MATH 189
Support Vector Machine

Wenxin Zhou
UC San Diego

Time: 2:00—3:20 & 3:30—4:50pm TueThur
Zoom ID: 985 7463 1067

Outline

- In previous lectures, we discussed Fisher's discriminant analysis and logistic regression method for classification.
  - Discriminant analysis
    - Bayes' formula
    - LDA (equal covariance matrix)
    - QDA (unequal covariance matrices)
  - Logistic regression (simple and multiple)

- Today we will introduce another classification method, named support vector machine.
  - Separating hyperplane
  - Maximal margin classifier
  - Support vector classifier

Support Vector Machine

- Support vector machine (SVM) is a classification method developed in the computer science community in 1990s, and has grown in popularity since then.

- Support vector machines have been shown to perform well in a variety of settings, and are often considered one of the best “out of the box” classifiers.

- Support vector machines are intended for the binary classification setting in which there are two classes. There are extensions of support vector machines to the case of more than two classes. SVM can also be extended to regression tasks.

- SVM is a generalization of a simple and intuitive classifier, called the maximal margin classifier.

Hyperplane

- In a p-dimensional space, a hyperplane is a flat affine subspace of dimension $p-1$. The word affine indicates that the subspace needs not pass through the origin.

- For instance, in two dimensions, a hyperplane is a straight line. In three dimensions, a hyperplane is a plane.

- The mathematical definition of a hyperplane is quite simple. A hyperplane in p-dimensional space is

$$\{\boldsymbol{X} = (X_1, \dots, X_p)' : \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0\}.$$

- For a p-vector $\boldsymbol{X} = (X_1, \dots, X_p)'$, we say $\boldsymbol{X}$ lies on the hyperplane if $\boldsymbol{X}$ satisfies the above equation.

Hyperplane (cont.)

- A $(p-1)$-dimensional hyperplane cuts a p-dimensional space into two "sides".
- For a p-vector $\boldsymbol{X} = (X_1, \ldots, X_p)'$ that does not lie on the hyperplane, it either lies on the positive side of hyperplane:

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

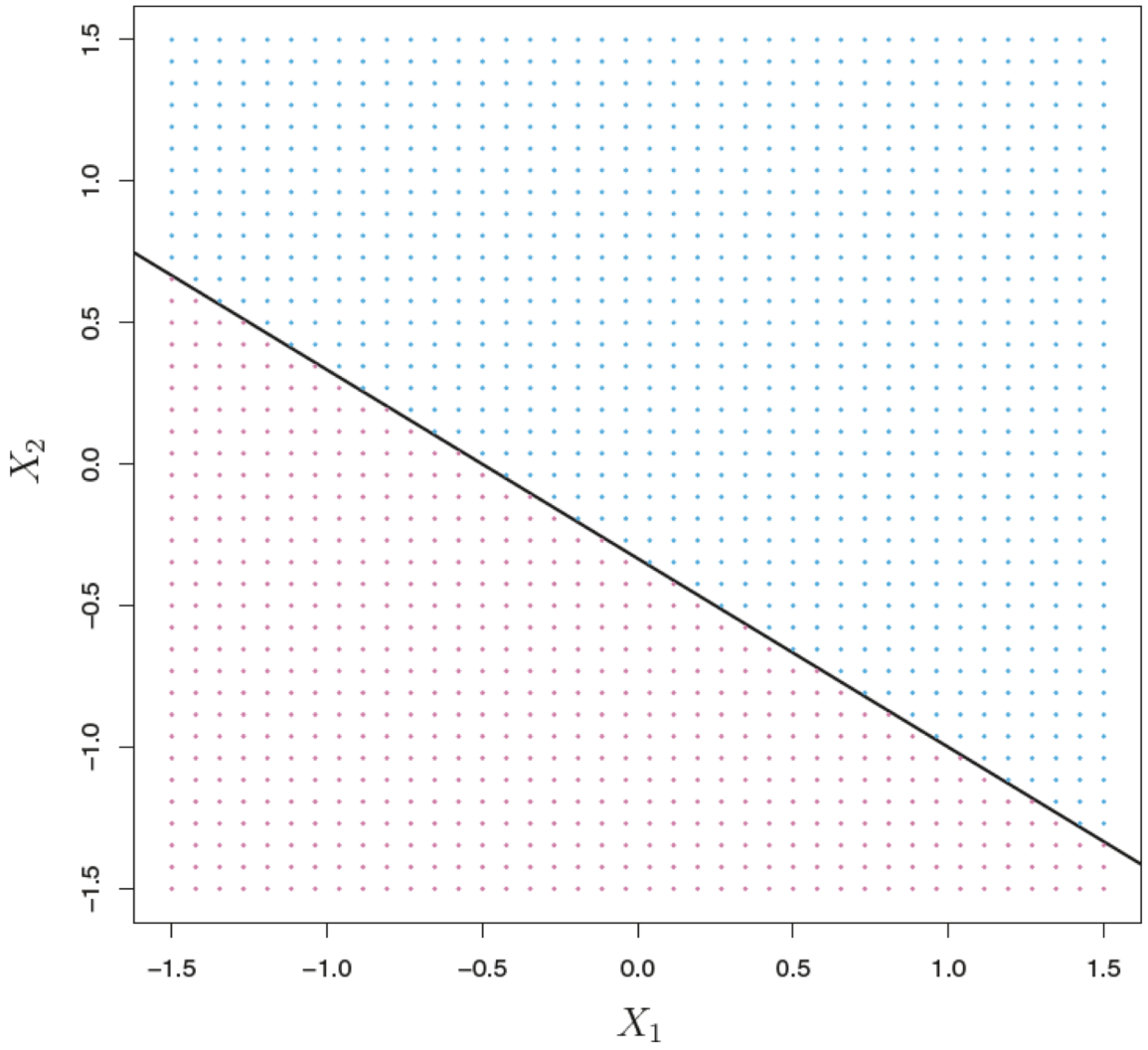
- Or lies on the negative side of the hyperplane:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0.$$

- For a given point $\boldsymbol{X} = (X_1, \ldots, X_p)'$, one can easily determine on which side of the hyperplane it lies by calculating the sign of

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p.$$

Example: Hyperplane in $\mathbb{R}^2$



- Here we draw a hyperplane
$$1 + 2X_1 + 3X_2 = 0$$
in $\mathbb{R}^2$, displayed using a **black solid line**.

- The blue region is the set of points for which
$$1 + 2X_1 + 3X_2 > 0.$$

- The purple region is the set of points for which
$$1 + 2X_1 + 3X_2 < 0.$$

- For a given point $\boldsymbol{X} = (X_1, X_2)'$, we can decide its region by calculating the sign of
$$1 + 2X_1 + 3X_2.$$

Classification using a Hyperplane

- Suppose that we have a $n \times p$ data matrix $X = (x_1, \dots, x_n)'$ that consists of n training observations in p-dimensional space,

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, x_2 = \begin{pmatrix} x_{21} \\ \vdots \\ x_{2p} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix},$$

  and assume these observations fall into two classes—that is, $y_1, \dots, y_n \in \{-1, 1\}$.

- For a given point $\boldsymbol{X} = (X_1, \dots, X_p)'$, we can determine on which side of the hyperplane it lies by calculating the sign of

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p.$$

- The property of hyperplane motivates us to consider if it is possible to construct a hyperplane that separates the training observations perfectly according to their class labels.

Separating Hyperplane

- The separating hyperplane that can perfectly classify the observations by their class labels must satisfy

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} > 0 \text{ if } y_i = 1,$$
$$\text{and } \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} < 0 \text{ if } y_i = -1.$$

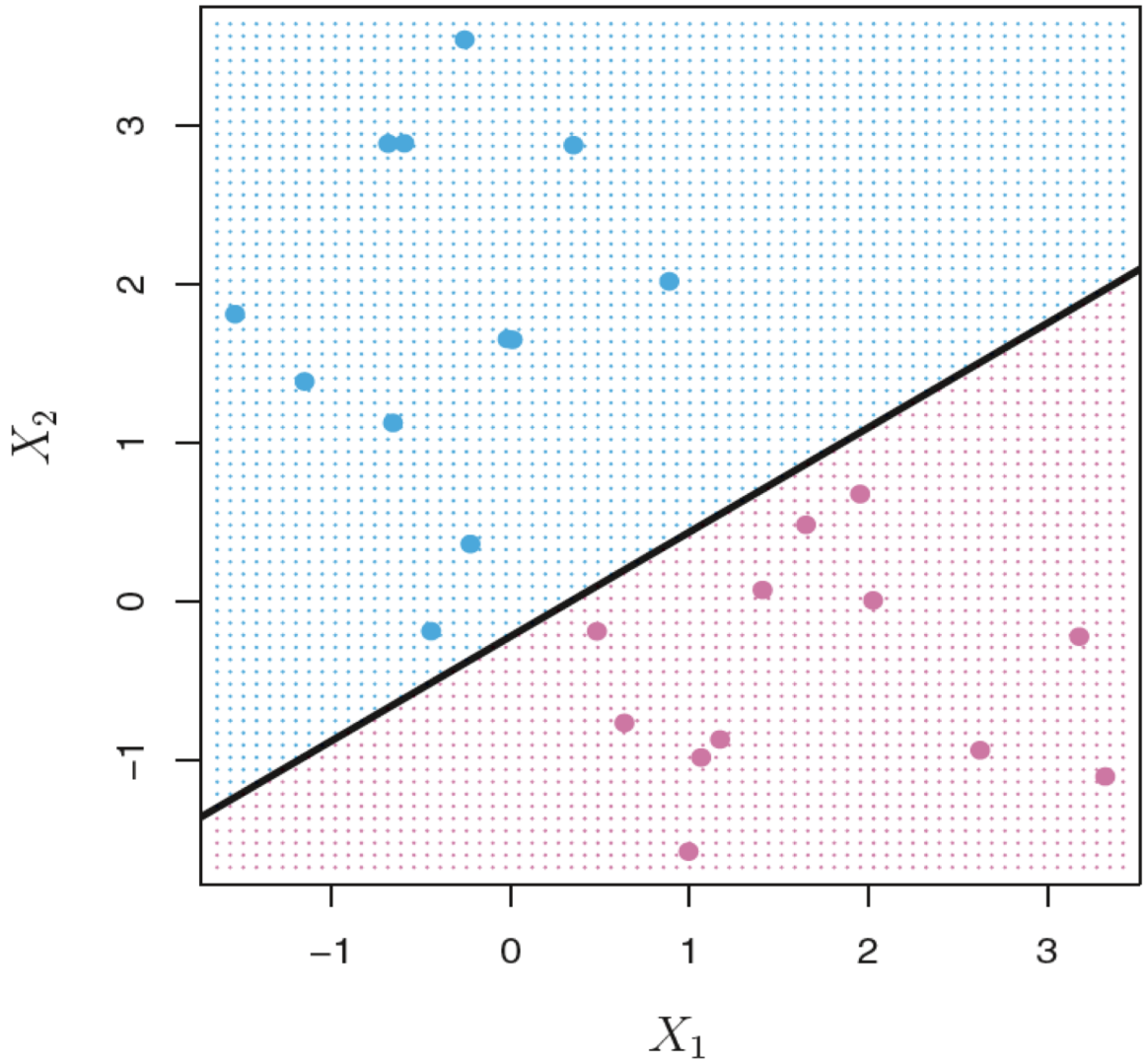
- Equivalently, a separating hyperplane has the property that

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > 0 \text{ for all } i = 1, \ldots, n.$$

- If a separating hyperplane exists, we can use it to construct a very natural classifier: a test observation is assigned to a class depending on which side of the hyperplane it is located.
- That is, we classify the test observation x^* based on the sign of

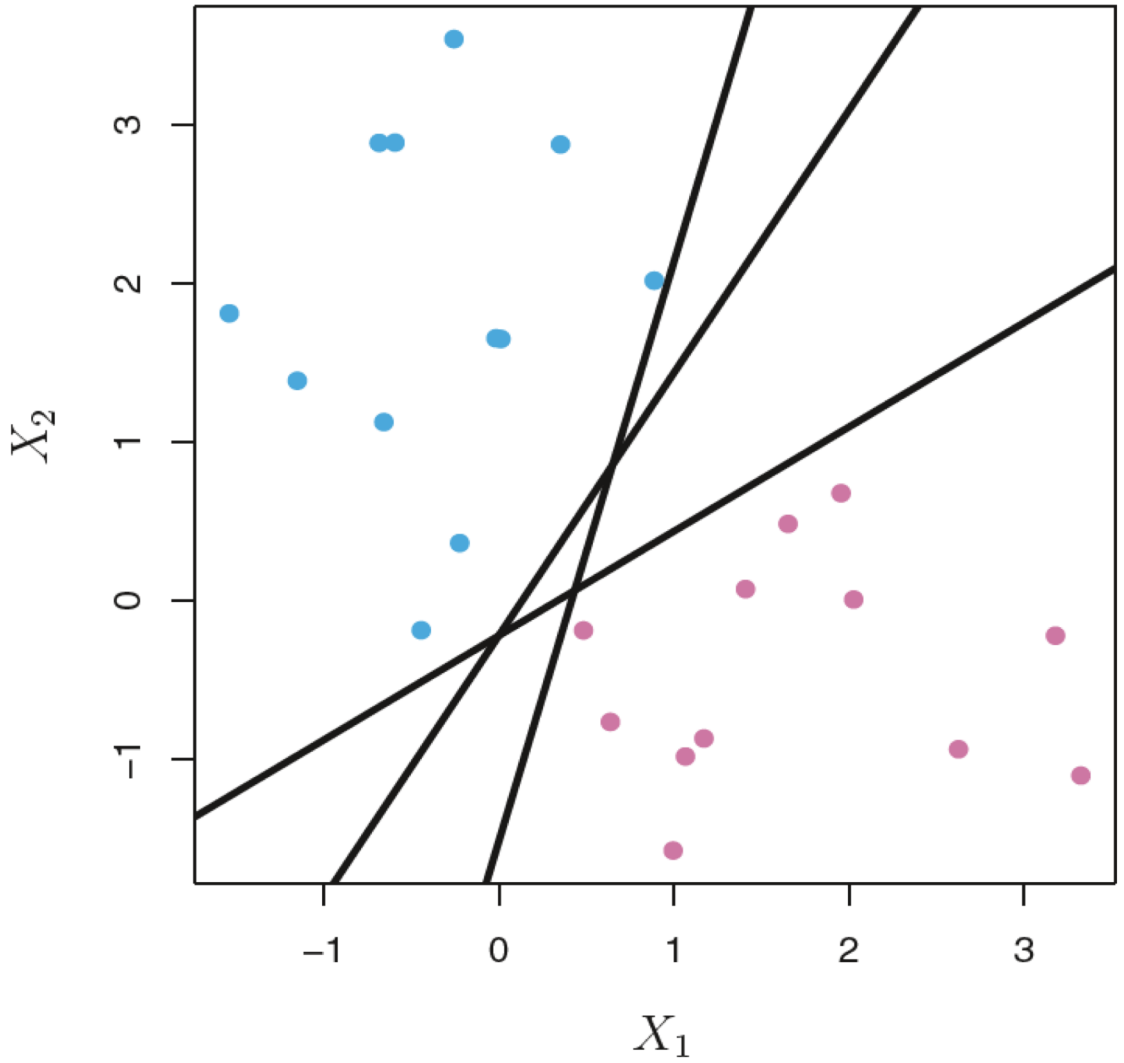
$$f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*.$$

Example: Separating Hyperplane



- Here we draw an example of a separating hyperplane in two dimensions.
- We color the observations with label $y_i = 1$ as blue and the observations with label $y_i = -1$ as purple.
- The **black solid line** shows a hyperplane that perfectly separates two classes.
- For a new observation x^*, we will predict $y^* = 1$ if x^* falls into the blue region and predict $y^* = -1$ if x^* falls into the purple region.

Issue of Separating Hyperplane



- If the data can be perfectly separated using a hyperplane, then there will in fact exist an infinite number of such 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 figure.
- In order to construct a classifier based upon a separating hyperplane, we must have a reasonable way to decide which one of the infinitely many legitimate separating hyperplanes to use.

Maximal Margin Classifier

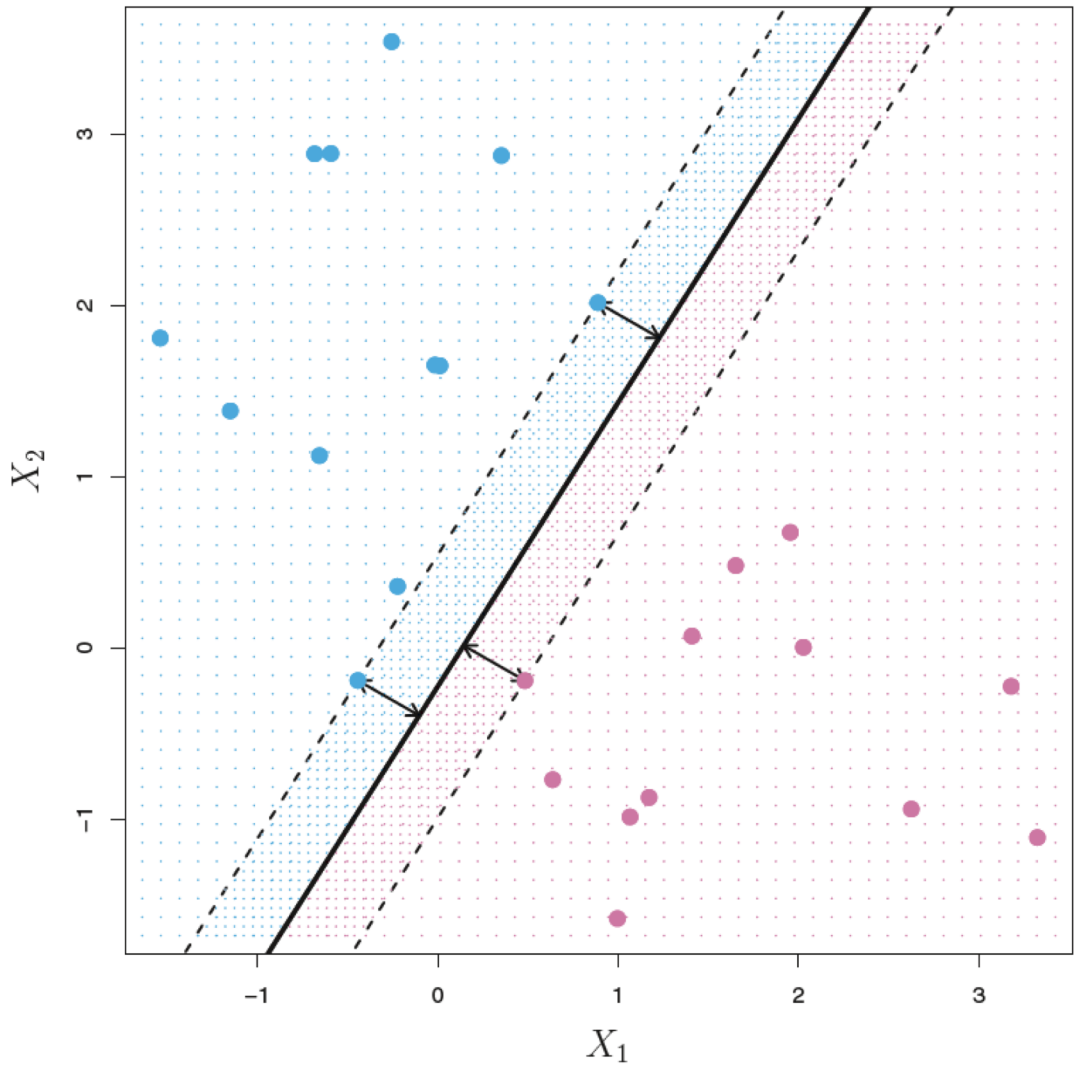
- A natural choice is the maximal margin hyperplane (also known as the optimal separating hyperplane), which is the separating hyperplane that is farthest from the training observations.

- That is, 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, which 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 largest minimum distance to the training observations.

- This is known as the maximal margin classifier. We can then classify a test observation based on which side of the maximal margin hyperplane it falls.

Example: Maximal Margin Classifier



- Here we draw the maximal margin hyperplane on this dataset. In a sense, the maximal margin hyperplane represents the mid-line of the widest "slab" that we can insert between the two classes.
- We see that 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 "support" the maximal margin hyperplane in the sense that if these points were moved slightly then the maximal margin hyperplane would move as well.

Construction of Maximal Margin Classifier

- We now consider the task of constructing the maximal margin hyperplane based on a set of n training observations $x_1, \dots, x_n \in \mathbb{R}^p$ and associated class labels $y_1, \dots, y_n \in \{-1,1\}$.

- Briefly, the maximal margin hyperplane is the solution to the optimization problem

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

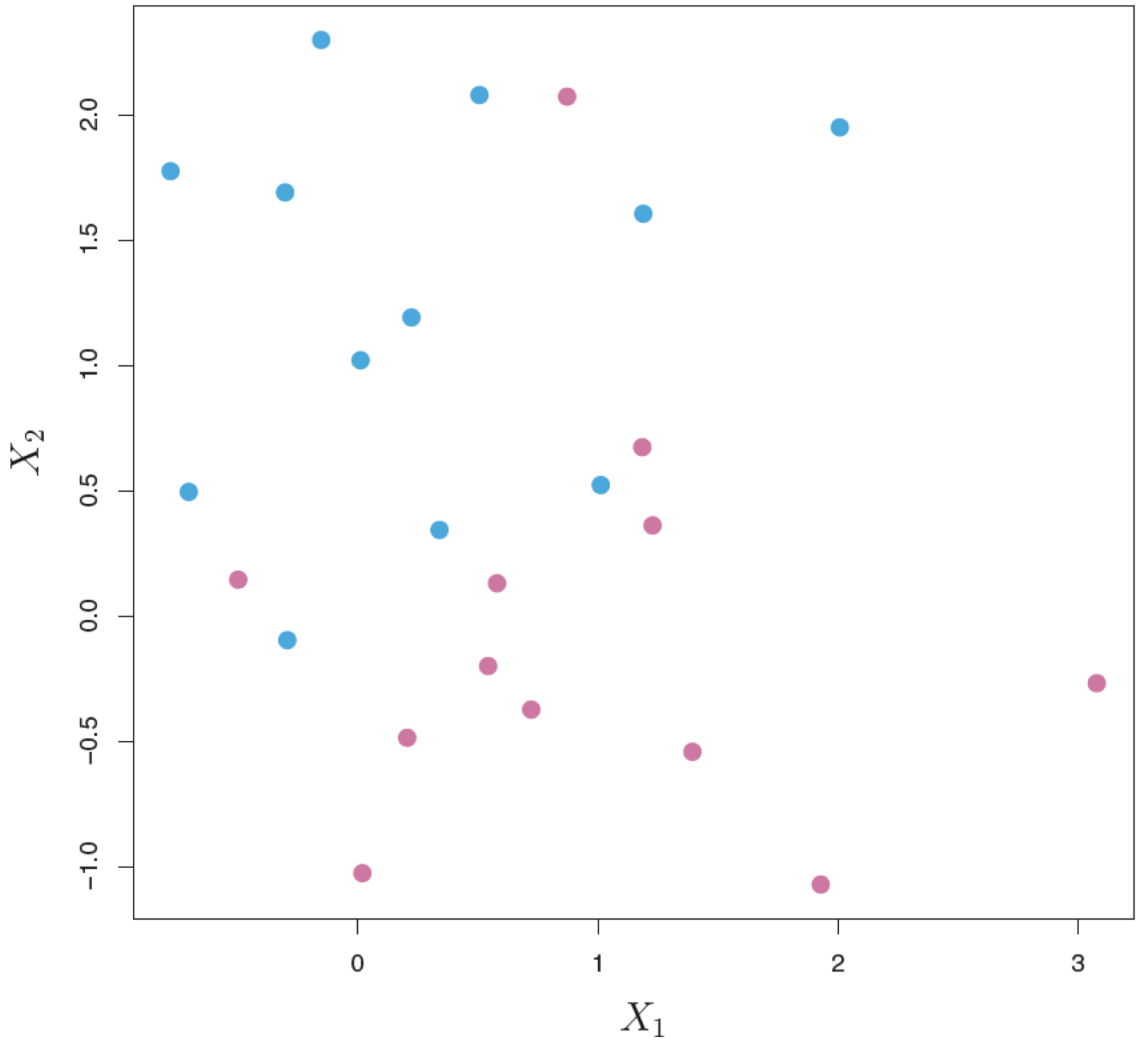
$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \text{ and} \qquad \text{(C1)}$$

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) > M \;\forall\, i = 1, \dots, n. \qquad \text{(C2)}$$

- This optimization problem is actually simpler than it looks.
  1. (C2) guarantees that each observation will be on the correct side of the hyperplane when M is positive.
  2. (C1) adds meaning to (C2). The perpendicular distance from the i-th observation to the hyperplane is given by

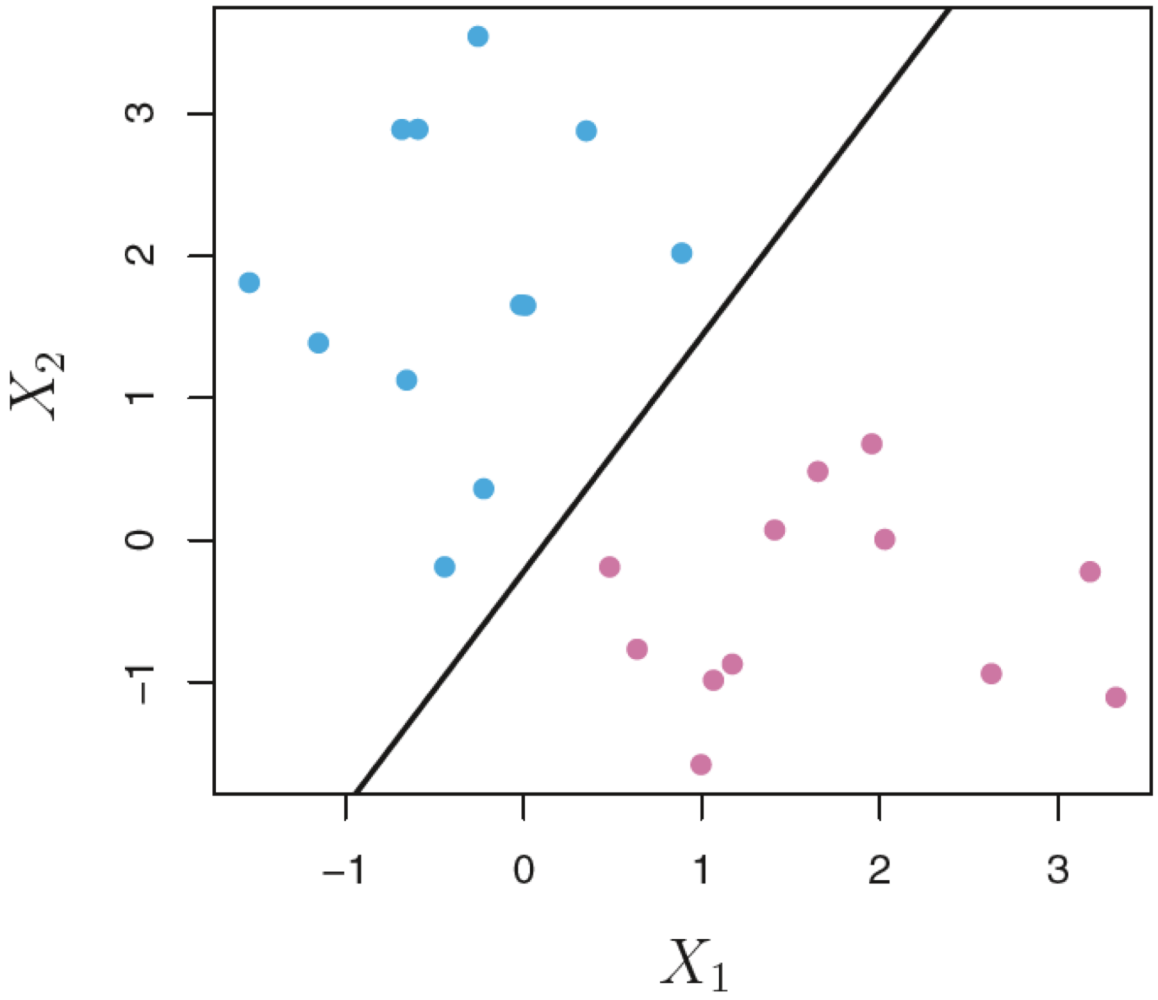
$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right).$$

Non-separable Case



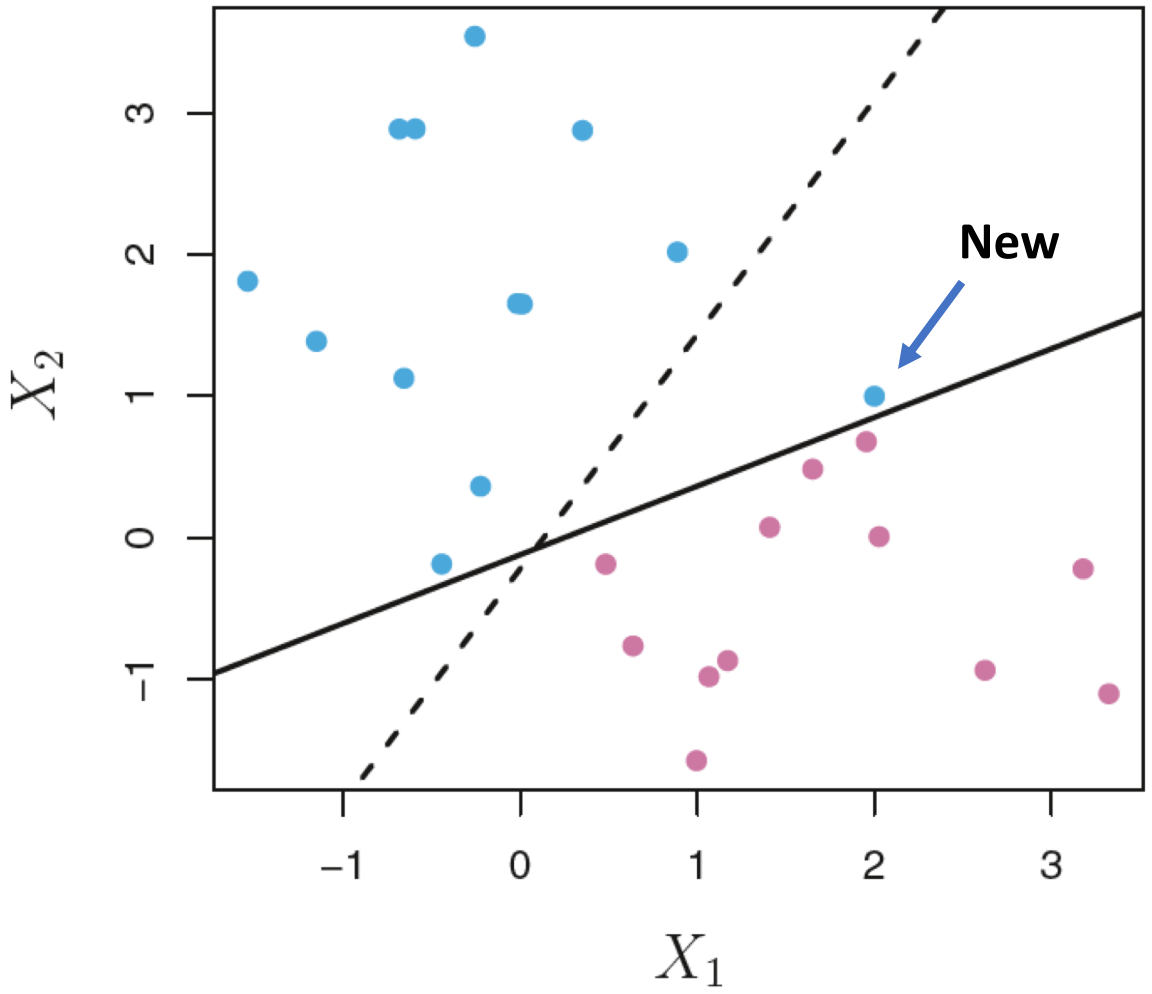
- The maximal margin classifier is a very natural way to perform classification, if a separating hyperplane exists.
- However, for more complex data no separating hyperplane exists, and so there is no maximal margin classifier.
- In such cases, the optimization problem in last slide has no solution with $M > 0$.
- An example is shown in the left: we cannot exactly separate the two classes using a line.

Issue of Maximal Margin Classifier



- In fact, even if a separating hyperplane does exist, there are also scenarios in which a maximal margin classifier may not be desirable.
- Classification based on a separating hyperplane that perfectly classifies all training observations can lead to sensitivity to individual observations.
- Consider the maximal margin classifier as shown in the left figure. Suppose that we add one single observation to the training set.

Issue of Maximal Margin Classifier



- The addition of a single observation leads to a dramatic change in the separating hyperplane (see the change from **dashed line** to **solid one**).
- The resulting maximal margin hyperplane is not satisfactory
  1. It has only a tiny margin.
  2. It is extremely sensitive to a change in a single observation.
- For better classification of ***most*** of the training observations, we may consider a classifier based on a hyperplane that does not perfectly separate the two classes.

Support Vector Classifier

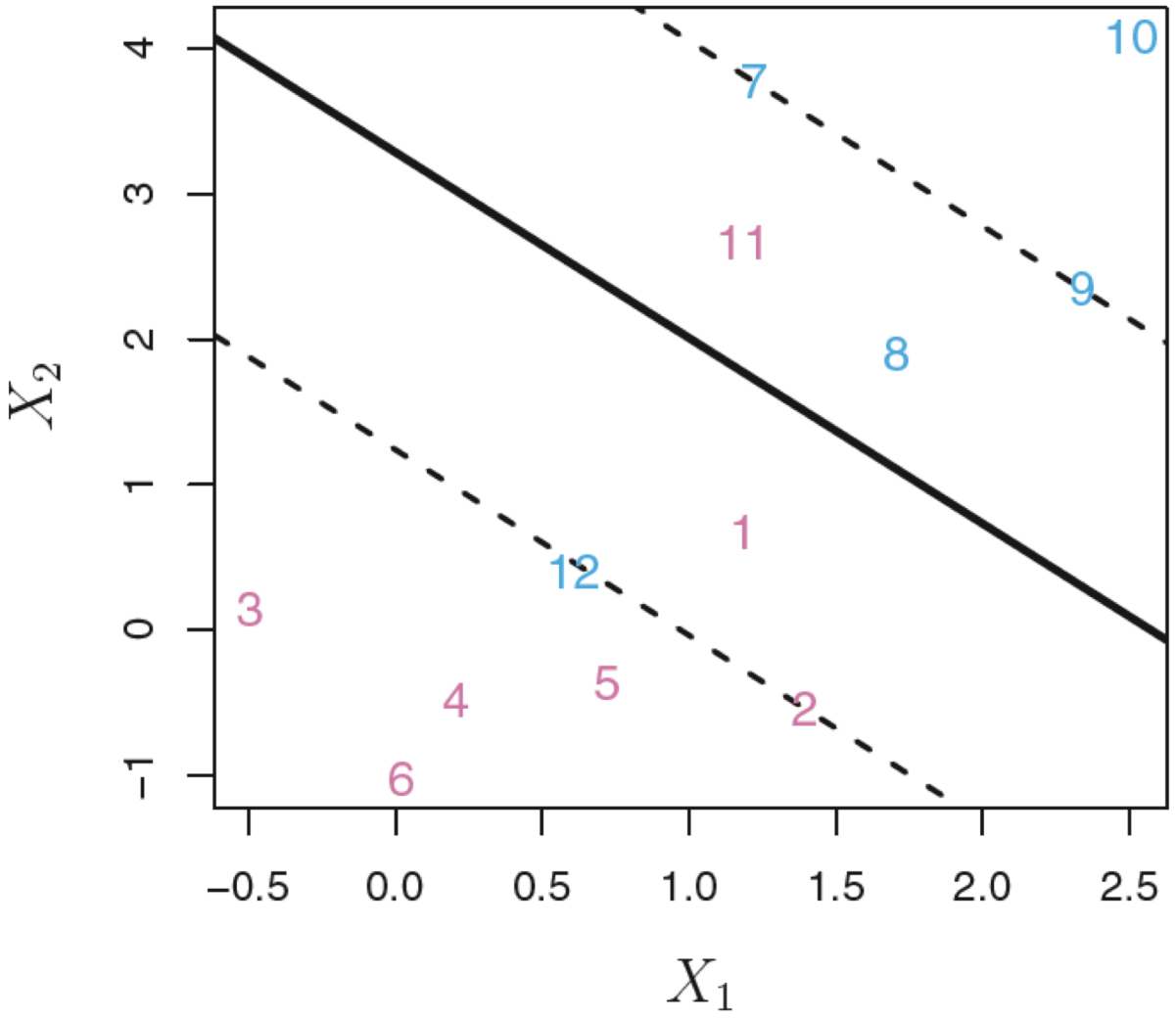
- We extend the concept of a separating hyperplane in order to develop a hyperplane that almost separates the two classes, using a so-called soft margin.

- The idea is as follows: rather than seeking the largest possible margin so that every observation is not only on the correct side of the hyperplane but also on the correct side of the margin, we instead allow some observations to be on the incorrect side of the margin, or even the incorrect side of the hyperplane.

- The margin is soft because it can be violated by some of the training observations. In fact, when the data is non-separating, such a situation is inevitable.

- This generalization of the maximal margin classifier to the non-separable case is known as the support vector classifier.

Example: Support Vector Classifier



- We fit a support vector classifier on a small toy dataset. The hyperplane is shown as a **solid line** and the margins are shown as dashed lines.
- Purple observations 3,4,5,6 and blue observation 10 are on the correct sides of hyperplane and margins.
- Purple observations 1, 2 and blue observations 7,8,9 are on the wrong sides of margins.
- Purple observation 11 and blue observation 12 are on the wrong sides of hyperplane.

Construction of Support Vector Classifier

- The support vector classifier finds the hyperplane that separate most of the training observations, but may misclassify a few of them. It is the solution of

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \qquad \text{(C1)}$$

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M(1 - \epsilon_i) \qquad \text{(C2')}$$

$$\epsilon_i > 0, \sum_{i=1}^{n} \epsilon_i \leq C, \qquad \text{(C3)}$$

  where $C \geq 0$ is a tuning parameter and M denotes the width of the margin.

- The quantities $\epsilon_1, \ldots, \epsilon_n$ are called slack variables that allow individual observations to be on the wrong side of the margin or the hyperplane
- The maximal margin classifier is a special case of support vector classifier with

$$\epsilon_1 = \cdots = \epsilon_n = 0 \text{ or } C = 0.$$

Support Vectors

- The optimization problem of support vector classifier has a very interesting property: it turns out that only observations that either lie on the margin or that violate the margin will affect the hyperplane, and hence the classifier obtained.

- Observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as support vectors.

- The fact that the support vector classifier's decision rule is based only on a potentially small subset of the training observations (the support vectors) means that it is quite robust to the behavior of observations that are far away from the hyperplane.

- This property is distinct from some of the other classification methods that we have discussed, such as linear discriminant analysis (LDA).

Slack Variables

- The slack variable ϵ_i tells us where the i-th observation is located, relative to the hyperplane and relative to the margin.
  1. If $\epsilon_i = 0$ then the i-th observation is on the correct side of hyperplane and margin.
  2. If $\epsilon_i > 0$ then the i-th observation is on the wrong side of margin.
  3. If $\epsilon_i > 1$ then the i-th observation is on the wrong side of hyperplane.

- The tuning parameter C controls the sum of ϵ_i's, and hence it determines the number and degree of the violations to the margin and hyperplane that we can tolerate.

- We can think of C as a budget for the amount that the margin can be violated by the n observations. For $C > 0$, no more than C observations can be on the wrong side of the hyperplane.

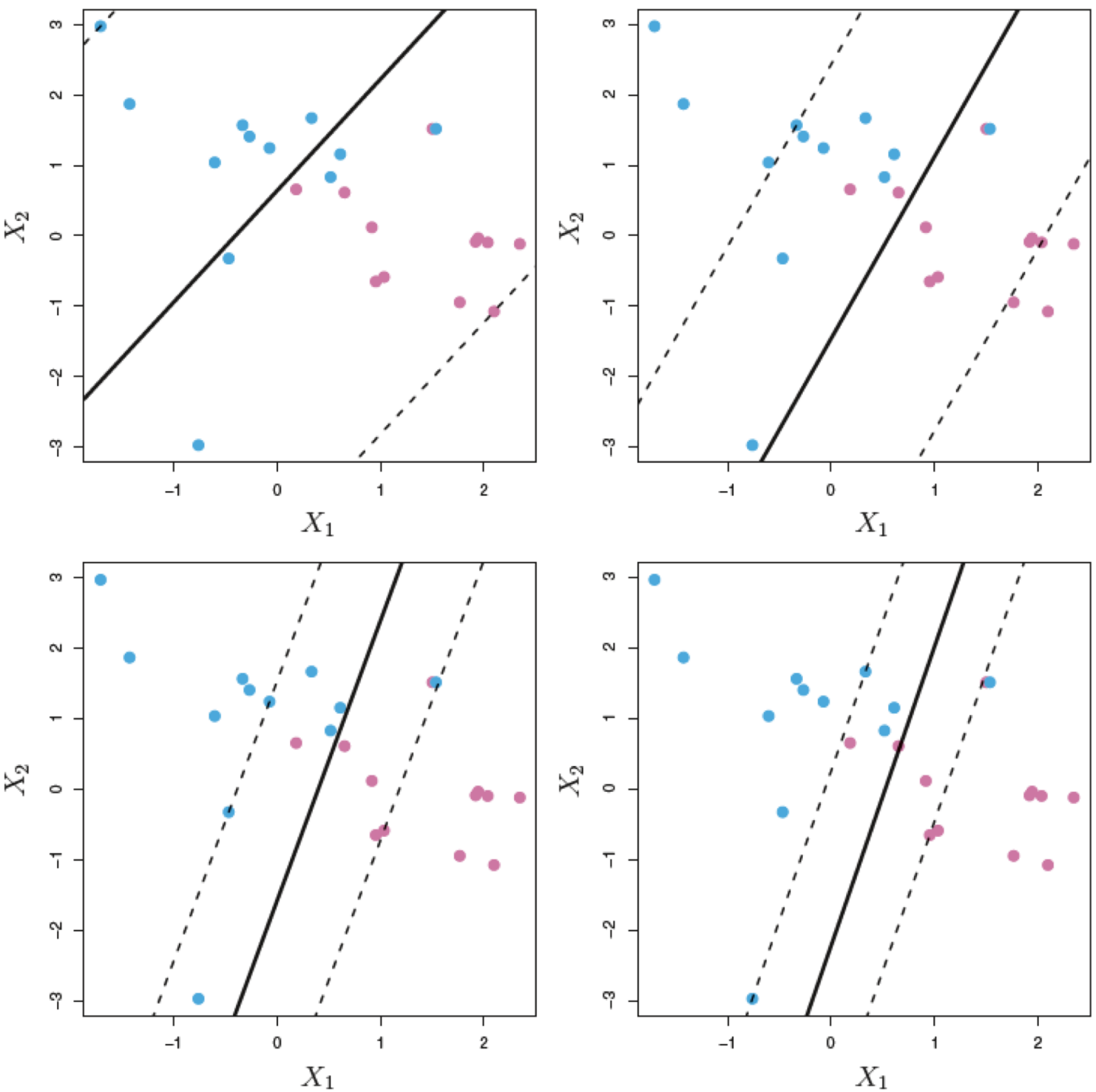
Bias-Variance Trade-off

- Similarly to the tuning parameters that we have seen throughout this course, the "violation budget" C controls the bias-variance trade-off of the support vector classifier.

- When C is small, we seek narrow margins that are rarely violated; this amounts to a classifier that is highly fit to the data, which may have low bias but high variance.

- On the other hand, when C is larger, the margin is wider and we allow more violations to it; this amounts to fitting the data less hard and obtaining a classifier that is potentially more biased but may have lower variance.

- In practice, we can choose C though Cross-Validation like other tuning parameters.



Example: Bias-Variance Trade-off

- Support vector classifier was fit using four different values of the tuning parameter C. The value of C decreases from top left to bottom right.
- The value of C can be visualized by the width of margins (i.e. gap between dashed lines).
- When C is large, 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 down.

Construction of Support Vector Classifier

- Recall that the support vector classifier finds the hyperplane that separates most of the training observations, whereas misclassify a few of them. It is the solution of

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \qquad \text{(C1)}$$

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M(1 - \epsilon_i) \qquad \text{(C2')}$$

$$\epsilon_i > 0, \sum_{i=1}^{n} \epsilon_i \leq C, \qquad \text{(C3)}$$

  where $C \geq 0$ is a tuning parameter and M denotes the width of the margin.

- The slack variables $\epsilon_1, \ldots, \epsilon_n$ allow individual observations to be on the wrong side of the margin or the hyperplane

A Penalized Problem

- The optimization problem that leads to support vector classifier can be re-written in a "Loss + Penalty" form:

$$\text{minimize}_{\beta_0,\ldots,\beta_p}\{L(\boldsymbol{X},\boldsymbol{y},\boldsymbol{\beta}) + \lambda P(\boldsymbol{\beta})\}$$

$$= \text{minimize}_{\beta_0,\ldots,\beta_p}\left\{\sum_{i=1}^{n} \max[0,1 - y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})] + \lambda \sum_{j=1}^{p} \beta_j^2\right\},$$

  where $\lambda \geq 0$ is a tuning parameter. The penalty term $\sum_{j=1}^{p} \beta_j^2$ is called the L_2-penalty

- This form gives us a better understanding of the bias-variance trade-off:
  - If λ is large then $\beta_1, \ldots, \beta_p$ are small, more violations to the margin are tolerated, and a low-variance but high-bias classifier will result.
  - When λ is small then few violations to the margin will occur; this amounts to a high-variance but low-bias classifier.
  - Thus, a small value of λ amounts to a small value of C.

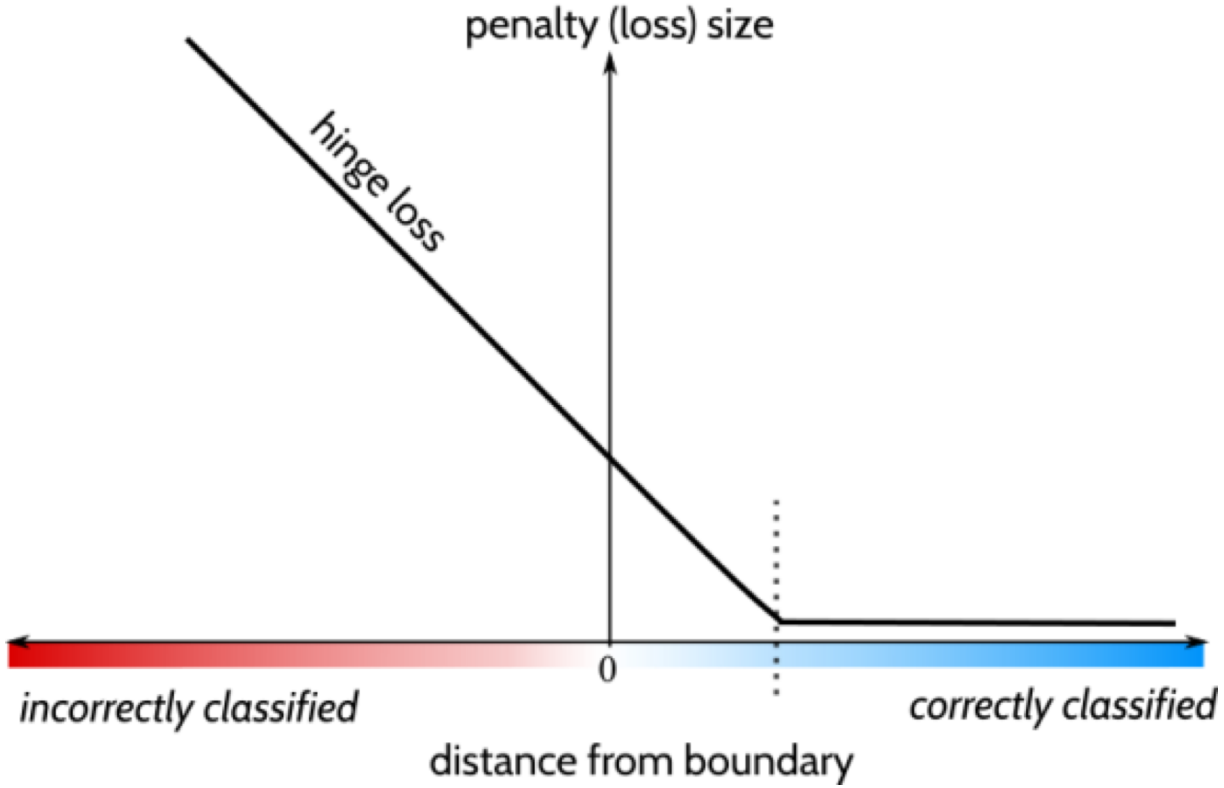
Hinge Loss

- The loss function for support vector classifier admits the form

$$L_n(\boldsymbol{\beta}) = \sum_{i=1}^{n} \max[0, 1 - y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})].$$

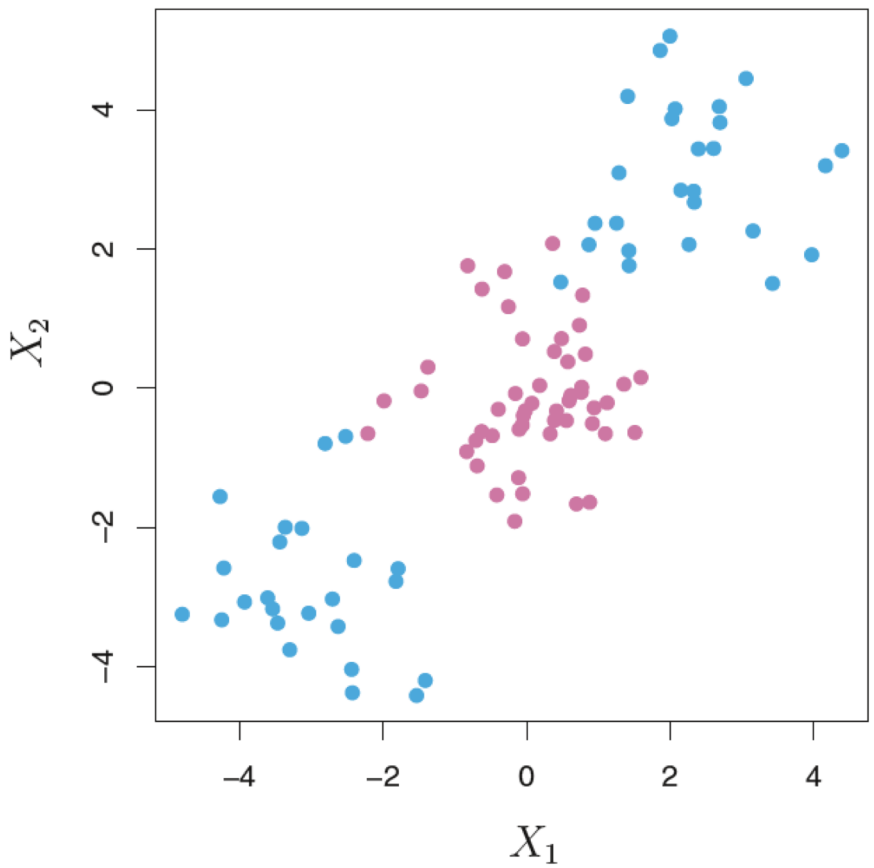
This Is known as hinge loss.

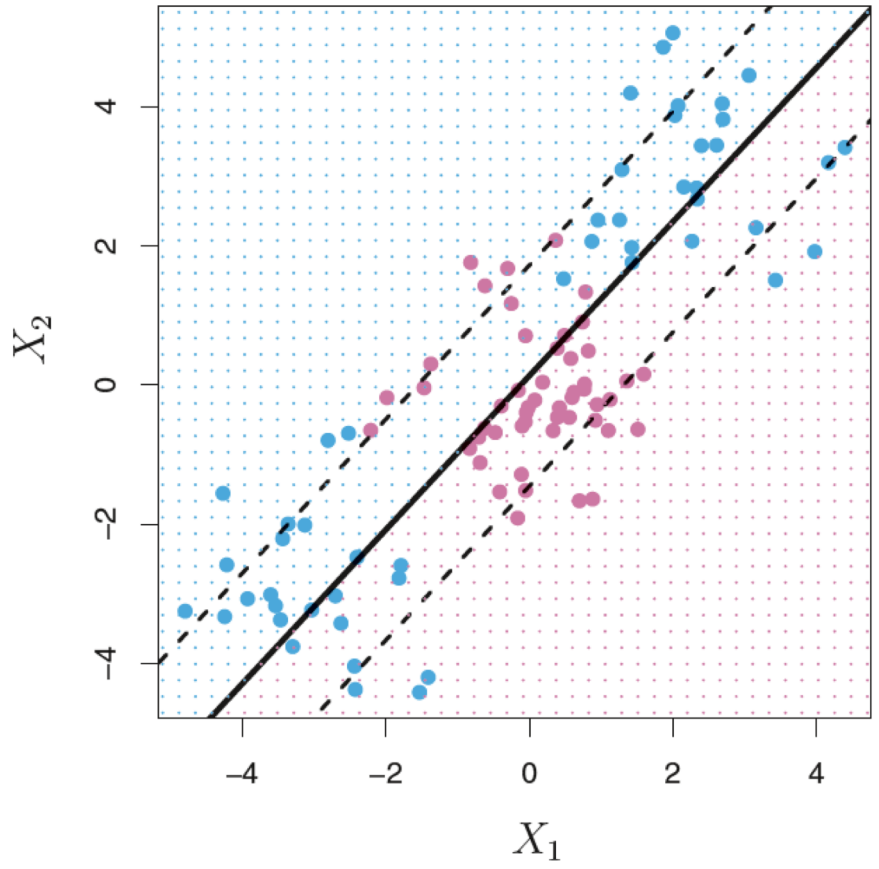
- The hinge loss is a piece-wise linear function which equals to 0 when $y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq 1.$
- The property of hinge loss shows us the reason that only support vectors play a role in the classifier obtained. The observations on the correct side of the margin do not affect it as they satisfy the above inequality.



Classification with Nonlinear Boundaries

- The support vector classifier is a natural approach for binary classification if the boundary between the two classes is linear. However, in practice we are sometimes faced with non-linear class boundaries.





Revisit Nonlinear Regression

- Recall that in multivariate linear regression, we faced a similar problem when the linearity assumption is violated.
- In that case, we consider enlarging the feature space using functions of the predictors, such as quadratic and cubic terms, in order to address this non-linearity.
- In the case of the support vector classifier, we can address the problem with possibly non-linear boundaries between classes in a similar way, by enlarging the feature space using quadratic, cubic, and even higher-order polynomial functions of the predictors.
- For instance, rather than fitting a support vector classifier using p features $X_1, \dots, X_p$, we could instead fit a support vector classifier using $2p$ features

$$X_1, X_1^2, X_2, X_2^2, \dots, X_p, X_P^2.$$

Construction of Nonlinear Support Vector Classifier

- With $2p$ quadratic features $X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_P^2$, the nonlinear support vector classifier is the solution of

$$\underset{\beta_0,\beta_{11},\beta_{12},\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to } \sum_{j=1}^{p}\sum_{k=1}^{2}\beta_{jk}^2 = 1, \qquad \text{(C1')}$$

$$\text{and } y_i\left(\beta_0 + \sum_{j=1}^{p}\beta_{j1}x_{ij} + \sum_{j=1}^{p}\beta_{j2}x_{ij}^2\right) \geq M(1-\epsilon_i) \qquad \text{(C2")}$$

$$\epsilon_i > 0, \sum_{i=1}^{n}\epsilon_i \leq C. \qquad \text{(C3)}$$

- In the enlarged feature space, the decision boundary or the above optimization problem is in fact linear.
- But in the original feature space, the decision boundary is of the form $q(x) = 0$, where q is a quadratic polynomial, and the solutions are generally non-linear.

Discussions on Nonlinear Support Vector Classifier

- One might additionally want to enlarge the feature space with higher-order polynomial terms, or with interaction terms of the form $X_j X_k$ for $j \neq k$.
- Alternatively, other functions of the predictors could be considered rather than polynomials.
- There are many possible ways to enlarge the feature space, and that unless we are careful, we could end up with a huge number of features. Computations would then become unmanageable.
- The support vector machine allows us to enlarge the feature space used by the support vector classifier in a way that leads to efficient computations.

Solution of Support Vector Classifier

- We have not discussed exactly how the support vector classifier is computed because the details become somewhat technical. However, it turns out that the solution to the support vector classifier involves only the inner products of the observations.

- It can be shown that, the linear support vector classifier can be represented as

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i y_i \langle \boldsymbol{x}, \boldsymbol{x}_i \rangle,$$

  where $\langle \boldsymbol{x}, \boldsymbol{x}_i \rangle = \sum_{j=1}^{p} x_j x_{ij}$ is the inner product between $\boldsymbol{x}$ and $\boldsymbol{x}_i$. There are n parameters $\alpha_1, \dots, \alpha_n$, one per training observation.

- To estimate the parameters α_i's and β_0, all we need are the $\binom{n}{2}$ inner products $\langle \boldsymbol{x}_i, \boldsymbol{x}_{i'} \rangle$ for all pairs $i \neq i' \in \{1, \dots, n\}$. Note that $\binom{n}{2} = n(n-1)/2$.

Support Vector and Inner Product

- According to the expression of linear support vector

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i y_i \langle \boldsymbol{x}, \boldsymbol{x}_i \rangle,$$

  in order to evaluate the function $f(x)$, we need to compute the inner product between the new point $\boldsymbol{x}$ and each $\boldsymbol{x}_i$ of the training points.

- However, it turns out that α_i is nonzero only for the support vectors in the solution. So if S is the collection of indices of support vectors, we can rewrite the above expression as

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i \in S} \alpha_i y_i \langle \boldsymbol{x}, \boldsymbol{x}_i \rangle,$$

  which typically involves far fewer terms than n.

- To estimate the parameters α_i's and β_0, all we need are the $\binom{|S|}{2}$ inner products $\langle \boldsymbol{x}_i, \boldsymbol{x}_{i'} \rangle$ for all pairs $i \neq i' \in S$.

- In summary, to construct a linear classifier $\operatorname{sign} f(x)$ and compute its coefficients, all we need are inner products!

Support Vector Machine

- This allows us to extend the support vector classifier to nonlinear decision boundaries by replacing the inner product by some nonlinear kernel functions,

$$\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \rightarrow K(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

- It essentially amounts to fitting a support vector classifier in an enlarged feature space with nonlinear features.

- When the support vector classifier is combined with a nonlinear kernel, the resulting classifier is known as a support vector machine (SVM).

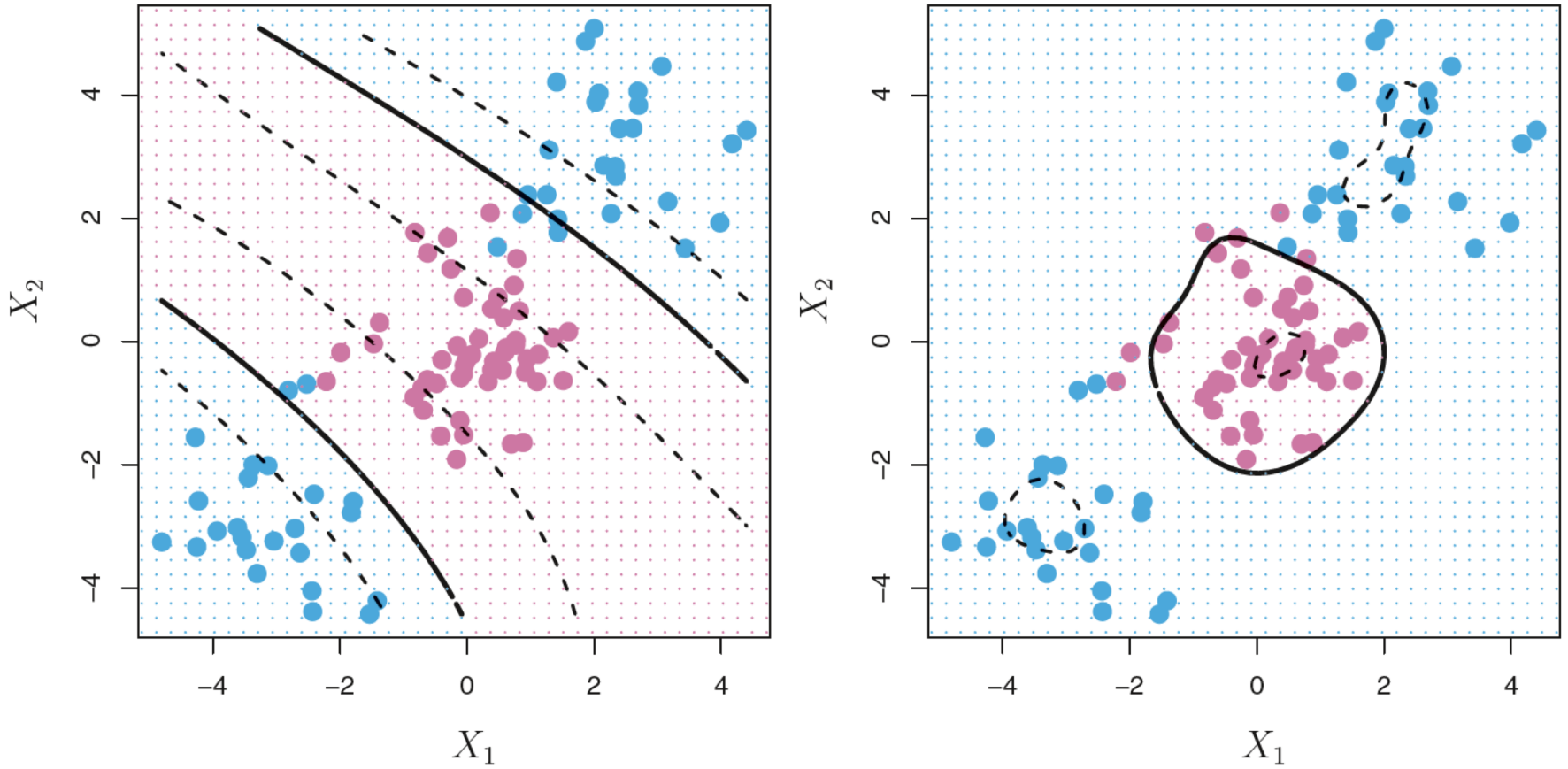
- SVM is an extension of the support vector classifier that results from enlarging the feature space in a specific way, using kernels:

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

Choice of Kernel Function

- Here are some popular choices of kernel functions:
  - Linear Kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$.
  - Polynomial Kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle)^d$, with $d = 2, 3, \ldots$
  - Gaussian Kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma \|x_i - x_j\|_2^2}$, with $\gamma > 0$.
- Using a polynomial kernel amounts to fitting a support vector classifier in a higher-dimensional space involving polynomials of degree d.
- The Gaussian kernel gives weights to the training observations according to their Euclidean distance to the test observation. Hence, the Gaussian kernel has a local behavior, in the sense that only nearby training observations have an effect on the class label of a test observation.

Example: Support Vector Machine



Left: SVM with a poly. kernel of degree 3. Right: SVM with a Gaussian kernel.

- In this example, both polynomial and Gaussian kernels manage to capture the decision boundary.

Kernel versus Enlarged Feature Space

- What is the advantage of using a kernel rather than simply enlarging the feature space using functions of the original features?

1. Computation: The dimensionality of enlarged feature space can be pretty high (e.g. a polynomial function of p). In contrast, kernel expression of SVM is in general more parsimonious as there are only $n + 1$ parameters to estimate.
2. Tractability: The kernel approach only involves compute (functional) inner products over distinct pairs of observations. This can be done without explicitly specifying the enlarged feature space. In many applications of SVMs, the enlarged feature space can be complicated or even implicit.

- For some kernels, such as the Gaussian kernel, the feature space is implicit and infinite-dimensional, so we could never do the computations there anyway!

SVMs with More Than Two Classes

- 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 SVMs to the more general case where we have some arbitrary number of classes?
- It turns out that the concept of separating hyperplanes upon which SVMs are based does not lend itself naturally to more than two classes.
- Though a number of proposals for extending SVMs to the K-class case have been made, the two most popular approaches are
  1. One-versus-one;
  2. One-versus-all.

One-Versus-One Classification

- Suppose we want to perform classification using SVMs, and there are $K > 2$ classes. A one-versus-one or all-pairs approach constructs $\binom{K}{2}$ SVMs, each of which compares a pair of classes.

- For example, one such SVM might compare the k-th class, coded as $+1$, to the l-th class ($k \neq l$), coded as -1.

- We classify a test observation using each of the $\binom{K}{2}$ classifiers, and we tally 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 by these $\binom{K}{2}$ pairwise classifiers.

One-Versus-All Classification

- The one-versus-all approach is an alternative procedure for applying SVMs in the case of $K > 2$ classes.
- We fit K SVMs, each time comparing one of the K classes to the remaining $K - 1$ classes.
- Let $\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$ denote the parameters that result from fitting an SVM comparing the k-th class (coded as +1) to the others (coded as −1).
- Let x^* denote a test observation. We assign the observation to the class for which

$$\beta_{0k} + \beta_{1k}x_i^* + \beta_{2k}x_2^* + \cdots + \beta_{pk}x_p^* \text{ is largest, for } k = 1, \ldots, K.$$

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.

Understanding SVM

- When SVMs were first introduced in the mid-1990s, they made quite a splash in the statistical and machine learning communities. This was due in part to their good performance, good marketing, and also to the fact that the underlying approach seemed both novel and mysterious.

- The idea of finding a hyperplane that separates the data as well as possible, while allowing some violations to this separation, seemed distinctly different from classical approaches for classification, such as logistic regression and linear discriminant analysis.

- Moreover, the idea of using a kernel to expand the feature space in order to accommodate nonlinear class boundaries appeared to be a unique and valuable characteristic.

- For the past two decades, many efforts has been made to find deep connections between SVMs and other more classical statistical methods.

Relationship to Logistic Regression

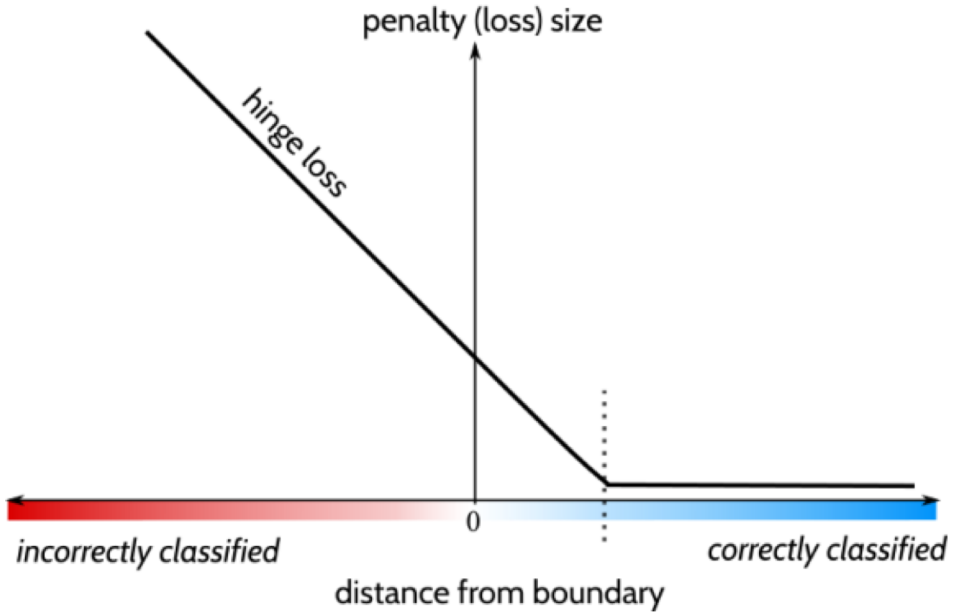
- Recall that the support vector classifier can be obtained by solving

$$\text{minimize}_{\beta_0,\ldots,\beta_p}\left\{\sum_{i=1}^{n}\max[0,1-y_i\left(\beta_0+\beta_1x_{i1}+\cdots+\beta_px_{ip}\right)+\lambda\sum_{j=1}^{p}\beta_j^2\right\}.$$

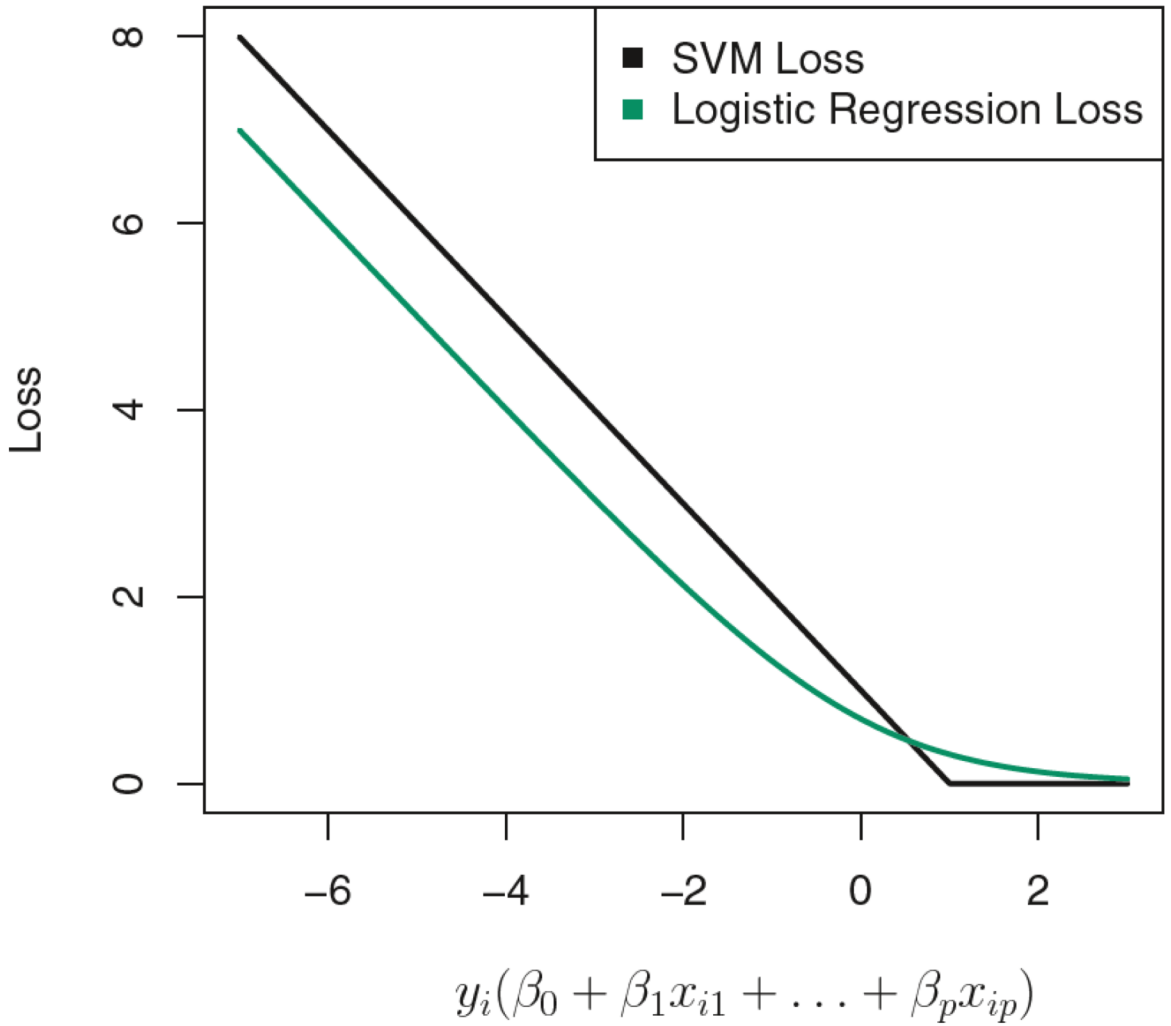
- The hinge loss function is

$$L(\boldsymbol{X},\boldsymbol{y},\boldsymbol{\beta})=\sum_{i=1}^{n}\max\{0,1-y_i\left(\beta_0+\beta_1x_{i1}+\cdots+\beta_px_{ip}\right)\}.$$

- It turns out that the hinge loss function is closely related to the loss function used in logistic regression.



Example: SVM vs Logistic



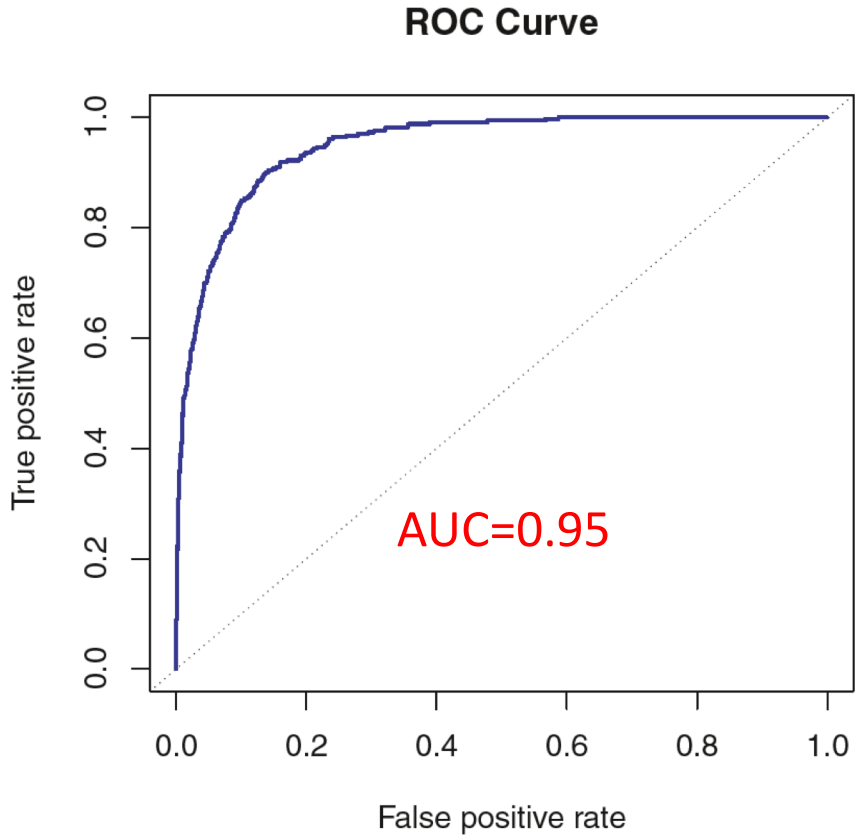
- Overall, the two loss functions have quite similar behavior. Hinge loss decays to exactly 0 when $y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq 1$. In contract, the loss of logistic regression is close to but not exactly 0.

- For SVM, only support vectors play a role in the classifier obtained. Due to the similarity of two losses, SVM and logistic regression often give very similar results.

- When the classes are well separated, SVMs tend to behave better than logistic regression; in more overlapping regimes, logistic regression is often preferred.

ROC Curve

- The ROC curve is a popular graphical tool for comparing the performance of multiple classifiers. The name "ROC" is an acronym for receiver operating characteristics which comes from communications theory.



- The ROC curve summarizes two types of error.
  1. True positive rate (sensitivity) : the fraction of observations in default class that are correctly identified.
  2. False positive rate (1 – specificity): the fraction of observations in non-default class that we classify incorrectly to default class.

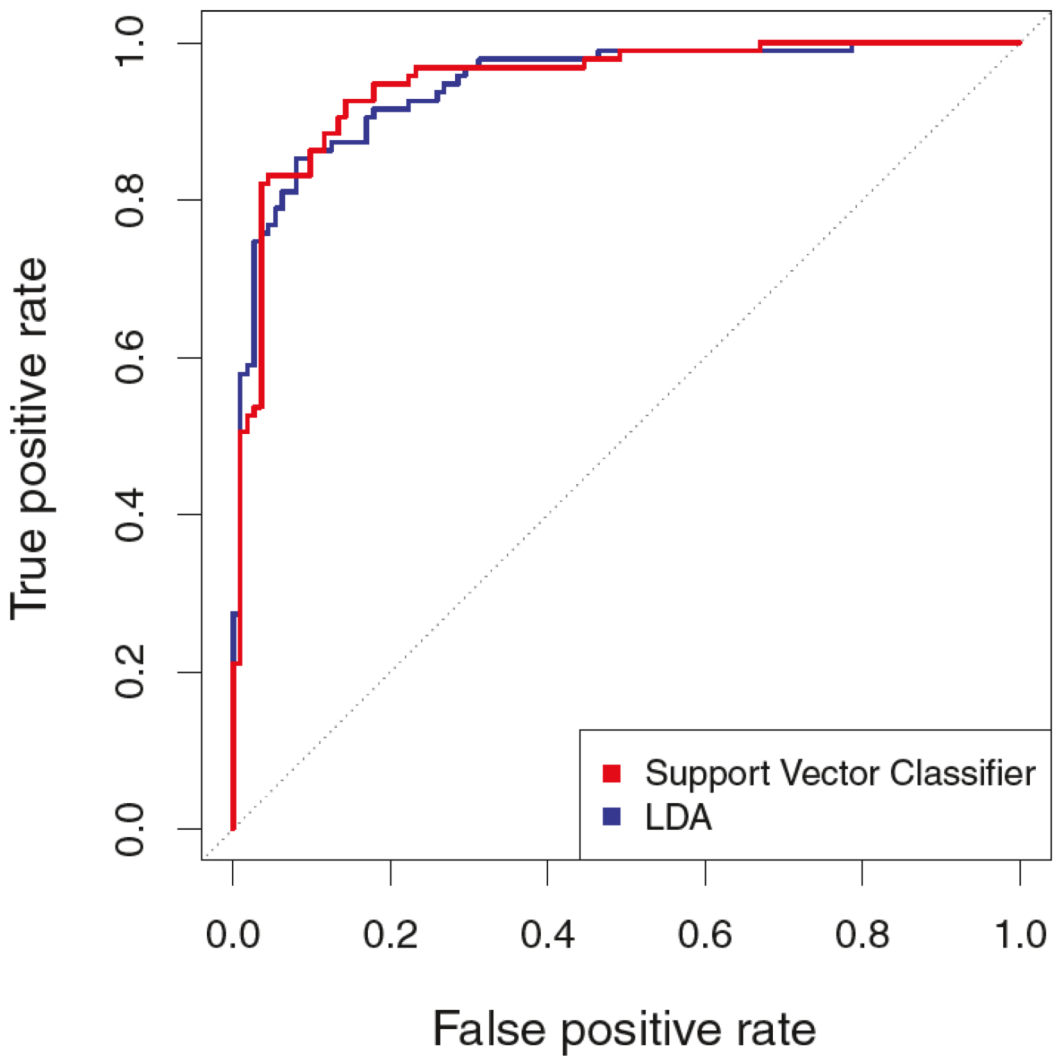
- The ideal ROC curve hugs the top left corner, indicating a high true positive rate and a low false positive rate.

- The overall performance of a classifier is given by the area under the (ROC) curve (AUC). So the larger the AUC the better the classifier.

Example: Heart Disease Dataset

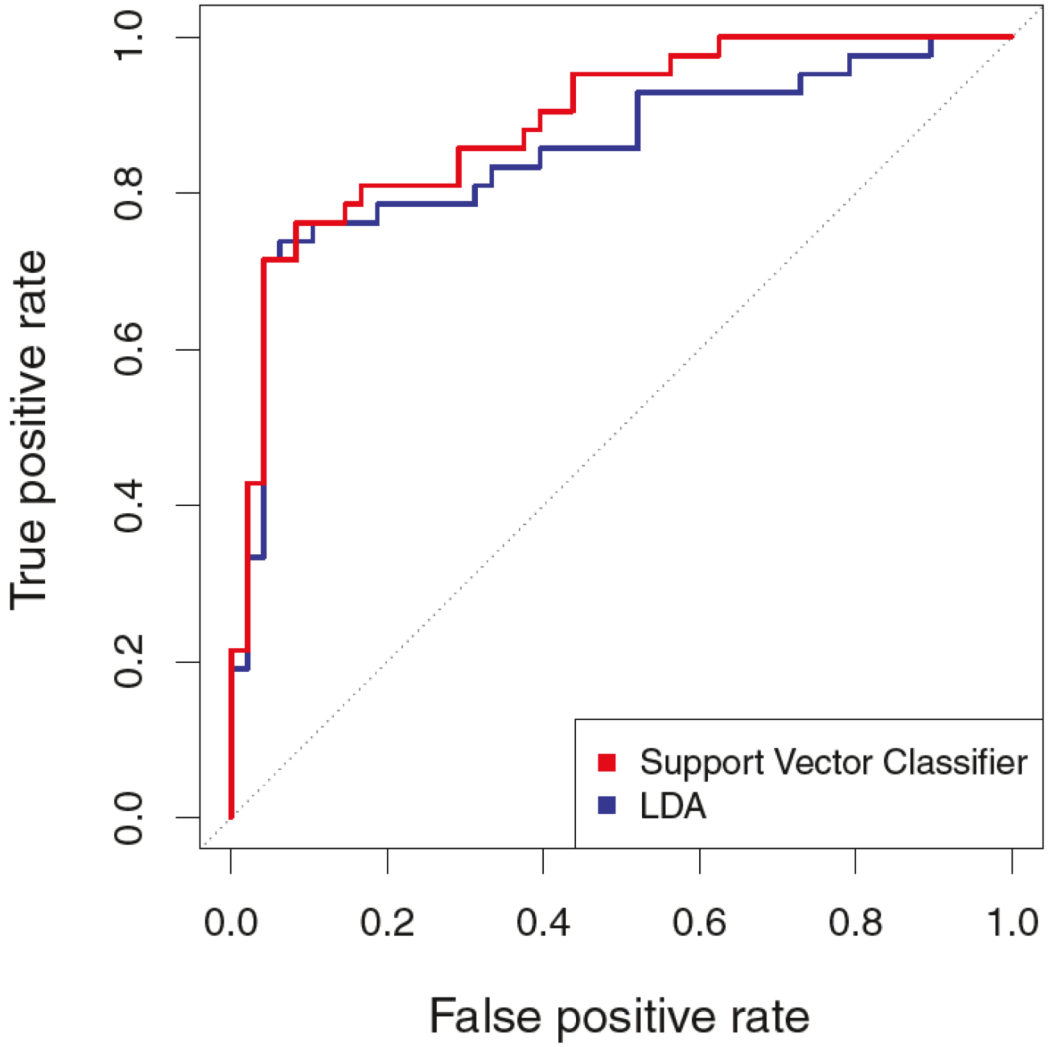
- This dataset collected the heart disease diagnostic results of **297 people**. Among them, 139 are diagnosed with some heart disease ($y_i = 1$) while the rest are not ($y_i = 0$).
- The dataset contains **13 predictors** including the age, sex, resting blood pressure, fasting blood sugar, chest pain type and so on.
- We randomly split the dataset into a training set of size 207 and a test set of size 90.
- We fit LDA, support vector classifier and SVMs on the training set. The classification results are compared on the test set.
- The classification performance is evaluated through the ROC curves for the test set.

Example: Support Vector Classifier vs LDA (Train)



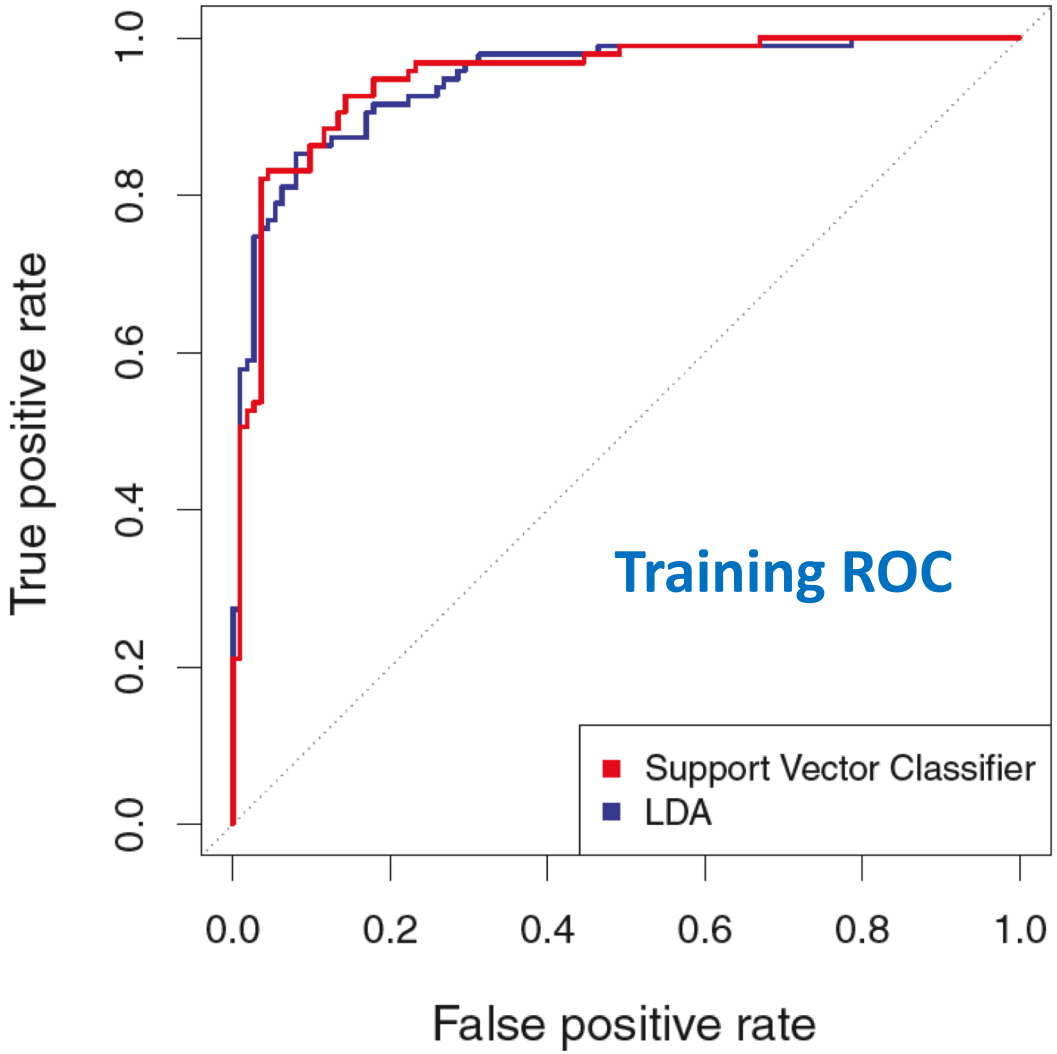
- We first fit linear discriminant analysis (LDA) and the support vector classifier to the training data.
- Note that the support vector classifier is an SVM using a polynomial kernel of degree $d = 1$.
- Both classifiers compute scores $\hat{f}(\boldsymbol{x}) = \boldsymbol{x}'\hat{\boldsymbol{\beta}}$. For any given cutoff t, we classify $\boldsymbol{x}$ into class 1 or class 0 depending on whether $\hat{f}(\boldsymbol{x}) > t$ or $\hat{f}(\boldsymbol{x}) < t$.
- Both LDA and the support vector classifier perform well, though there is some evidence that the support vector classifier may be slightly superior.

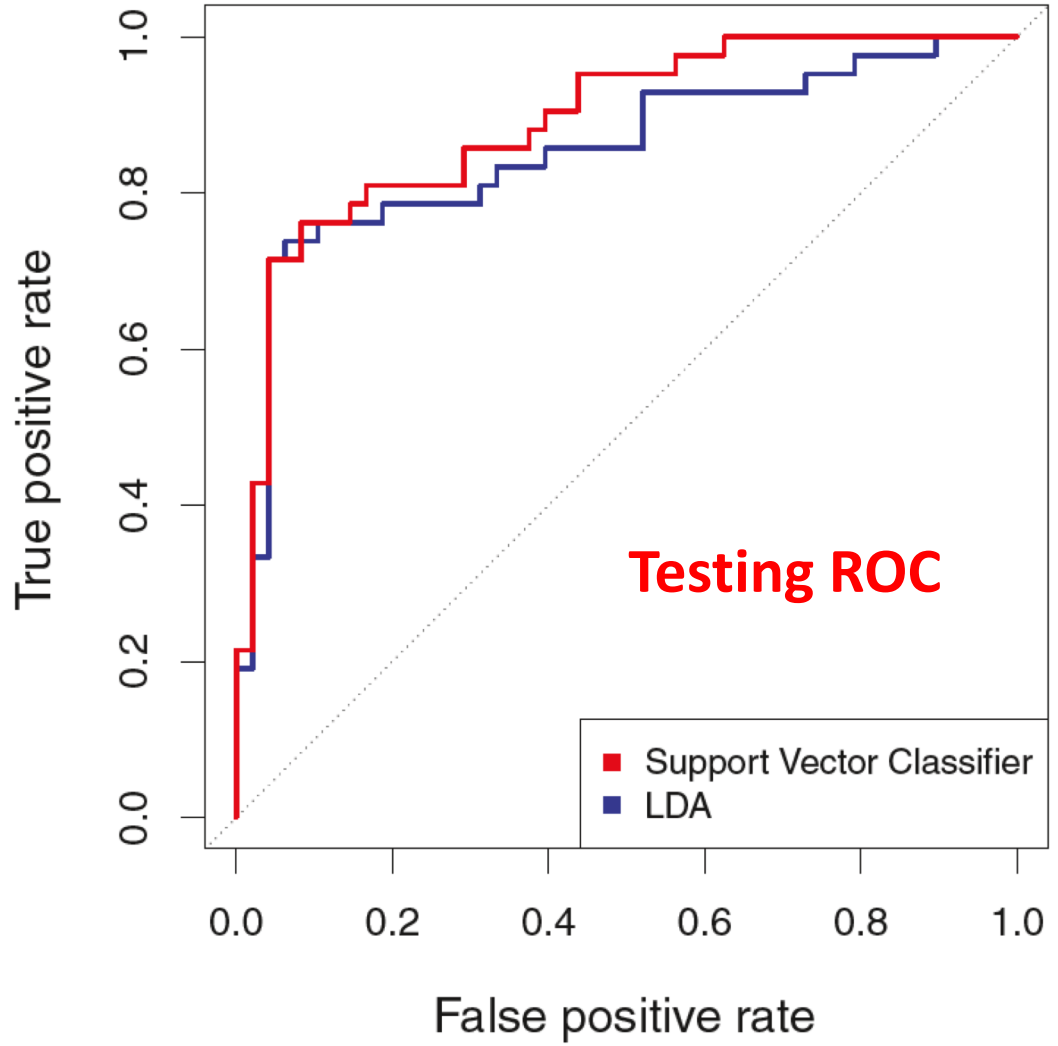
Example: Support Vector Classifier vs LDA (Test)



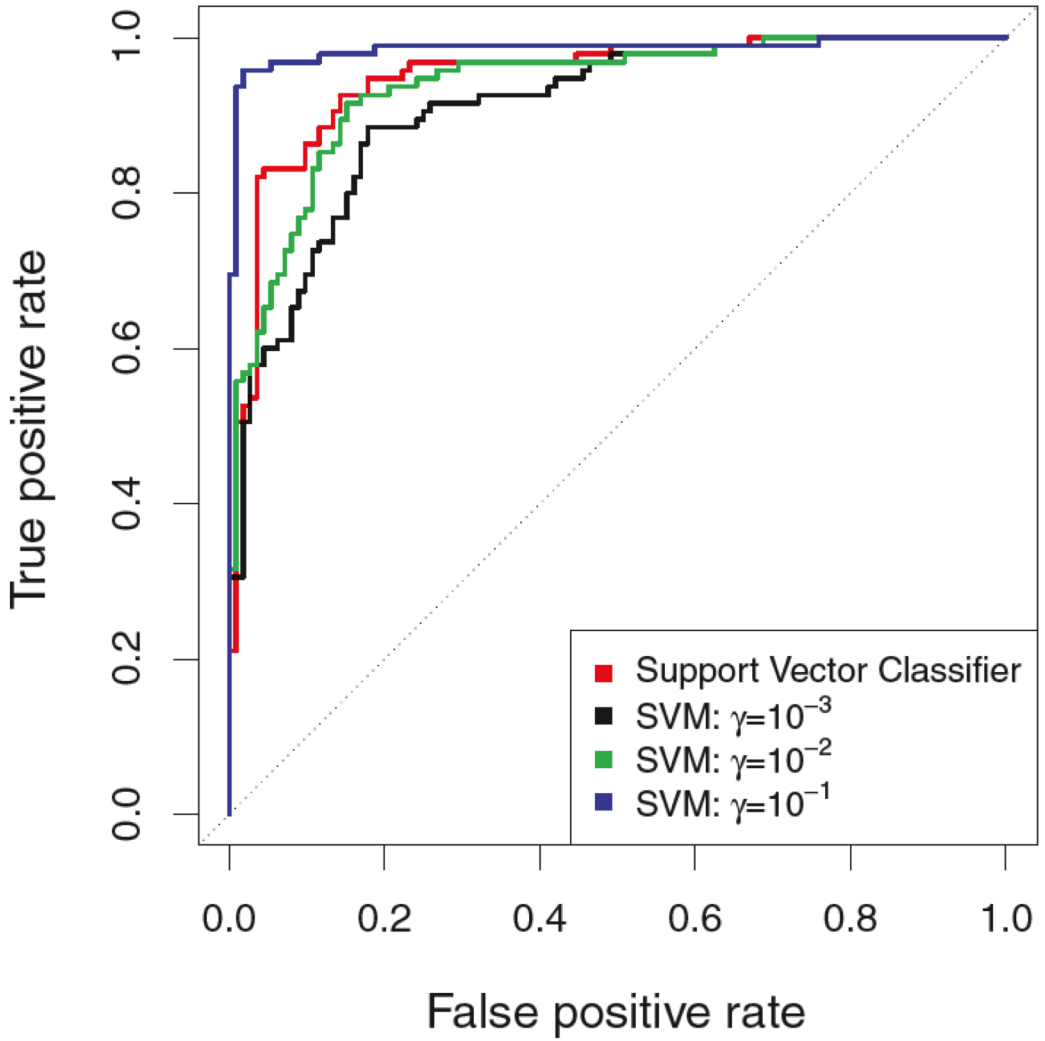
- Now, we compare the prediction performance in the test set.
- We observe some differences from the training ROC curves.
- Both methods have a smaller AUC compared with training ROC curves.
- Again, the support vector classifier appears to have a small advantage over LDA (although these differences are not statistically significant).

Example: Support Vector Classifier vs LDA



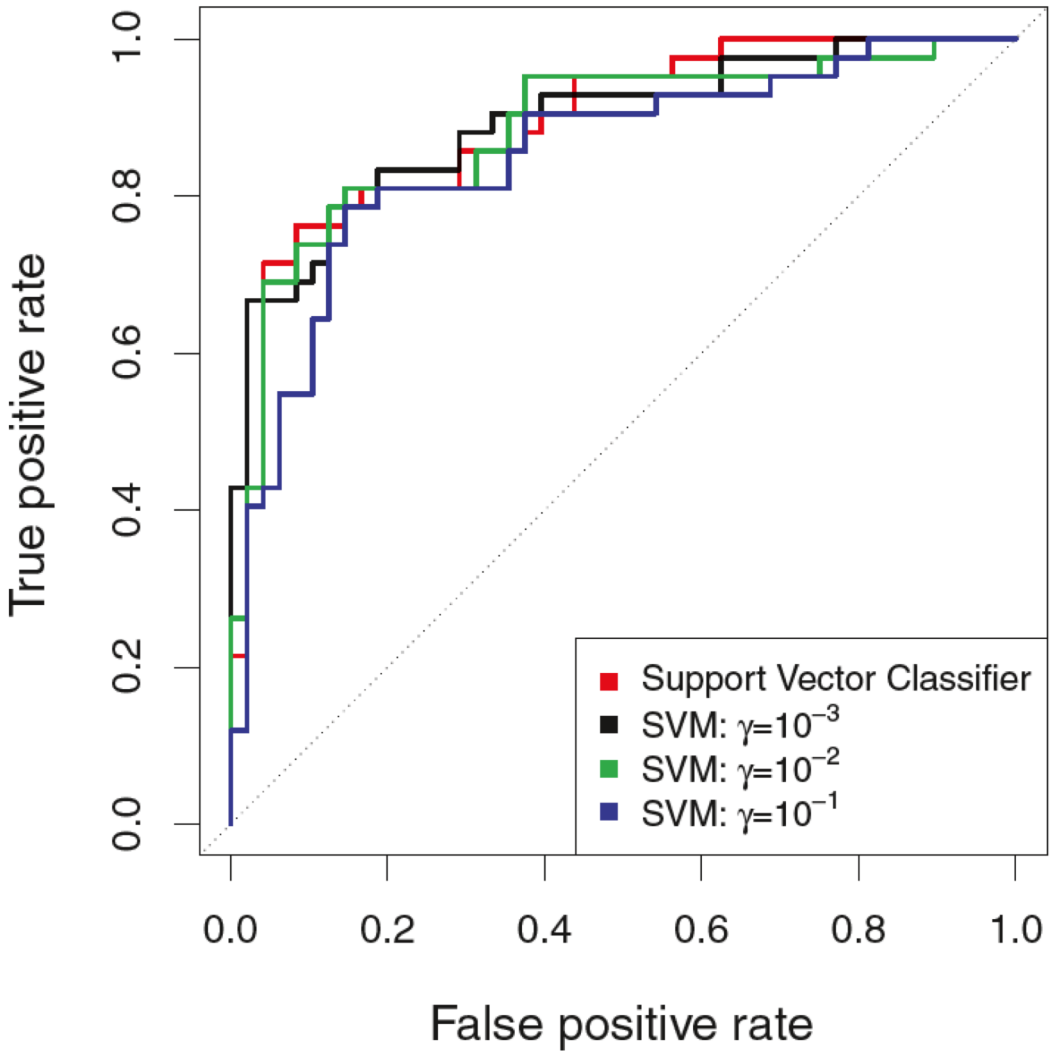


Example: Support Vector Classifier vs SVM (Train)



- Now, we compare support vector classifier with SVMs using a Gaussian kernel. We set the parameter γ in Gaussian kernel to be 10^{-1}, 10^{-2} and 10^{-3}, respectively.
- As γ increases and the fit becomes more nonlinear, 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.

Example: Support Vector Classifier vs SVM (Test)



- Now, we compare the prediction performance in the test set.
- The SVM using $\gamma = 10^{-1}$, which showed the best results on the training data, produces the worst estimates on the test data.
- The SVMs with $\gamma = 10^{-2}$ and 10^{-3} perform comparably to the support vector classifier, and all three outperform the SVM with $\gamma = 10^{-1}$.

Example: Support Vector Classifier vs SVM