Classification

Classification

- Examples
  - Who will default on their loans?
  - Identify a road or building in a picture.
  - Identify the party of a politician based on speech.
- Econometrics
  - Binary: OLS, logit, probit.
  - Multiple unordered classes: multnomial and conditional logit, multinomial probit.
- Machine Learning
  - Discriminant analysis
  - Naive Bayes
  - Support Vector Machines,...

Classification

- Qualitative variables take values in an unordered set $\mathcal{C}$, such as:
  `eye color`$\in \{$`brown`, `blue`, `green`$\}$
  `email`$\in \{$`spam`, `ham`$\}$.
- Given a feature vector X and a qualitative response Y taking values in the set $\mathcal{C}$, the classification task is to build a function $C(X)$ that takes as input the feature vector X and predicts its value for Y; i.e. $C(X) \in \mathcal{C}$.
- Often we are more interested in estimating the *probabilities* that X belongs to each category in $\mathcal{C}$.

Classification with large p

1. Regularized Logistic Regression
2. Support Vector Machine

L1 regularized maximum likelihood

The standard maximum likelihood estimator solves

$$\max_{\beta} l\left(x,\beta\right).$$

Suppose that we have a linear index model, where $l\left(x,\beta\right)=l\left(\beta' x\right)$. The L1 regularized maximum likelihood solves

$$\max_{\beta} l\left(x,\beta\right)-\lambda\sum_{j=1}^{p}\left|\beta_j\right|,$$

where the intercept β_0 is not penalized.

Likelihood

For independent binary response data we can write the likelihood as

$$L(\theta, x) = F(x_1, ..., x_n, \theta) = \prod_{i=1}^{n} F(x_i, \theta)$$

and the log likelihood as

$$\ln L(\theta, x) = l(\theta, x) = \sum_{i=1}^{n} \ln F(x_i, \theta).$$

- In the **Logit model**, $F(z) = P[y = 1 \mid x]$ is the logistic function (let $z = \beta' x$)

$$F(z) = \frac{e^z}{1 + e^z}.$$

L1 regularized logistic regression

The likelihood is

$$L = \prod_{i=1}^{n} \left(F\left[\beta' x_i\right]\right)^{y_i} \left(1 - F\left[\beta' x_i\right]\right)^{(1-y_i)}, \text{ so}$$

$$\begin{aligned} l &= \log L = \sum_{i=1}^{n} y_i \ln\left(F\left[\beta' x_i\right]\right) + (1 - y_i) \ln\left(1 - F\left[\beta' x_i\right]\right) \\ &= \sum_{i=1}^{n} \left(y_i \beta' x_i - \ln\left(1 + e^{\beta' x_i}\right)\right). \end{aligned}$$

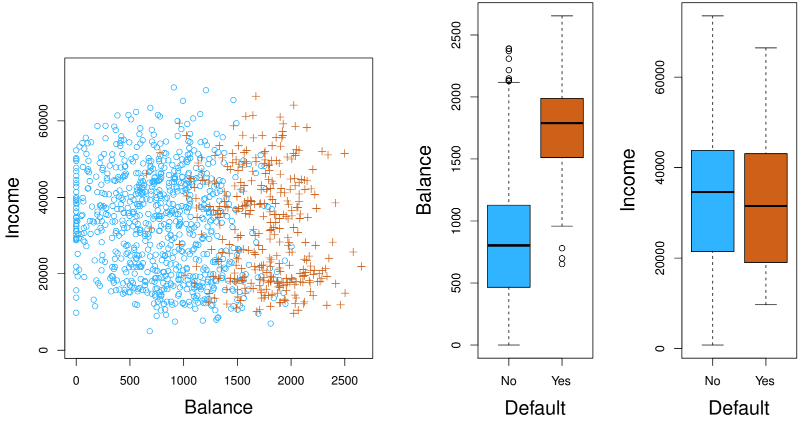
and we solve

$$\max_{\beta} \sum_{i=1}^{n} \left(y_i \beta' x_i - \ln\left(1 + e^{\beta' x_i}\right)\right) - \lambda \sum_{j=1}^{p} \left|\beta_j\right|.$$

Estimation

- Path algorithms such as LAR for lasso are more difficult.
- The R package *glmnet* can fit coefficient paths for very large logistic regression problems efficiently (large in N or p). Their algorithms can exploit sparsity in the predictor matrix X, which allows for even larger problems.
- Stata 16 command *lasso* fits regularized logit (and probit and poisson) models.

Example: Credit Card Default



Credit Data

		True Default Status		
		No	Yes	Total
Predicted	No	9644	252	9896
Default Status	Yes	23	81	104
	Total	9667	333	10000

$(23 + 252)/10000$ errors — a 2.75% misclassification rate!

Some caveats:

- This is *training* error, and we may be overfitting. Not a big concern here since $n = 10000$ and $p = 2$!

Credit Data

		True Default Status		
		No	Yes	Total
Predicted	No	9644	252	9896
Default Status	Yes	23	81	104
	Total	9667	333	10000

$(23 + 252)/10000$ errors — a 2.75% misclassification rate!

Some caveats:

- This is *training* error, and we may be overfitting. Not a big concern here since $n = 10000$ and $p = 2$!
- If we classified to the prior — always to class `No` in this case — we would make $333/10000$ errors, or only 3.33%.

Credit Data

		True Default Status		
		No	Yes	Total
Predicted	No	9644	252	9896
Default Status	Yes	23	81	104
	Total	9667	333	10000

$(23 + 252)/10000$ errors — a 2.75% misclassification rate!

Some caveats:

- This is *training* error, and we may be overfitting. Not a big concern here since $n = 10000$ and $p = 2$!
- If we classified to the prior — always to class `No` in this case — we would make 333/10000 errors, or only 3.33%.
- Of the true `No`'s, we make $23/9667 = 0.2\%$ errors; of the true `Yes`'s, we make $252/333 = 75.7\%$ errors!

Types of errors

False positive rate: The fraction of negative examples that are classified as positive — 0.2% in example.

False negative rate: The fraction of positive examples that are classified as negative — 75.7% in example.

We produced this table by classifying to class `Yes` if

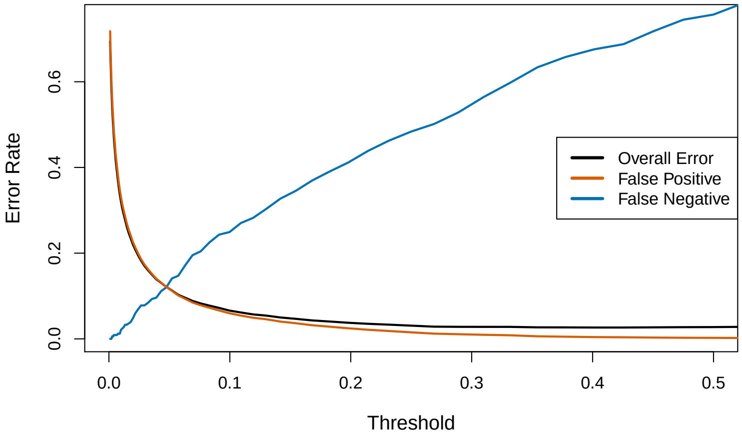
$$\widehat{\Pr}(\texttt{Default} = \texttt{Yes}|\texttt{Balance}, \texttt{Student}) \geq 0.5$$

We can change the two error rates by changing the threshold from 0.5 to some other value in $[0, 1]$:

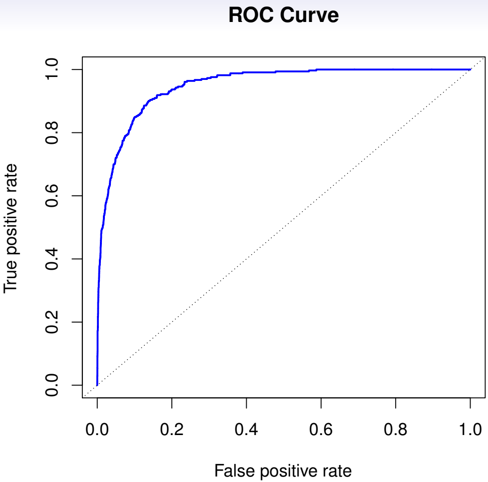
$$\widehat{\Pr}(\texttt{Default} = \texttt{Yes}|\texttt{Balance}, \texttt{Student}) \geq \textit{threshold},$$

and vary *threshold*.

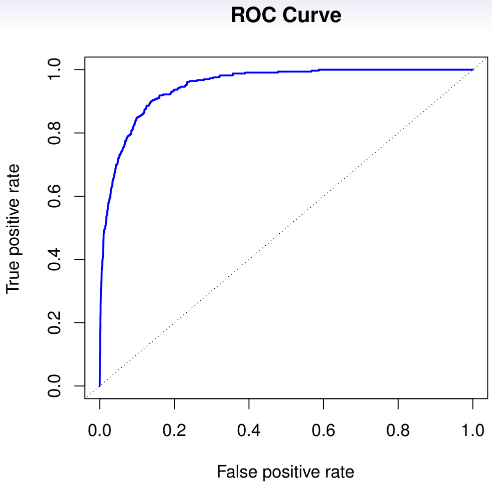
Varying the *threshold*



In order to reduce the false negative rate, we may want to reduce the threshold to 0.1 or less.



The *ROC plot* displays both simultaneously.



The *ROC plot* displays both simultaneously.

Sometimes we use the AUC or *area under the curve* to summarize the overall performance. Higher AUC is good.

Confusion matrix

- Table of true and predicted classes.
- Terminology

		True class		
Predicted class		No	Yes	Total
	No	True Neg: (TN)	False Neg. (FN)	N*
	Yes	False Pos. (FP)	True Pos. (TP)	P*
	Total	N	P	

Name	Definition	Synonyms
False Pos. rate	FP/N	Type I error, 1−Specificity
True Pos. rate	TP/P	1−Type II error, power, sensitivity, recall
Pos. Pred. value	TP/P^*	Precision, 1−false discovery proportion
Neg. Pred. value	TN/N^*	

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 cannot, 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.

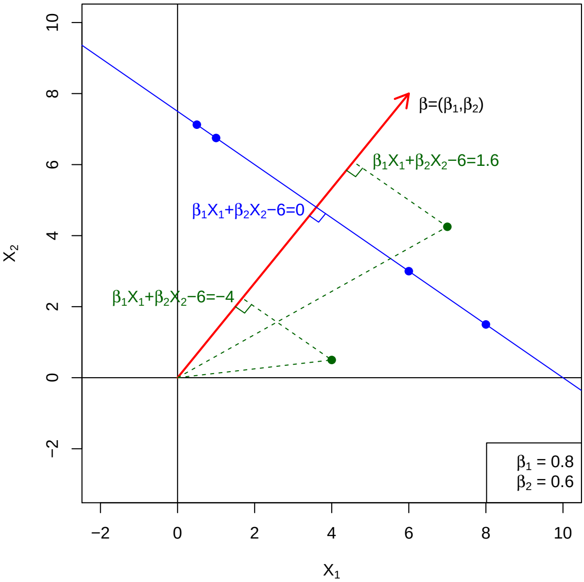
What is a Hyperplane?

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

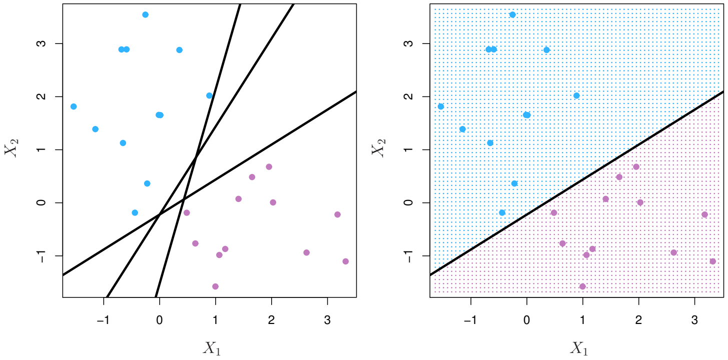
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$

- In $p = 2$ dimensions a hyperplane is a line.
- If $\beta_0 = 0$, the hyperplane goes through the origin, otherwise not.
- The vector $\beta = (\beta_1, \beta_2, \cdots, \beta_p)$ is called the normal vector — it points in a direction orthogonal to the surface of a hyperplane.

Hyperplane in 2 Dimensions



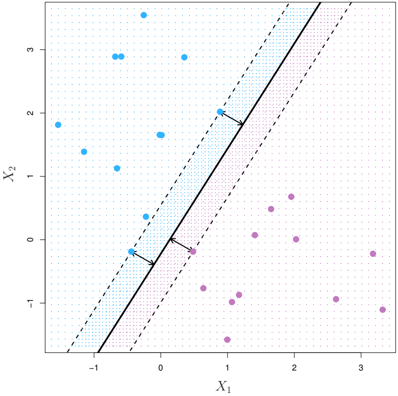
Separating Hyperplanes



- 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, say, and $Y_i = -1$ 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

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M$$

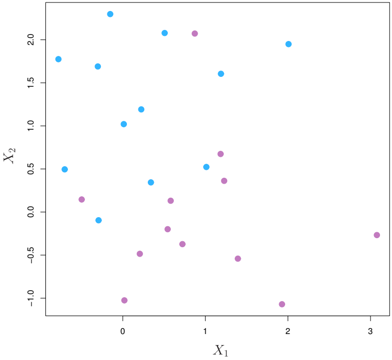
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M$$

$$\text{for all } \; i = 1, \ldots, N.$$

This can be rephrased as a convex quadratic program, and solved efficiently. The function `svm()` in package `e1071` solves this problem efficiently

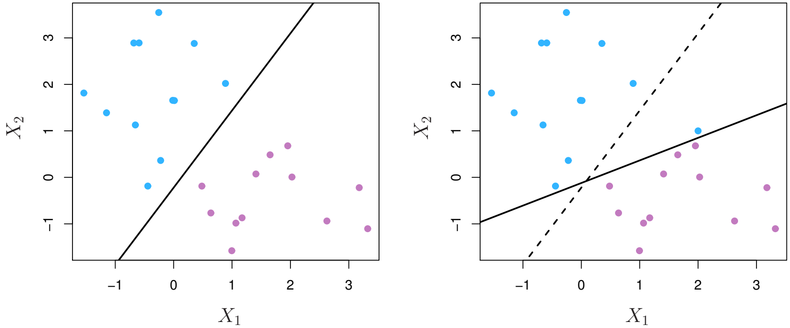
Non-separable Data



The data on the left are not separable by a linear boundary.

This is often the case, unless $N < p$.

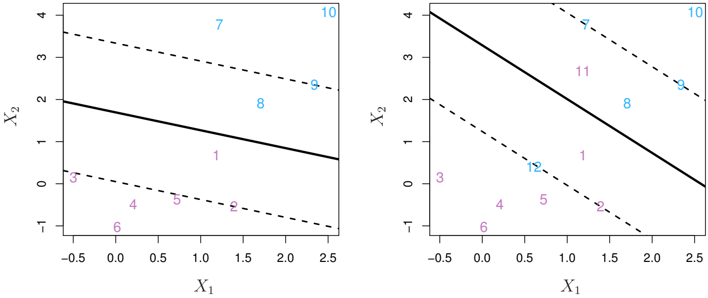
Noisy Data



Sometimes the 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.

Support Vector Classifier



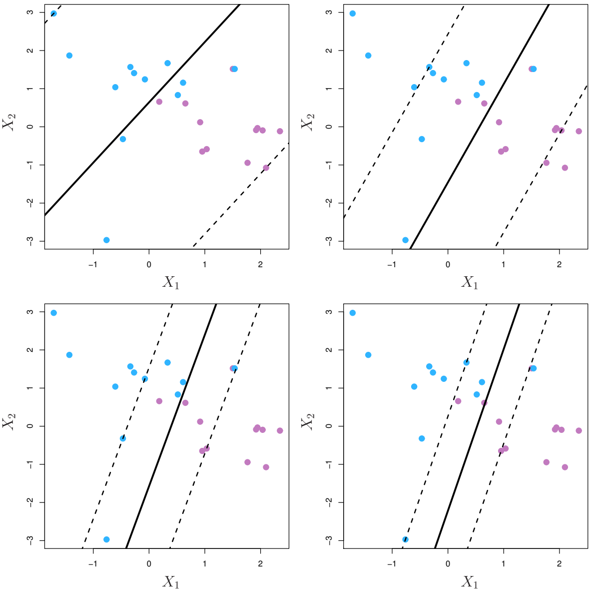
$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

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

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

NB: different C than below.

C is a regularization parameter



Separating hyperplanes

A hyperplane or affine set L is defined by the equation

$$f(x) = \beta_0 + x'\beta = 0.$$

If we are in $\mathbb{R}^2$, this is a line. Some properties

1. For any two points x_1 and x_2 lying in L, $v = x_1 - x_2$ is parallel to the hyperplane. Since $(x_1 - x_2)'\beta = 0$, $\beta^* = \beta / \|\beta\|$ is the vector normal to the surface of L.
2. The signed distance of any point x to L is given by $(x_1'\beta = -\beta_0)$

$$(x - x_1)' \beta^* = \frac{1}{\|\beta\|} (x - x_1)' \beta = \frac{1}{\|\beta\|} \left(x'\beta + \beta_0\right) = \frac{1}{\|\beta\|} f(x).$$

Hence $f(x)$ is proportional to the signed distance from x to the hyperplane defined by $f(x) = 0$.

Our training data consists of n pairs $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$. Consider the optimization problem of choosing the maximal separating margin

$$\max_{\beta, \beta_0} M$$

subject to

$$y_i \left(x_i' \beta + \beta_0\right) \frac{1}{\|\beta\|} \geq M, \quad \forall i,$$

or equivalently,

$$y_i \left(x_i' \beta + \beta_0\right) \geq M \|\beta\|, \quad \forall i.$$

We can arbitrarily set the scaling of $\|\beta\|$ to $1/M$. Thus we get ($\text{argmin}\|\beta\| = \text{argmin}\frac{1}{2}\|\beta\|^2$)

$$\min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2,$$

subject to

$$y_i \left(x_i' \beta + \beta_0\right) \geq 1, \quad i = 1, 2, ..., n.$$

($\|\beta\| = 1$ m better than $\|\beta\| = 1$ km).

The Lagrangeian is

$$\min_{\beta,\beta_0} \mathcal{L} = \frac{1}{2}\|\beta\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i \left(x_i'\beta + \beta_0 \right) - 1 \right].$$

The first order conditions w.r.t β and β_0 are

$$\begin{aligned} \beta &= \sum_{i=1}^{n} \alpha_i y_i x_i, \qquad (1) \\ 0 &= \sum_{i=1}^{n} \alpha_i y_i. \end{aligned}$$

Substituting in $\mathcal{L}$ we obtain the so-called Wolfe dual

$$\mathcal{L}_D = \underbrace{\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i' x_j}_{\beta'\beta} - \sum_{i=1}^{n} \alpha_i y_i \underbrace{\sum_{i=1}^{n} \alpha_j y_j x_j' x_i}_{\beta'} - \beta_0 \underbrace{\sum_{i=1}^{n} \alpha_i y_i}_{=0} + \sum_{i=1}^{n} \alpha_i$$

$$\min_{\alpha} \mathcal{L}_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha_i \alpha_k y_i y_k x_i' x_k. \qquad (2)$$

subject to

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

This is a simpler convex optimization problem, for which standard software can be used. In addition the solution must satisfy the Kuhn–Tucker conditions, which include the above conditions and

$$\alpha_i \left[y_i \left(x_i' \beta + \beta_0 \right) - 1 \right] = 0, \quad i = 1, 2, ..., n.$$

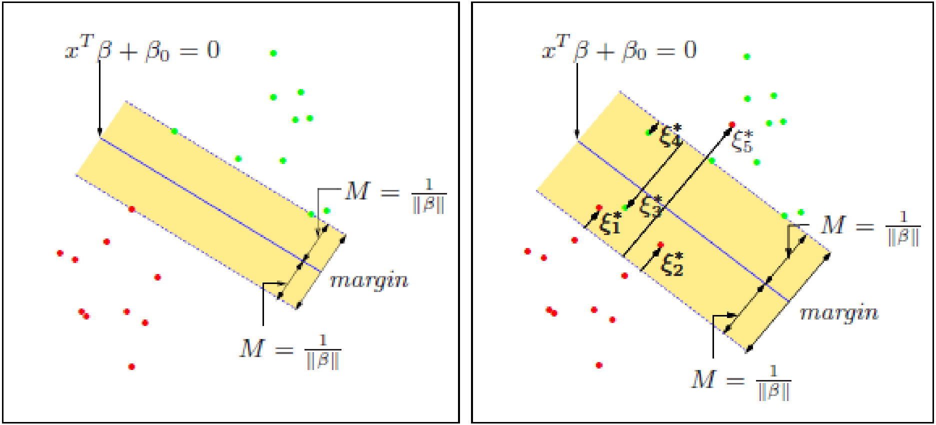
Thus

- if $\alpha_i > 0$, then $y_i \left(x_i' \beta + \beta_0 \right) = 1$, in other words, x_i is on the boundary,
- if $y_i \left(x_i' \beta + \beta_0 \right) > 1$ then $\alpha_i = 0$ and x_i is not on the boundary.
- From equation 1, we see that the solution vector is defined in terms of a linear combination of the support points x_i — those points defined to be on the boundary with $\alpha_i > 0$.

Support Vector Machines

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

$$\xi = (\xi_1, \xi_2, ..., \xi_n) .$$



We now solve

$$\min_{\beta,\beta_0} \frac{1}{2} \|\beta\|^2 ,$$

subject to

$$\begin{aligned} y_i \left(x_i'\beta + \beta_0\right) &\geq 1 - \xi_i, \quad \forall i, \\ \xi_i &\geq 0, \quad \sum_{i=1}^{n} \xi_i \leq \text{ constant}, \end{aligned}$$

resulting in Lagrangeian

$$\mathcal{L} = \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left[y_i \left(x_i'\beta + \beta_0\right) - (1 - \xi_i) \right] - \sum_{i=1}^{n} \mu_i \xi_i .$$

The first-order conditions w.r.t. β and β_0 are as in equation (1) and the new (wrt ξ_i)

$$\alpha_i = C - \mu_i, \quad \forall i,$$

as well as $\alpha_i, \mu_i, \xi_i > 0\ \forall i$ The dual Lagrangeian is as in equation (2). We minimize this subject to

$$\begin{aligned} 0 &\leq \alpha_i \leq C \\ 0 &= \sum_{i=1}^{n} \alpha_i y_i. \end{aligned}$$

The Kuhn-Tucker conditions include

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

for all i.

- The coefficient vector again has the form

$$\widehat{\beta} = \sum_{i=1}^{n} \widehat{\alpha}_i y_i x_i$$

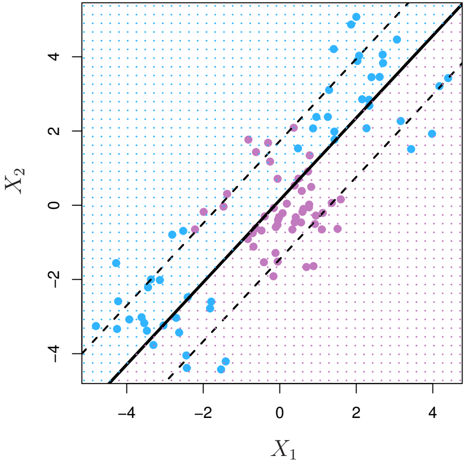
  with nonzero coefficients $\widehat{\alpha}_i$ i only for those observations i for which $y_i \left(x_i' \beta + \beta_0 \right) = (1 - \xi_i)$ due to the first Kuhn-Tucker condition.
- These observations are called the support vectors, since $\widehat{\beta}$ is represented in terms of them alone. Among these support points, some will lie on the edge of the margin ($\xi_i = 0$), and hence will be characterized by $0 \leq \widehat{\alpha}_i \leq C$ $(\widehat{\alpha}_i = C - \widehat{\mu}_i)$. The remainder will have $\alpha_i = C$.
- C is a tuning parameter. A large value of C will discourage any positive ξ_i.

- Once you have the $\widehat{\alpha}_i$, classification is easy

$$\begin{aligned} f(x) &= \beta_0 + x'\beta = \beta_0 + x' \sum_{i=1}^{n} \widehat{\alpha}_i y_i x_i \\ &= \beta_0 + \sum_{i=1}^{n} \widehat{\alpha}_i y_i x' x_i . \end{aligned}$$

  - Only observations that are support vectors ($\widehat{\alpha}_i > 0$) matter.
  - Positive correlation ($x'x_i > 0$) with positive examples ($y_i = 1$) and negative correlation ($x'x_i < 0$) with negative examples ($y_i = -1$) increase $f(x)$ chance of being coded as positive
  - Correlation with examples $K(x, x_i)$ can be made more general, see below.

Linear boundary can fail



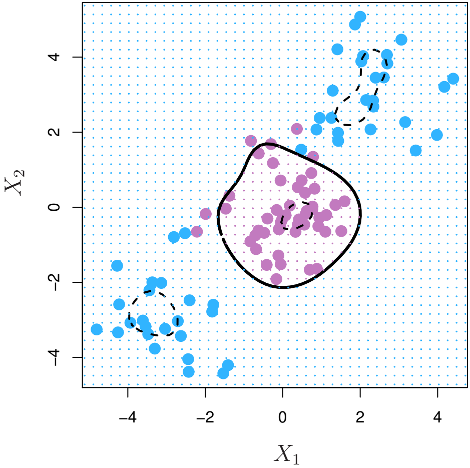
Sometime a linear boundary simply won't work, no matter what value of C.

The example on the left is such a case.

What to do?

Radial Kernel

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2).$$

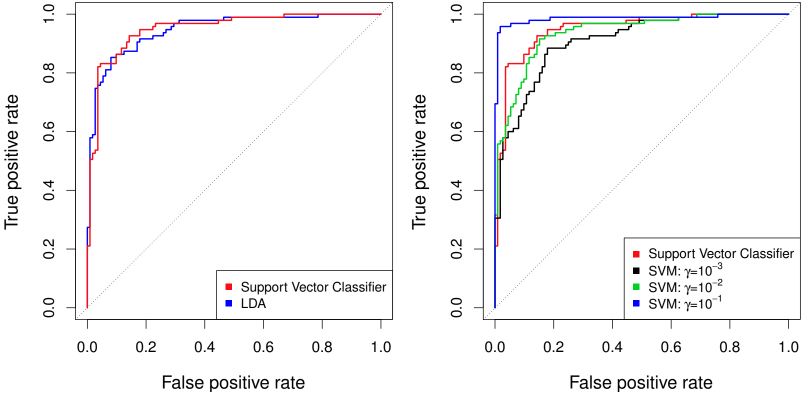


$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i)$$

Implicit feature space; very high dimensional.

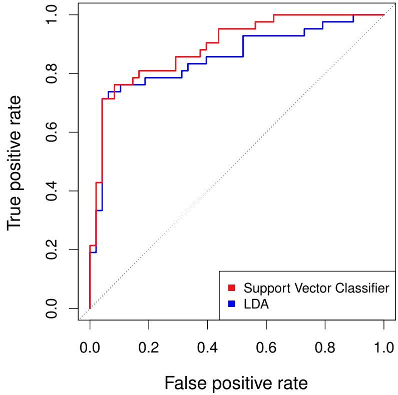
Controls variance by squashing down most dimensions severely

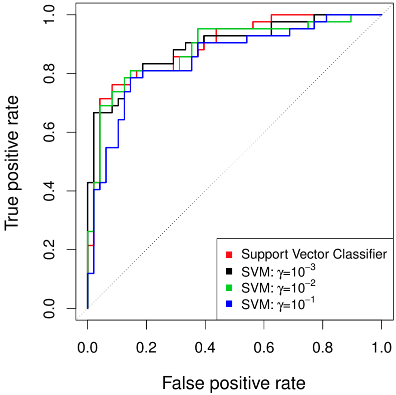
Example: Heart Data



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 see ROC curves on training data.

Example continued: Heart Test Data





SVMs: more than 2 classes?

The SVM as 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 $\hat{f}_{k\ell}(x)$. Classify x^* to the class that wins the most pairwise competitions.

Which to choose? If K is not too large, use OVO.

Which to use: SVM or Logistic Regression

- 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 nonlinear boundaries, kernel SVMs are popular. Can use kernels with LR and LDA as well, but computations are more expensive.

Readings

Classification:
Introduction to Statistical Learning: Chapter 4.

Support Vector Machines:
Introduction to Statistical Learning: Chapter 9.
Elements of Statistical Learning: Chapter 12