a hyperplane in 2 dimensions, blue in color. The red vector is the normal vector.

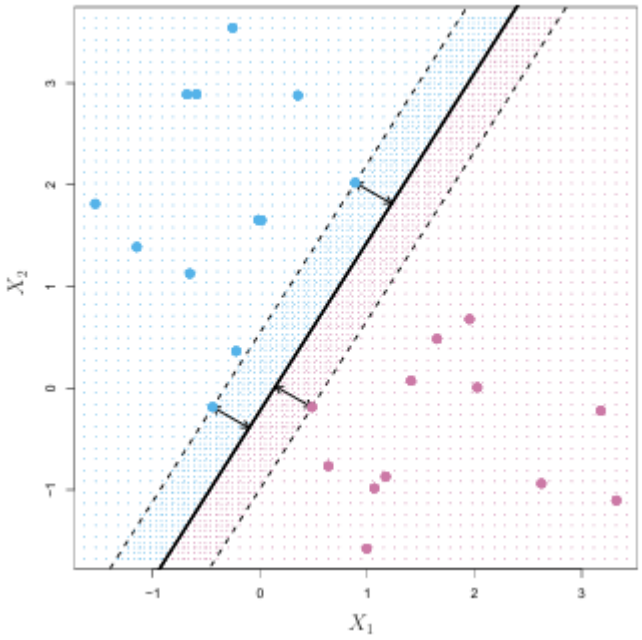
$Separating\ Hyperplanes$:



- If $f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$, then $f(X) > 0$ for points on one side of the hyperplane, and $f(X) < 0$ for points on the other.
- If we code the colored points as $Y_i = +1$ for blue and say , $Y_i = -1$ for mauve, then if $Y_i \cdot f(X_i) > 0$ for all i ,$f(X) = 0$ defines a **separating hyperplane.**

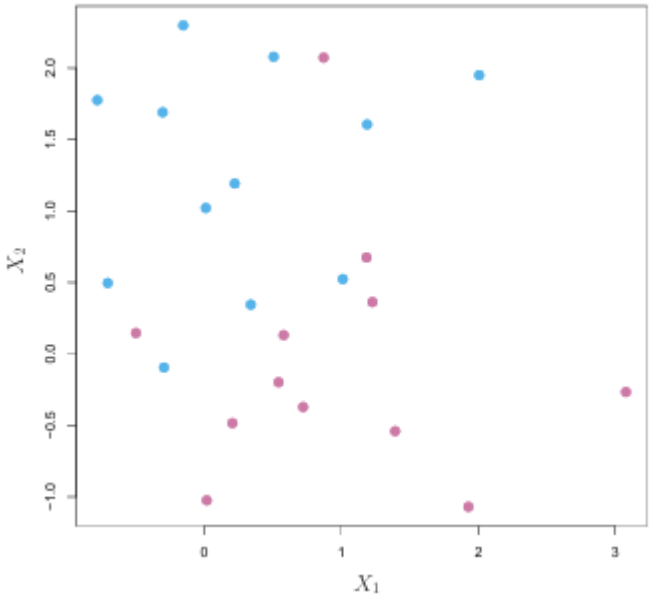
$Maximal\ Margin\ Classifier$

Among all separating hyperplanes, find the one that makes the biggest gap or margin between the two classes.



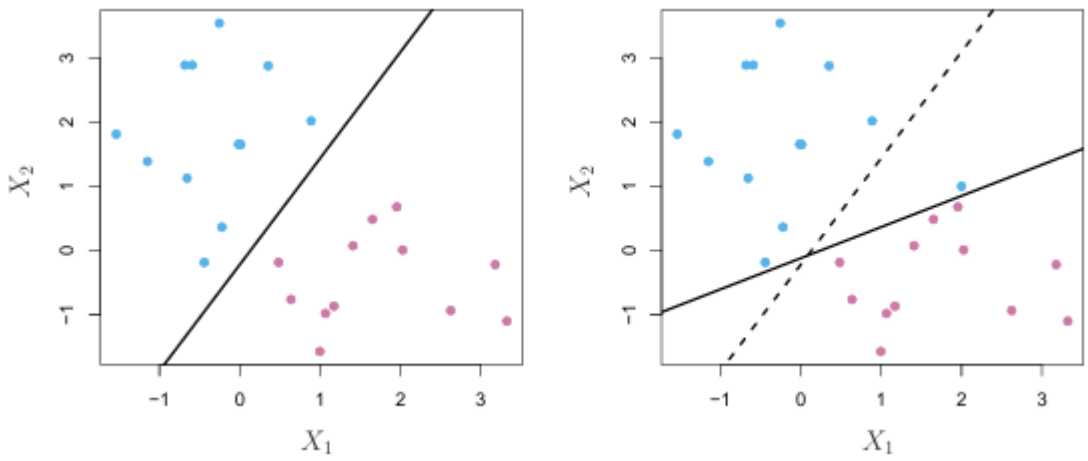
Constrained Optimization problem :-
$\max_{\beta_0,\beta_1,\ldots,\beta_p} M$ subject to $\sum_{j=1}^{p} \beta_j^2 = 1$, $y_i \cdot (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq M$ for all $i = 1, 2, \ldots, N$

Non Separable Data

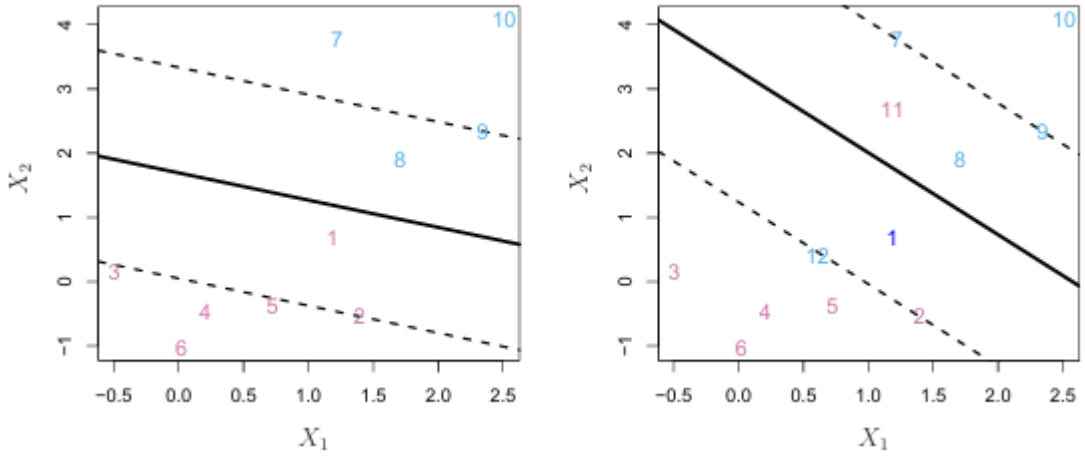


There are two classes of observations, shown in blue and in purple. In this case, the two classes are not separable by a hyperplane, and so the maximal margin classifier cannot be used.
This is often the case unless $N < p$, in cases like genomics. Here you can always separate the data with a hyperplane.

Noisy Data



Sometimes data are separable but noisy. This can lead to a poor solution for the maximal-margin classifier.
The support vector classifier maximizes a $soft$ margin.



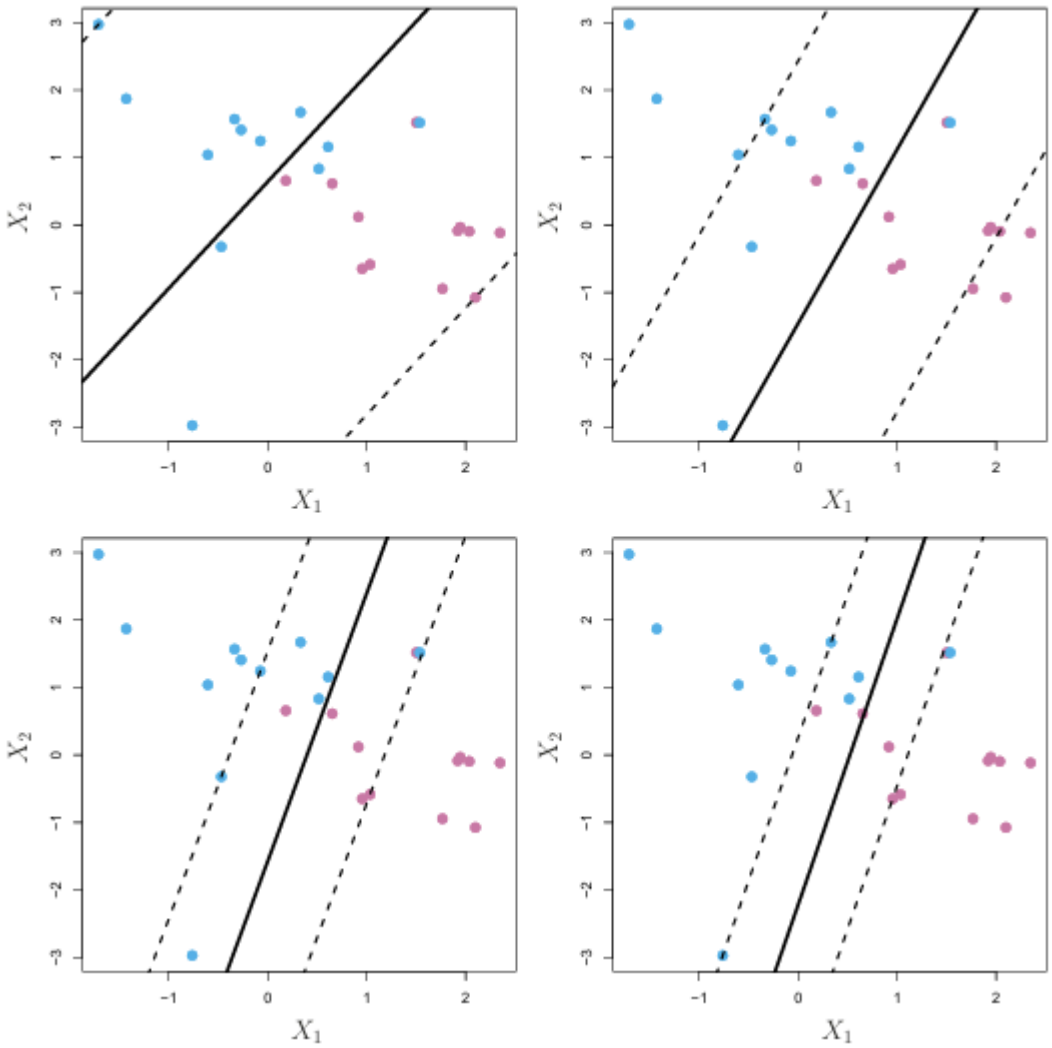
- Left: A support vector classifier was fit to a small data set. The hyperplane is shown as a solid line and the margins are shown as dashed lines.
- Purple observations: Observations 3,4,5, and 6 are on the correct side of the margin, observation 2 is on the margin, and observation 1 is on the wrong side of the margin.
- Blue observations: Observations 7 and 10 are on the correct side of the margin, observation 9 is on the margin, and observation 8 is on the wrong side of the margin.
- No observations are on the wrong side of the hyperplane.
- Right: Same as left panel with two additional points, 11 and 12. These two observations are on the wrong side of the hyperplane and the wrong side of the margin.

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C,$$

- We are going to allow some slack to the margin M, which is to be controlled by the budget C , which is a tuning parameter.

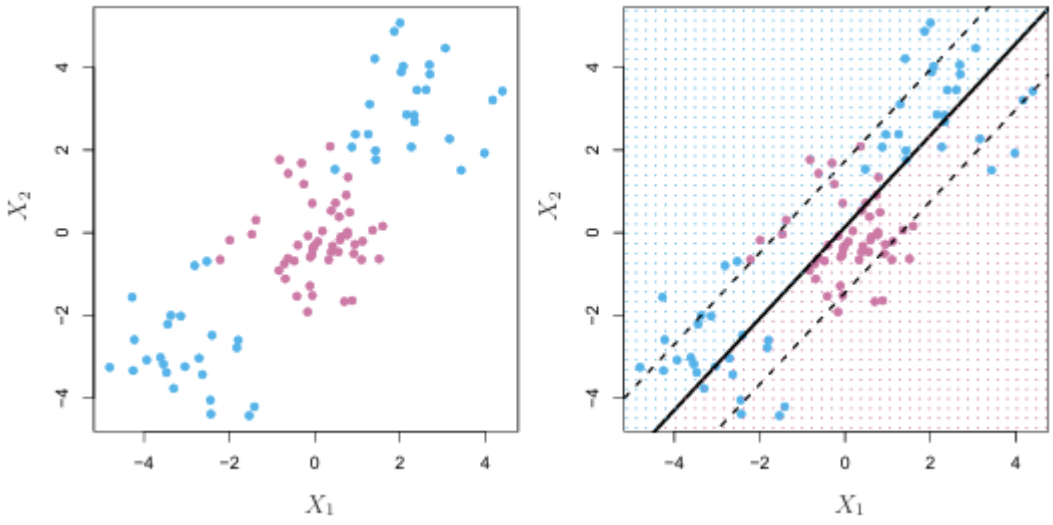


- For the above figure, a support vector classifier was fit using four different values of the tuning parameter C .
- The largest value of C was used in the top left panel, and smaller values were used in the top right, bottom left, and bottom right panels.
- When C is large, then there is a high tolerance for observations being on the wrong side of the margin, and so the margin will be large.
- As C decreases, the tolerance for observations being on the wrong side of the margin decreases, and the margin narrows.

$Note$:

- We see later that the number of points that are on the wrong side of the margin, in other words all points inside the margin become the points that are controlling the orientation of the margin. So, in some sense the more points that are involved in the orientation of the margin , the more stable it becomes. And that means as C gets bigger , the more stable the margin becomes. We see a bias-variance trade-off as we change C .So, it's really a regularization parameter.
- Also like Lasso and ridge regression, variables should be standardized here. We are taking Euclidean distance , so units matter.

Sometimes a linear boundary simply won't work, no matter what value of C . The example below is such a case.



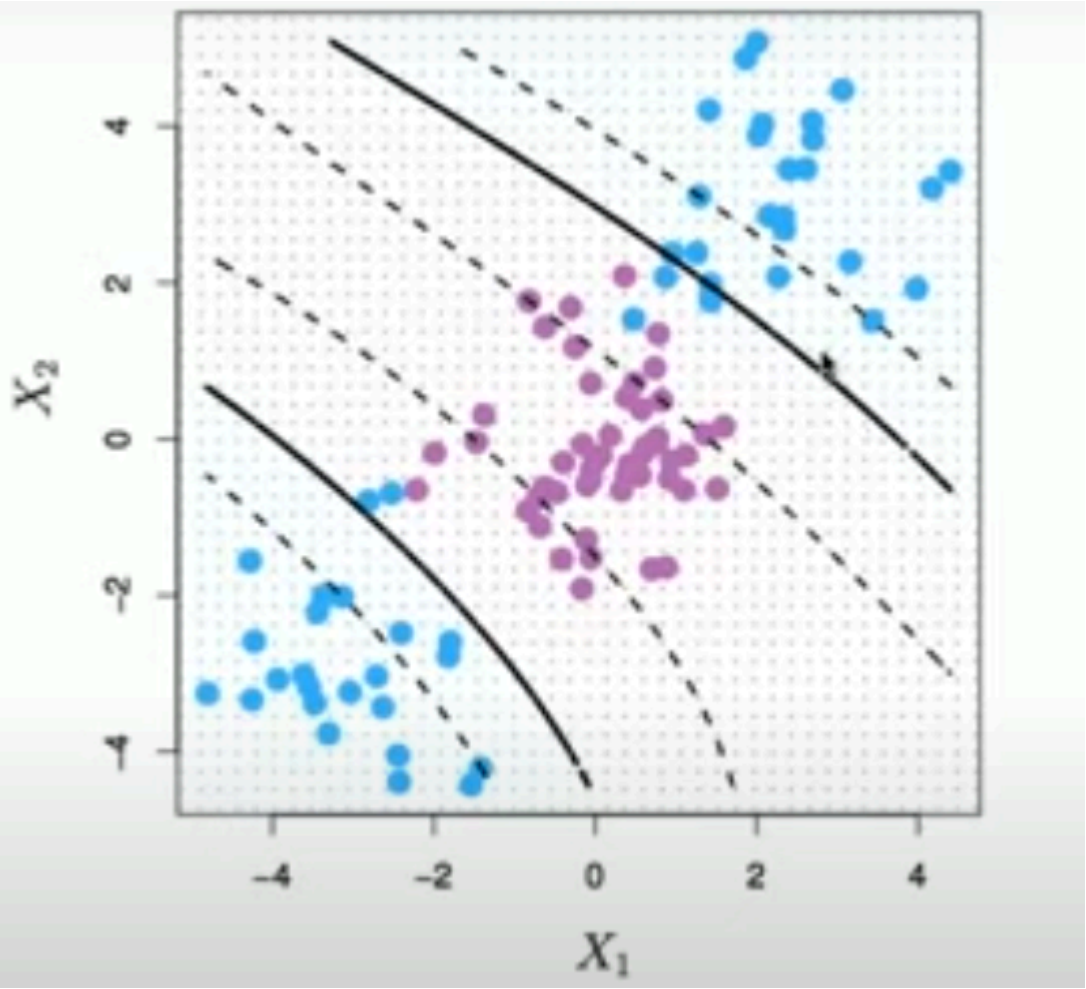
Feature Expansion

- Enlarge the space of features by including transformations; e.g. $X_1^2, X_2^2, X_1X_2, X_1X_2^2, \ldots$. Hence go from a p- dimensional space to a $m > p$ dimensional space.
- Fit a support-vector classifier in the enlarged space.
- This results in a non-linear decision boundaries in the original space.
  Example: Suppose we use $(X_1, X_2, X_1^2, X_2^2, X_1X_2)$ instead of just (X_1, X_2) . Then the decision boundary would be of the form

$$\beta_0 + \beta_1X_1 + \beta_2X_2 + \beta_3X_1^2 + \beta_4X_2^2 + \beta_5X_1X_2 = 0$$

  This leads to nonlinear decision boundaries in the original space (quadratic conic sections).

$Cubic\ \ Polynomials$



Here we use a basis expansion of cubic polynomials. From 2 variables to 9. The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space. Below is the equation:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 + \beta_6 X_1^3 + \beta_7 X_2^3 + \beta_8 X_1 X_2^2 + \beta_9 X_1^2 X_2 = 0$$

Non-linearities and Kernels

- Polynomials (especially high-dimensional ones) get wild rather fast.
- There is a more elegant and controlled way to introduce non-linearities in support-vector classifiers - through the use of $kernels$
- But you need to understand inner products in support vectors first. The inner product between two vectors is given by:

$$< x_i, x_{i'} >= \sum_{j=1}^{p} x_{ij} x_{i'j'}$$

- The linear support vector classifier with n parameters can be represented as

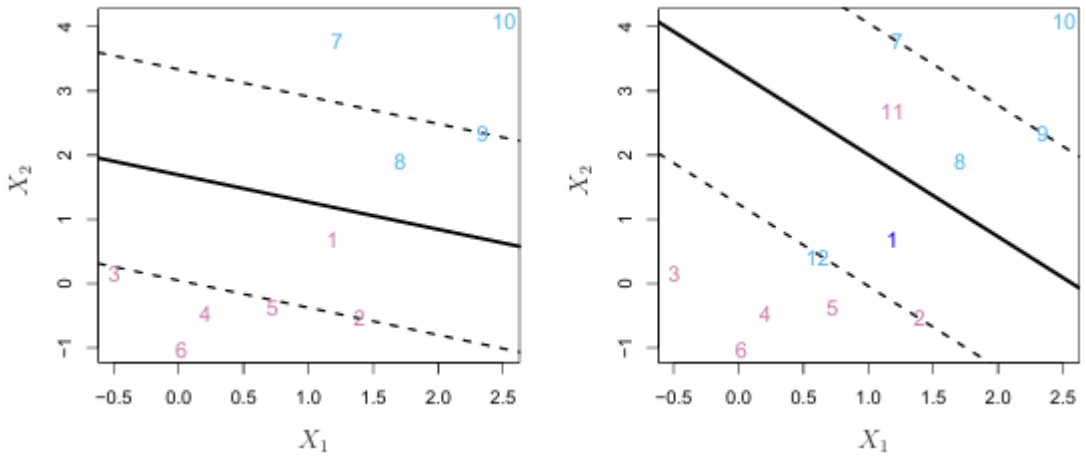
$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i < x, x_i >$$

- To estimate the parameters $\alpha_1, \ldots, \alpha_n$ and β_0 , all we need are the $\binom{n}{2}$ inner products $< x_i, x_i' >$ between all pairs of training observations.

It turns out that most of the $\hat{\alpha}_i$ can be zero:

$$f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i < x, x_i >$$

S is the $support\ set$ of indices i such that $\hat{\alpha}_i > 0$.



If you look into the image on the left, which are the support vectors or points with non-zero alphas ? Points 1,5,2,8 and 9 . Why?
Certainly the ones on the margin, because they define the direction. But the ones are also gonna define the direction. The points on the other side of the margin aren't support points because even if we moved it and kept in the same area, it's not gonna affect the solution.

$Kernel\ and\ Support\ Vector\ Machines$

- If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!
- Some special **kernel functions** can do this for us. E.g.

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$$

computes the inner-products needed for d dimensional polynomials — $\binom{p+d}{d}$ basis functions!

- **Try it for p = 2 and d = 2.**
- The solution has the form

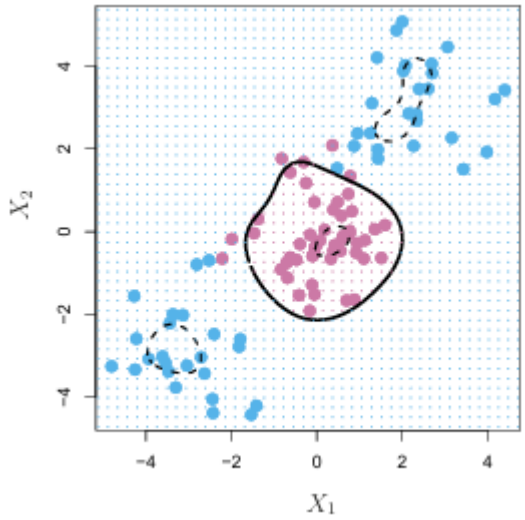
$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i).$$

Radial Kernel

- $$K(x_i, x_i') = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$
- Implicit feature space; very high dimensional

  $$f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i K(x, x_i)$$

  Controls variance by squashing down most dimensions severely.
  γ is a tuning parameter. If it is very large it's like having a small standard deviation and you get much more wiggly decision boundaries and smoother boundaries if it's small.



Let us again take the example of the heart data.

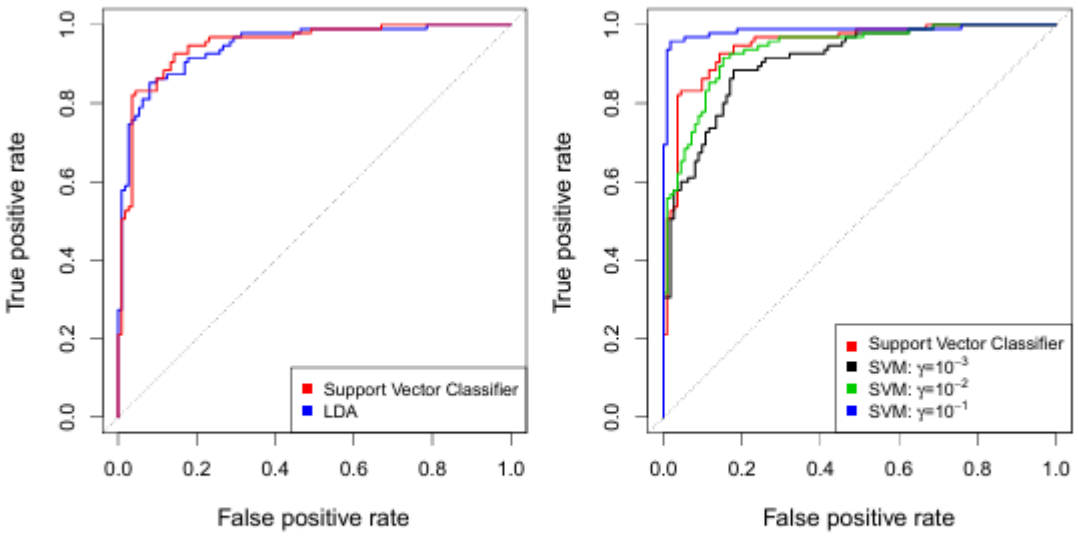
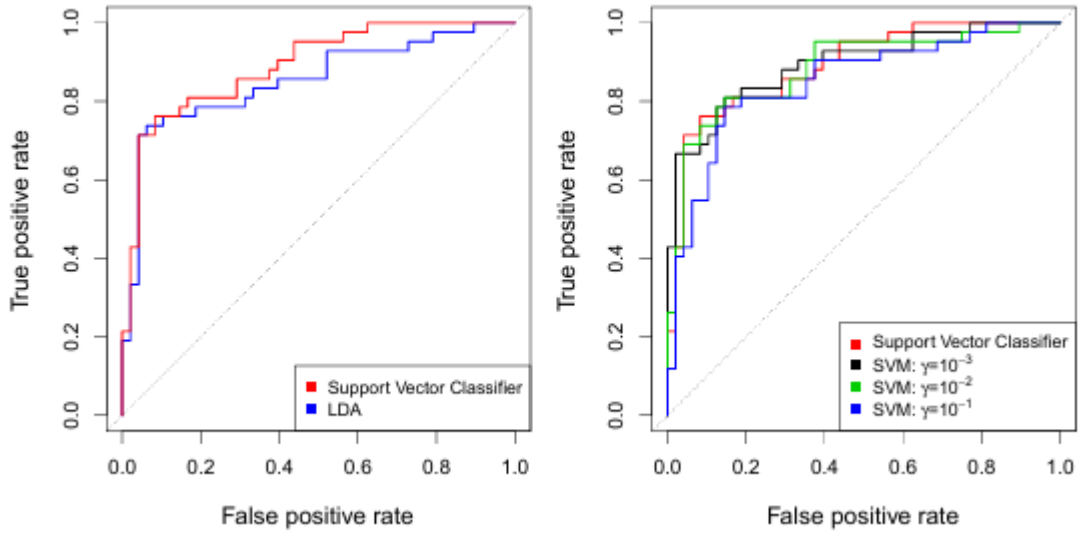


FIGURE 9.10. *ROC curves for the* `Heart` *data training set.* Left: *The support vector classifier and LDA are compared.* Right: *The support vector classifier is compared to an SVM using a radial basis kernel with* $\gamma = 10^{-3}$, 10^{-2}, *and* 10^{-1}.

ROC curve is obtained by changing the threshold 0 to threshold t in $\hat{f}(X) > t$, and recording $false\ positive$ and $true\ positive$ rates as t varies. Here we use ROC curves on training data. Just to remind, the more the curve is close to the top-left corner , the better. The above curves are for the training data.
Also AUC (Area Under Curve) is a good measurement of how close you went. Unity is the perfect score, you should always remain above the 45^o line, i.e. $0.5 < AUC < 1$.

Let us look at the test data now:



Using 80 observations put aside as test observations. Notice in the right curve that the SVM with the largest γ is doing worst now, while it was doing best in the training data. Support Vector

Classifier or linear classifier is doing pretty okay. Keep in mind that we have another tuning parameter C.

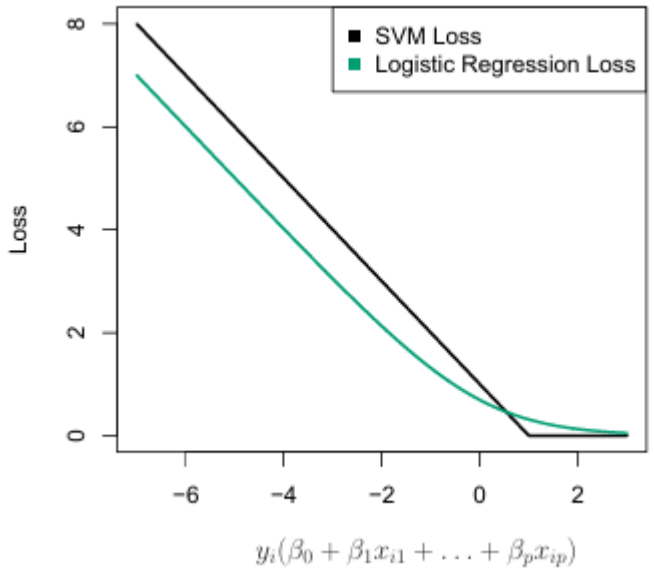
SVMs : more than 2 classes?

- The SVM are defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?
  OVA : One versus All . Fit K different 2-class SVM classifiers $\hat{f}_k(x)$, $k = 1, \ldots, K$; each class versus the rest. Classify x^* to the class for which $\hat{f}_k(x^*)$ is largest.
  OVO : One versus One. Fit all $\binom{K}{2}$ pairwise classifiers $\hat{f}_{kl}(x)$. Classify x^* to the class that wins the most pairwise competitions.
- Which to choose? If K is not too large, use OVO.

Support Vector vs Logistic Regression

With $f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$ can rephrase support-vector classifier optimization problem as

$$\max_{\beta_0, \beta_1, \ldots, \beta_p} \{ \sum_{i=1}^{n} \max[0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^{p} \beta_j^2 \}$$



This has the form $loss\ plus\ penalty$. The loss is known as $hinge\ loss$. Very similar to loss in logistic regression (negative log-likelihood).

So which to choose?

- When classes are (nearly) separable, SVM does better than LR. So does LDA.
- When not , LR (with ridge penalty) and SVM very similar.

- If you wish to estimate probabilities, LR is the choice.
- For non-linear boundaries, kernel SVMs are popular. Can use kernels with LR and LDA as well, but computations are more expensive.
- Support vectors don't choose features, so high-dimensional problems it can be a problem not so much for the classification problem but for the interpretation. Also the drawback with probabilities, many cases like cancer detection, you want them. Because 0.51 and 0.99 will give same classification but implications are very different.
- The SVM community has tried addressing these using post ad-hoc add ons like recursive feature elimination but you can do that directly using Lasso-regurlarized Logistic regression.