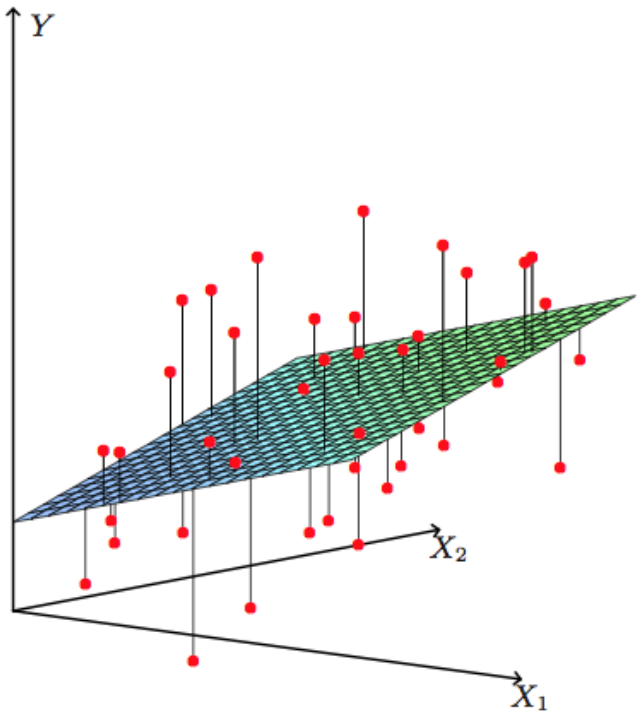
Data Mining and Machine Learning

Kuangnan Fang

Department of Statistics, Xiamen University
Email: xmufkn@xmu.edu.cn



- Vladimir Vapnik was born in the Soviet Union. He received his master's degree in mathematics from the Uzbek State University in 1958 and Ph.D in statistics at the Institute of Control Sciences, Moscow in 1964.
- He worked at this institute from 1961 to 1990 and became Head of the Computer Science Research Department.
- At the end of 1990, Vladimir Vapnik moved to the USA and joined in AT&T Bell Labs. While at AT&T, Vapnik developed the theory of the support vector machine.
- Vapnik left AT&T in 2002 and joined NEC Laboratories in Princeton.Fellow of the U.S. National Academy of Engineering in 2006. In 2014, Vapnik joined Facebook AI Research.

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.*

maximal margin classifer

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.

support vector classifier

support vector machine

一般把Mximal margin classifier, support vectro classifier, support vector machine 统称为 SVM

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

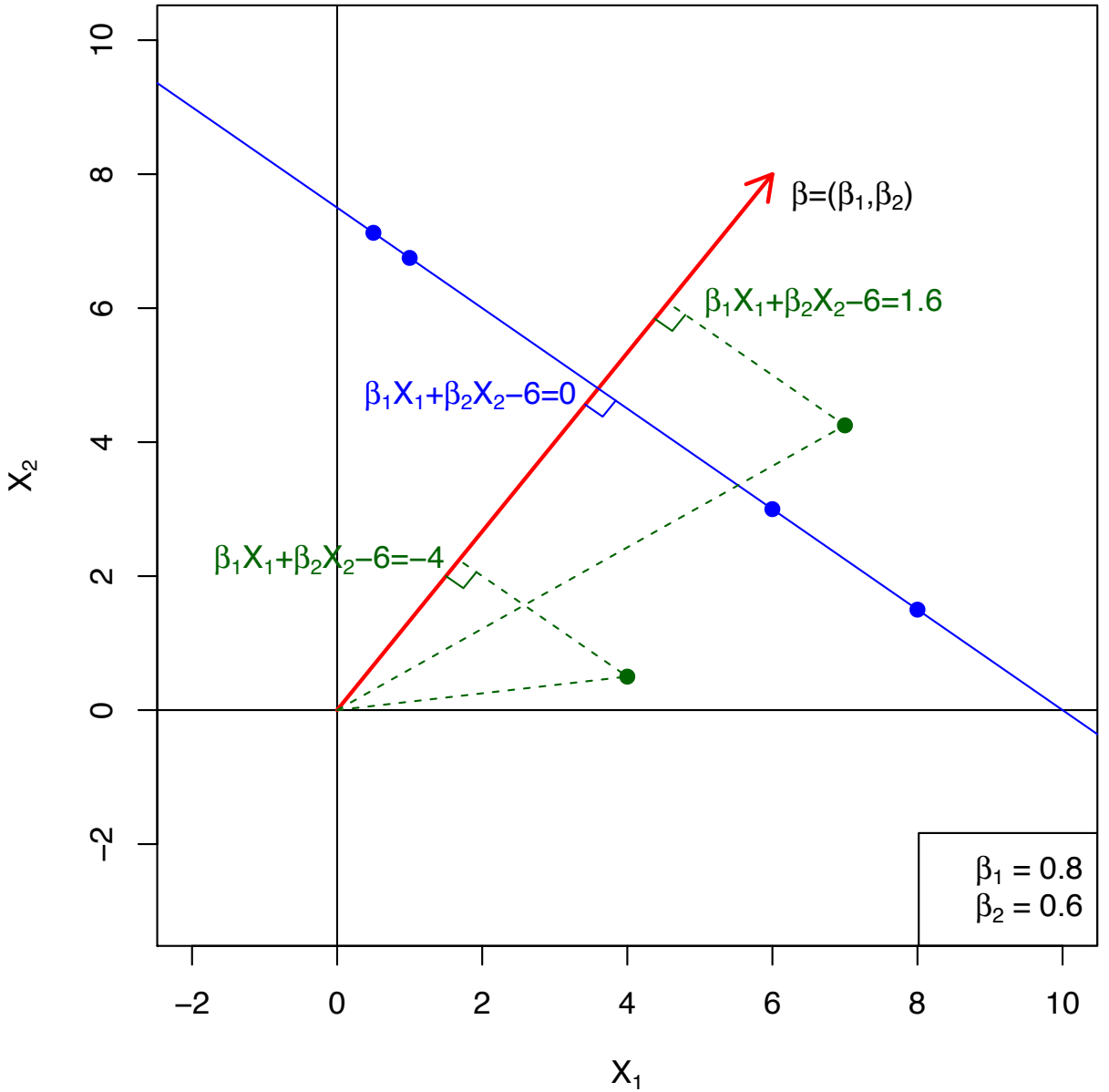
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$

b0+b1X1+b2X2=0 (X1,X2)

- In $p = 2$ dimensions a hyperplane is a line. P=3?
- 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.

法向量

Hyperplane in 2 Dimensions



Hyperplane divid p-dimensional space into two halves

Now, suppose that X does not satisfy (9.2); rather,

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p > 0 \qquad (9.2)$$

Then this tells us that X lies to one side of the hyperplane. On the other hand, if

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p < 0 \qquad (9.4)$$

then 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 the left hand side of (9.2). A hyperplane in two-dimensional space is shown in Figure 9.1.

A hyperplane in two-dimensional space

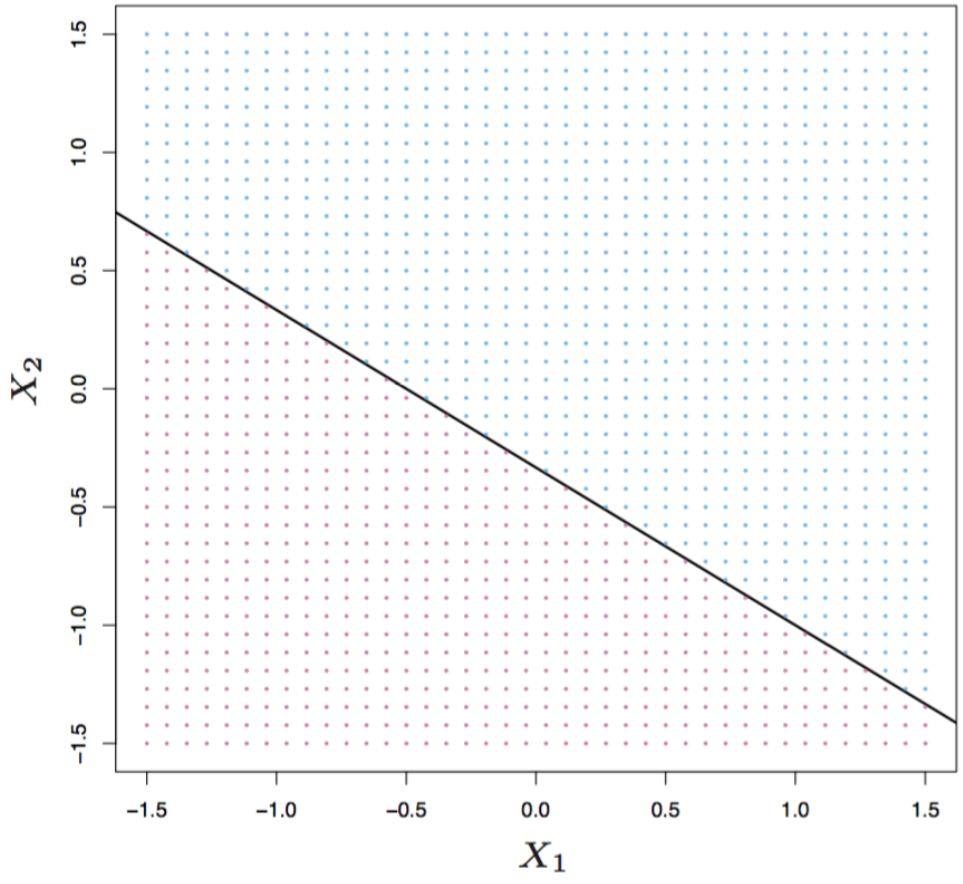


FIGURE 9.1. *The hyperplane* $1 + 2X_1 + 3X_2 = 0$ *is shown. The blue region is the set of points for which* $1 + 2X_1 + 3X_2 > 0$*, and the purple region is the set of points for which* $1 + 2X_1 + 3X_2 < 0$*.*

Classification Using a Separating Hyperplane

Now suppose that we have a $n \times p$ data matrix X that consists of n training observations in p -dimensional space,

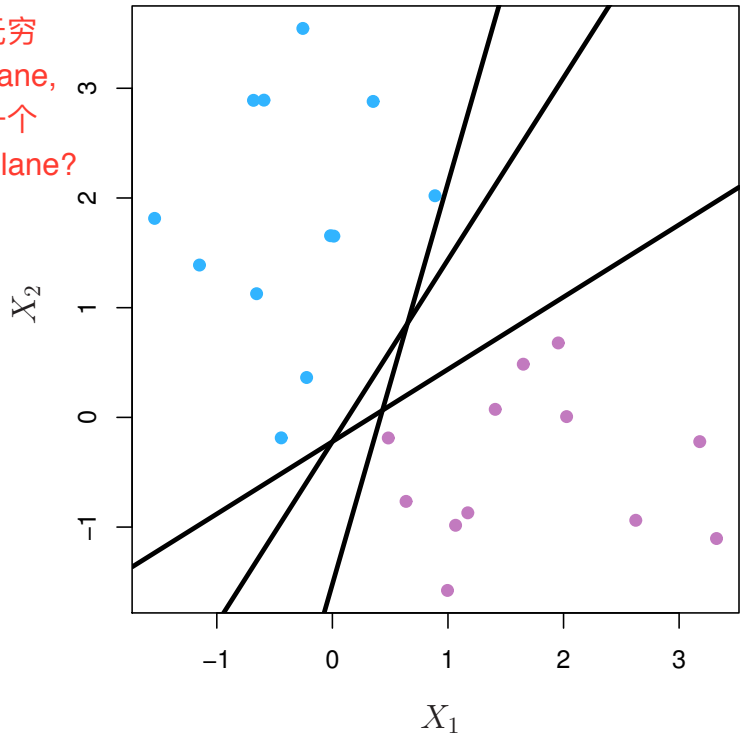
$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix} \tag{9.5}$$

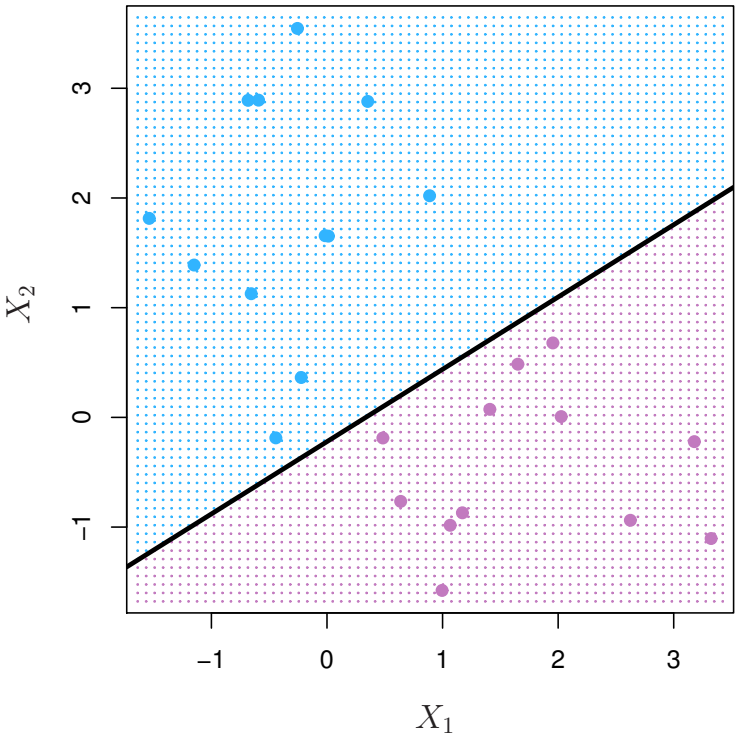
and that these observations fall into two classes – that is, $y_1, \cdots, y_n \in \{-1, 1\}$ where -1 represents one class and 1 the other class. We also have a test observation, a p-vector of observed features $x^* = (x_1^*, \cdots, x_p^*)^T$. Our goal is to develop a classifier based on the training data that will correctly classify the test observation using its feature measurements.

Separating Hyperplanes

magnitude of f(x∗):
(1).f(x∗) 离0越远，对分类结果越肯定
(2).f(x∗) 离0越近，对分类结果越不肯定

理论上有无穷多个hyperplane，如何确定一个最优的hyperplane?





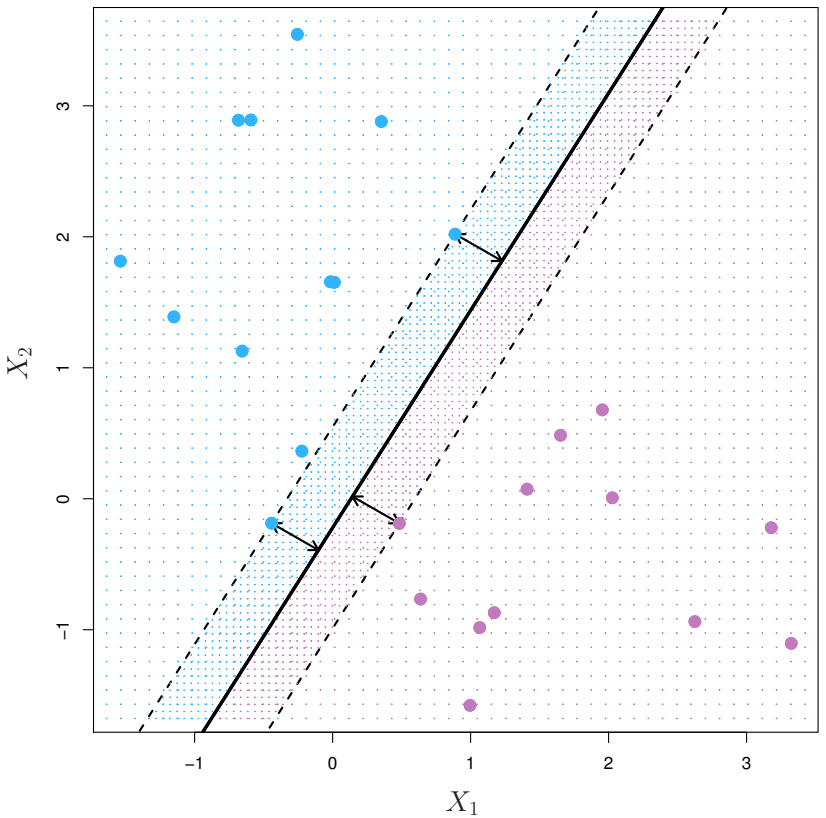
- If $f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$, then $f(X) > 0$ for points on one side of the hyperplane, and $f(X) < 0$ for points on the other.
- If we code the colored points as $Y_i = +1$ for blue, say, and $Y_i = -1$ for mauve, then if $Y_i \cdot f(X_i) > 0$ for all i, $f(X) = 0$ defines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

test: f(x∗) = β0+β1x∗1+β2x∗2+. . .+βpx∗p

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

Maximal Margin Classifier

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



Constrained optimization problem

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

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

唯一性

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

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

实际中，只要yi(b0+..+bpxip)>=0即可，M>0要求更严格

Margin? 样本点离超平面最近的距离

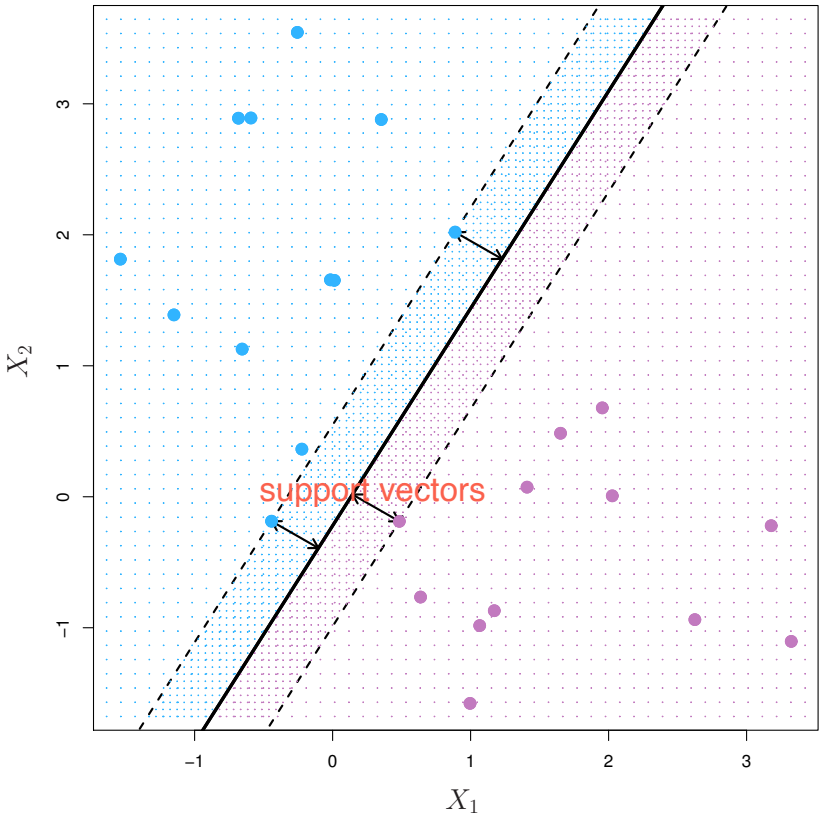
Maximal Margin hyperplane：分得最开的超平面

Maximal Margin classifier:基于 maximal margin hyperplane的分类

Support Vector? 虚线上的点，每个都是p维的vector, 用来support超平面，点细微的移动，会改变超平面的位置！
超平面只依赖于少数的support vector,而跟其他点无关！

Maximal Margin Classifier

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



Constrained optimization problem

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

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

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

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

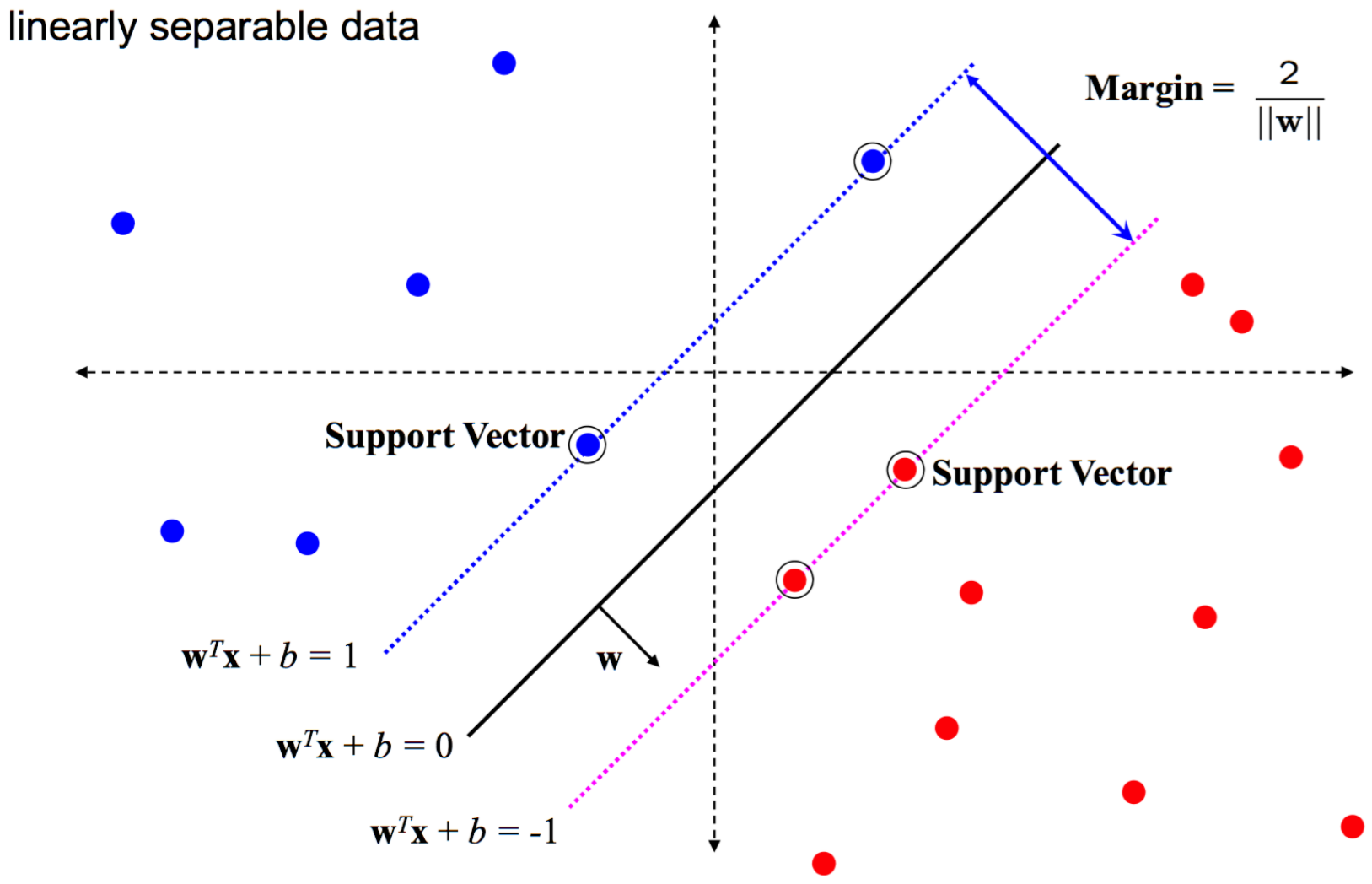
This can be rephrased as a convex quadratic program, and solved efficiently. The function `svm()` in package `e1071` solves this problem efficiently

Maximal margin classifier

- since $\beta^\top \mathbf{x} + \beta_0 = 0$ and $c\left(\beta^\top \mathbf{x} + \beta_0\right) = 0$ define the same plane, we have the freedom to choose the normalization of $\mathbf{w}$
- Choose normalization such that $\beta^\top \mathbf{x}_+ + \beta_0 = +1$ and $\beta^\top \mathbf{x}_- + \beta_0 = -1$ for the positive and negative support vectors respectively
- Then the margin is given by

$$\frac{\beta}{\|\beta\|} \cdot (\mathbf{x}_+ - \mathbf{x}_-) = \frac{\beta^\top (\mathbf{x}_+ - \mathbf{x}_-)}{\|\beta\|} = \frac{2}{\|\beta\|}$$

Maximal margin classifier



Maximal margin classifier

- Learning the SVM can be formulated as an optimization:

$$\max_{\beta} \frac{2}{\|\beta\|} \text{ subject to } \beta^T x_i + \beta_0 \begin{array}{l} \geq +1 \text{ if } y_i = 1 \\ \leq -1 \text{ if } y_i = -1 \end{array} \text{ for } \quad i = 1, \cdots, N$$

- Or equivalently

$$\min_{\beta} \|\beta\|^2 \text{ subject to } y_i \left(\beta^\top \mathbf{x}_i + \beta_0\right) \geq 1 \text{ for } i = 1 \ldots N$$

- This is a quadratic optimization problem subject to linear constraints and there is a unique minimum

Constrained Optimization

- Constrained optimization has the form

$$\begin{aligned} \min \quad & Q(\theta) \\ \text{subject to} \quad & \theta \in \mathcal{S} \subset \mathbb{R}^d \end{aligned}$$

- $Q(\theta)$: objective function
- $\mathcal{S}$: feasible set
- Convex optimization: both objective function and feasible set are convex.

Lagrange Multiplier

- Consider equality constraint

$$\begin{aligned} \min \quad & Q(\theta) \\ \text{subject to} \quad & R(\theta) = 0 \end{aligned}$$

- $\mathcal{S} = \{\theta : R(\theta) = 0\}$ is a$(d-1)$ dimensional surface in R^d.
- For every θ such that $R(\theta) = 0$, $\bigtriangledown R(\theta)$ is orthogonal to the surface.
- If θ^* is a local minimum, then $\bigtriangledown Q$ is orthogonal to the surface at θ^*.

Lagrange Multiplier

- Conclusion: at a local minimum, there exists $\lambda \in \mathbb{R}$ such that
$$\nabla Q\left(\theta^*\right) = \lambda \nabla R\left(\theta^*\right)$$
- This leads us to introduce the Lagrangian
$$L(\theta, \lambda) = Q(\theta) - \lambda R(\theta)$$
where λ is the Lagrange multiplier.
- We have argued that a local minimum corresponds to a stationary point of the Lagrangian.
Furthermore, we can reverse our logic to deduce that a stationary point of the Lagrangian is a local optimum.

Lagrange Multiplier

Inequality constraint

Now consider(the primal problem)

$$\begin{aligned} \min \quad & Q(\theta) \\ \text{subject to} \quad & R(\theta) \geq 0 \end{aligned}$$

Suppose θ^* is a local minimum. There are two cases:

- Inactive constraint: $R(\theta^*) > 0 \Rightarrow \nabla Q(\theta^*) = 0 \Rightarrow$ stationary point of $L(\theta, \lambda)$ with $\lambda = 0$
- Active constraint: $R(\theta^*) = 0 \Rightarrow$ same as equality constraint except we require $\lambda > 0$.

Lagrange Multiplier

In either case, we have $\lambda \cdot R(\theta^*) = 0$. Therefore, a local minimum satisfies (Karush -Kuhn-Tucker conditions)

$$\begin{aligned} \nabla L(\theta^*) &= \nabla Q(\theta^*) - \lambda \nabla R(\theta^*) = 0 \\ \lambda R(\theta^*) &= 0 \\ \lambda &\geq 0 \end{aligned}$$

- Often the KKT conditions may be used to transform the primal problem to an equivalent dual problem, where the variables being optimized are the Lagrange multipliers.

Quadratic Programming

Equivalently

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

$$\text{subject to } y_i\left(\beta_0 + x_i^\top \beta\right) \geq 1, i = 1, \ldots, n$$

The Lagrange primal is

$$L_p = \frac{1}{2}\|\beta\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i\left(\beta_0 + x_i^\top \beta\right) - 1\right]$$

where $\alpha_i \geq 0$.

Quadratic Programming

- Dual problem:

$$\max_{\alpha} \min_{\beta_0, \beta} L_p$$

- Firstly,

$$\min_{\beta_0, \beta} L_p$$

- Setting the derivatives to zero, we get

$$\frac{\partial}{\partial \beta} : \beta = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\frac{\partial}{\partial \beta_0} : 0 = \sum_{i=1}^{n} \alpha_i y_i$$

Quadratic Programming

Substituting into the Lagrange primal, we obtain the Lagrange dual

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} y_i y_{i'} x_i^\top x_{i'}$$

We maximize L_D subject to

$$\alpha_i \geq 0$$

and

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

Quadratic Programming

- Let $\alpha^\star = (\alpha_1^\star, \ldots, \alpha_n^\star)$ is the solution of dual problem, then we can get the solution of the primal problem by

$$\begin{cases} \beta^\star = \sum_{i=1}^{n} \alpha_i^\star y_i x_i \\ \beta_0^\star = y_j - \sum_{i=1}^{n} \alpha_i^\star y_i (x_i \cdot x_j), \forall \alpha_j > 0 \end{cases}$$

- Show the above solution. KKT condition
- Is the solution β_0 unique?
- Hyperplane: $\beta_0^\star + \beta^\star X = 0$
- Classification dexision function: $f(x) = sign(\beta_0^\star + \beta^\star X)$

Quadratic Programming

- Minimize L_p with respect to primal variables β_0,β
- Maximize L_D with respect to dual variables α_i
- Maximizing the dual is often a simpler convex QP than the primal.

Support Vectors

The Karush-Kuhn-Tucker conditions include

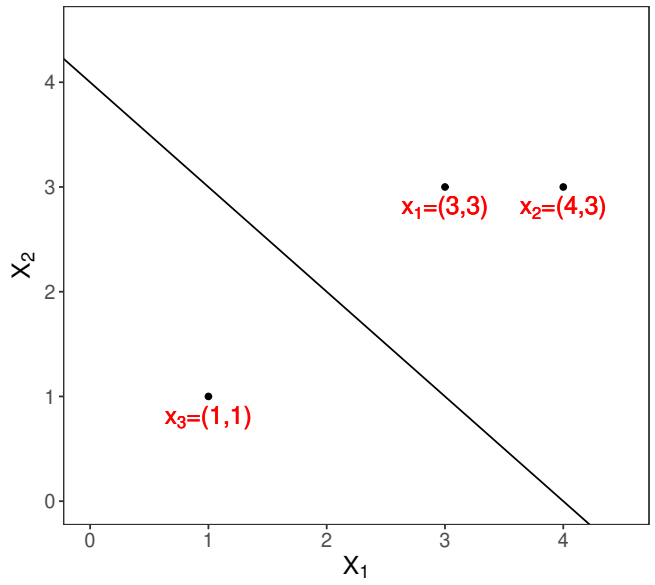
$$\hat{\alpha}_i \left[y_i \left(\hat{\beta}_0 + x_i^\top \hat{\beta} \right) - 1 \right] = 0$$

This imply

- If $y_i \hat{f}(x_i) > 1$, then $\hat{\alpha}_i = 0$, the point outside the margin boundary.
- If $\hat{\alpha}_i > 0$, then $y_i \hat{f}(x_i) = 1$, or in other words, $\beta_0 + \beta x_i = \pm 1$,$x_i$ is on the boundary of the "slab". support vectors.
- The solution $\hat{\beta} = \sum_{i=1}^{n} \alpha_i y_i x_i$ is defined in terms of a linear combination of the support points.

Example

Example: $x_1 = (3, 3)$, $y_1 = 1$, $x_2 = (4, 3)$, $y_2 = 1$, $x_3 = (1, 1)$, $y_3 = -1$.



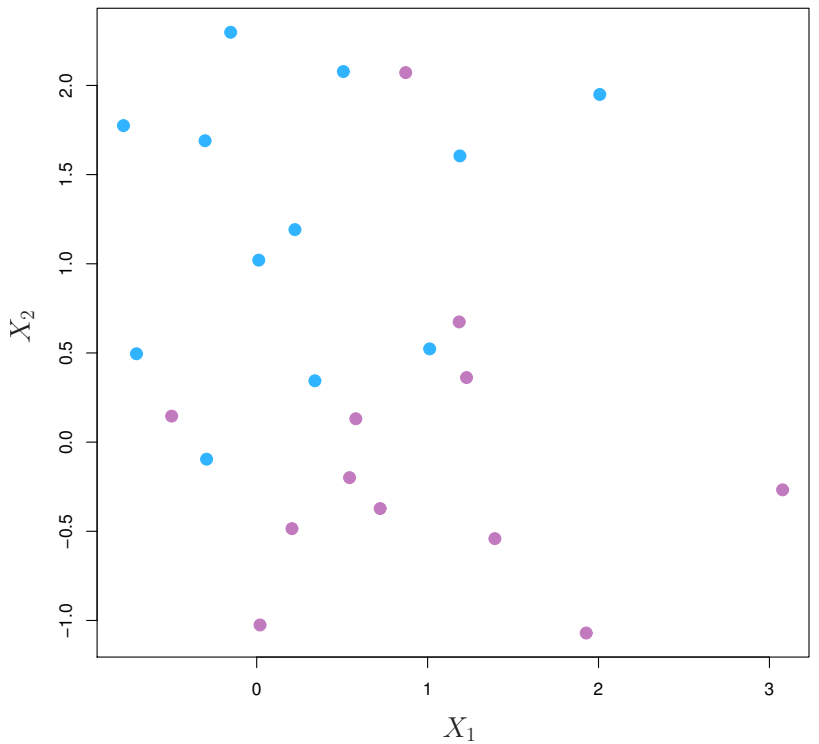
- Primal problem

$$\min_{\beta_0,\beta} \frac{1}{2}\left(\beta_1^2 + \beta_2^2\right) \quad \text{subject to} \begin{cases} 3\beta_1 + 3\beta_2 + \beta_0 \geq 1 \\ 4\beta_1 + 3\beta_2 + \beta_0 \geq 1 \\ -\beta_1 - \beta_2 - \beta_0 \geq 1 \end{cases}$$

  Solution: $\beta_1 = \beta_2 = \frac{1}{2}$, $\beta_0 = -1$.
  Hyperplane: $\frac{1}{2}X_1 + \frac{1}{2}X_2 - 2 = 0$.

- Dual problem ?

Non-separable Data



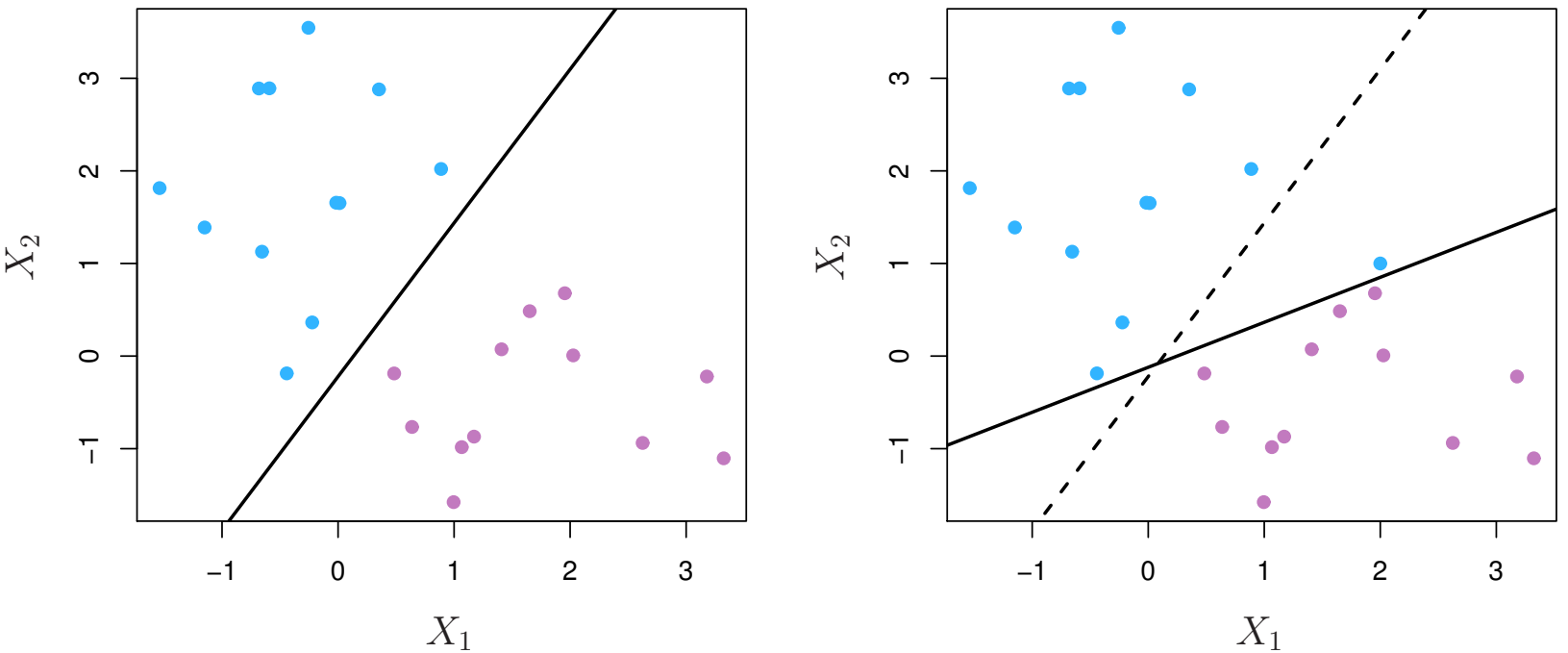
The data on the left are not separable by a linear boundary.

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

overlap

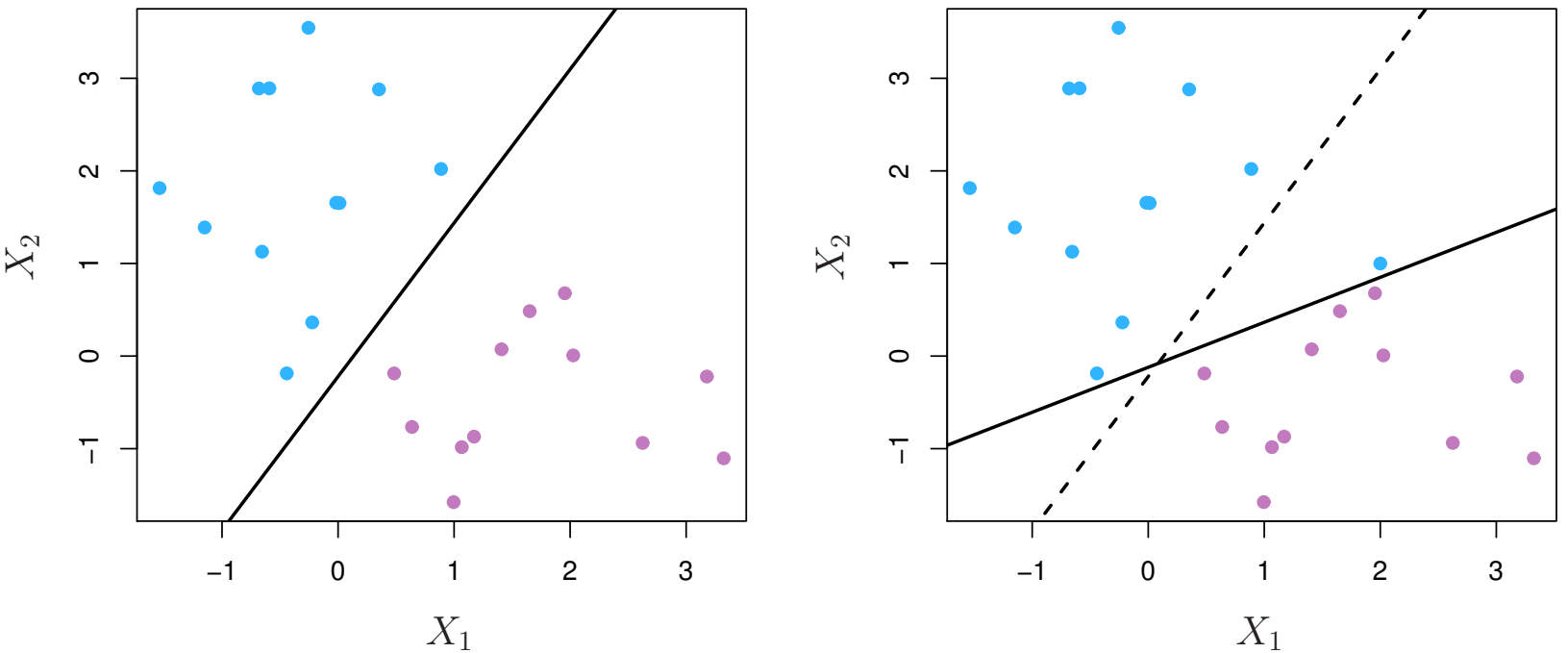
则找不到 maximal margin classifier 怎么办?

Noisy Data



Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier.

Noisy Data



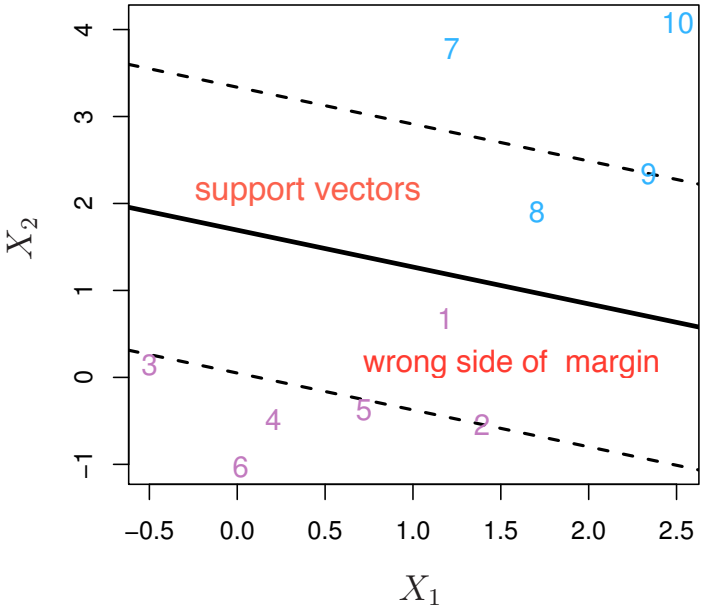
Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier.

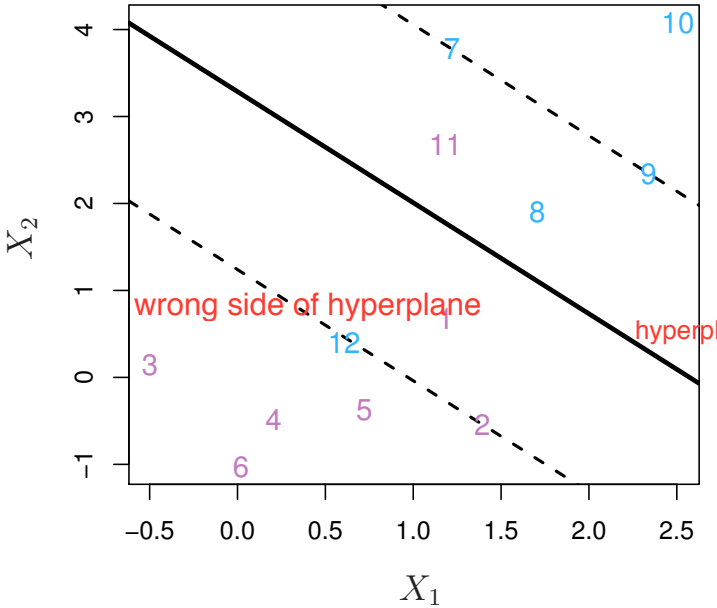
对于个别点过于敏感，容易出现overfitting　不要求完美区分两类，允许部分点分错，但是对其他的点分得更好，而且更加稳健

The *support vector classifier* maximizes a *soft* margin.

soft margin classifier

Support Vector Classifier





support vectors:observations that lie directly on the margin, or on the wrong side of margin

hyperplane只受support vector的影响，不受其他点的影响

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

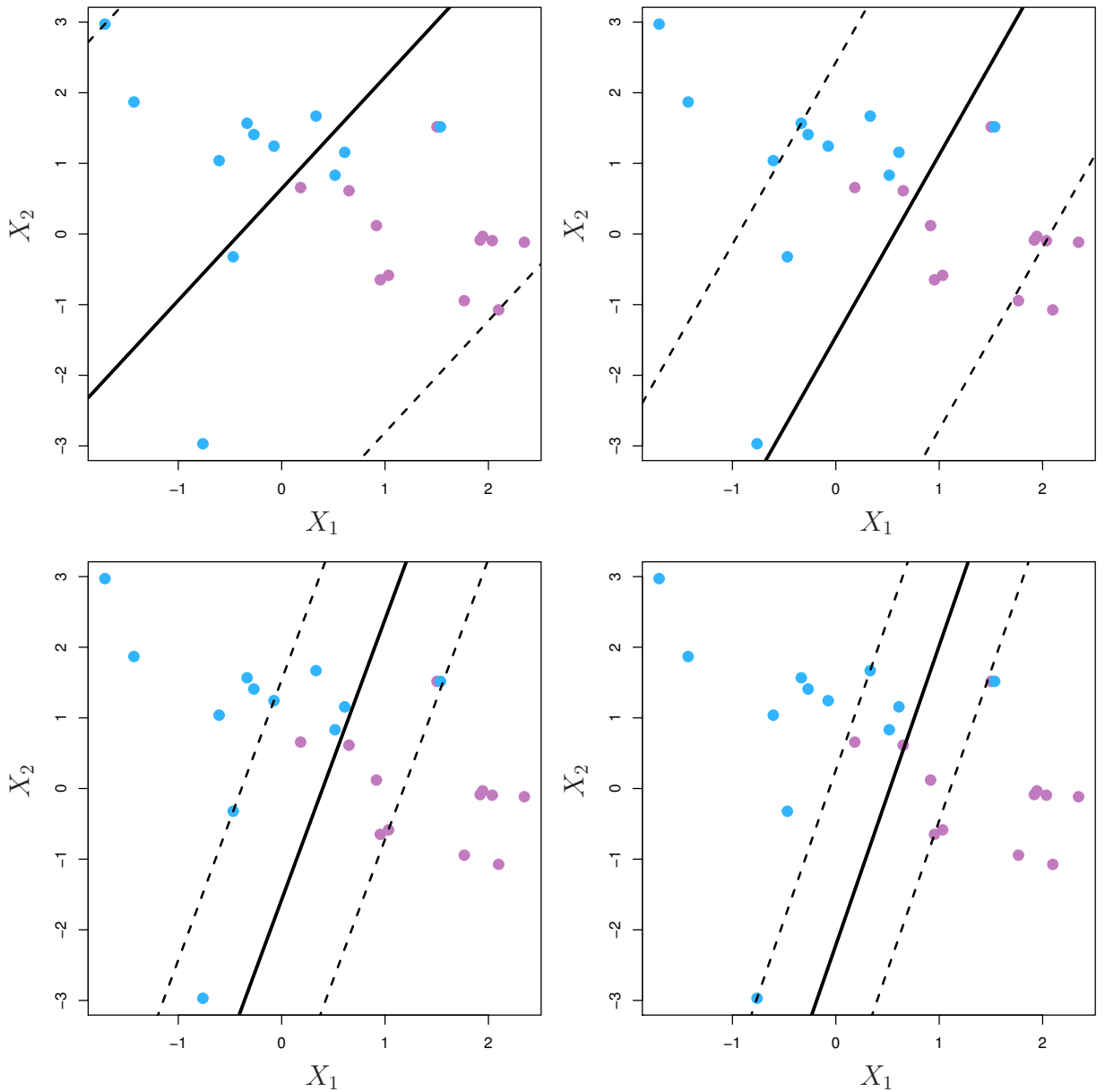
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

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

最多不超过C个分错，因为\epsilon>1分错

budge
tunning parameter choosing by CV approach
C=0, \epsilon =0, maximal margin hyperplane
\epsilon >0, on the wrong margin
\epsilon >1, on the wrong hyperplane

C is a regularization parameter



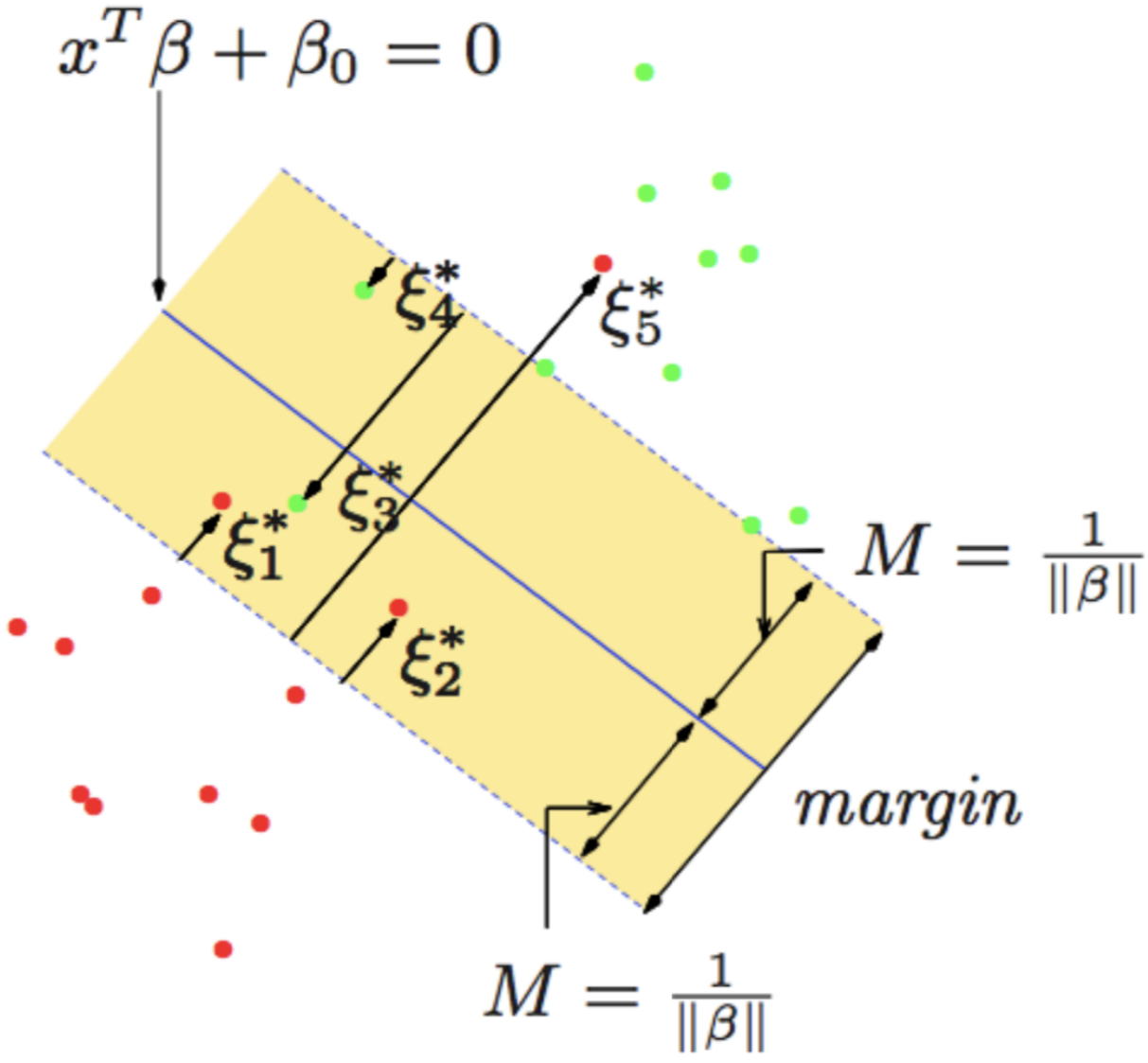
Computing the Support Vector Classifier

we can drop the norm constraint on β, define $M = \frac{1}{\|\beta\|}$ the equivalent form

$$\min \|\beta\|^2 \quad \text{subject to} \quad \begin{cases} y_i \left(x_i^T \beta + \beta_0\right) \geq 1 - \xi_i, \forall i \\ \xi_i \geq 0, \sum \xi_i \leq \text{ constant} \end{cases} \tag{1}$$

- This is the usual way the support vector classifier is defined for the non-separable case.
- points well inside their class boundary do not play a big role in shaping the boundary.

Support Vector Classifier



Computing the Support Vector Classifier

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i \tag{12.8}$$
$$\text{subject to} \quad \xi_i \geq 0, y_i\left(x_i^T\beta + \beta_0\right) \geq 1 - \xi_i \forall i$$

where the "cost" parameter C replaces the constant in (12.7); the separable case corresponds to $C = \infty$.
The Lagrange (primal) function is

$$L_p = \frac{1}{2}\|\beta\| + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i\left(x_i^T\beta + \beta_0\right) - (1-\xi_i)\right] - \sum_{i=1}^{N}\mu_i\xi_i \tag{12.9}$$

Computing the Support Vector Classifier

Which we minimize w.r.t β, β_0 and ξ_i. Setting the respective derivatives to zero, we get

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{12.10}$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i \tag{12.11}$$

$$\alpha_i = C - \mu_i, \forall i \tag{12.12}$$

as well as the positivity constraints $\alpha_i, \mu_i, \xi_i \geq 0, \forall i$. By substituting (12.10)-(12.12) into (12.9), we obtain the Lagrangian (Wolfe) dual objective function

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} x_i^T x_{i'} \tag{12.13}$$

Computing the Support Vector Classifier

- We can get the dual problem of (12.8)

$$\max_{\alpha} L_D \quad \text{subject to} \quad \begin{cases} 0 \le \alpha_i \le C, \forall i \text{Why} \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases}$$

- Karush-Kuhn-Tucker conditions include the constraints

$$\alpha_i \left[y_i \left(x_i^T \beta + \beta_0 \right) - (1 - \xi_i) \right] = 0 \tag{12.14}$$

$$\mu_i \xi_i = 0 \tag{12.15}$$

$$y_i \left(x_i^T \beta + \beta_0 \right) - (1 - \xi_i) \ge 0 \tag{12.16}$$

- Together these equations (12.10)-(12.16) uniquely characterize the solution to the primal and dual problem.
- From (12.10) we see that the solution for β has the form

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i \tag{12.17}$$

Computing the Support Vector Classifier

- For $\hat{\alpha}_i = 0$, then $\hat{\xi}_i = 0$, points are outside the margin.
- For $\hat{\alpha}_i > 0$, which are called the support vectors.
- For $0 < \hat{\alpha}_i < C$, then $\hat{\xi}_i = 0$ why, means these support points lie on the edge of the margin;
- For $\hat{\alpha}_i = C$ and $0 < \hat{\xi}_i < 1$
- For $\hat{\alpha}_i = C$ and $\hat{\xi}_i = 1$
- For $\hat{\alpha}_i = C$ and $\hat{\xi}_i > 1$
- From (12.14) we can see that any of these margin points $(0 < \hat{\alpha}_i, \hat{\xi}_i = 0)$ can de used to solve for β_0, and we typically use an average of all the solutions for numerical stability.
- Given the solution $\hat{\beta}_0$ and $\hat{\beta}$, the decision function is

$$\hat{G}(x) = \text{sign}[\hat{f}(x)] = \text{sign}\left[x^T\hat{\beta} + \hat{\beta}_0\right] \tag{12.18}$$

The SVM as a Penalization Method

- The standard SVM:

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

$$\text{subject to} \xi_i \geq 0, y_i\left(\beta_0 + \boldsymbol{x}_i^\top\boldsymbol{\beta}\right) \geq 1-\xi_i, i = 1, 2, \cdots, n \qquad (1)$$

  where $\boldsymbol{\xi} = (\xi_1, \xi_2, \cdots, \xi_n)$ are the slack variables and $C > 0$ is a constant.

- The constraint of (1) can be written as

$$\xi_i \geq \left[1 - y_i\left(\beta_0 + \boldsymbol{x}_i^\top\boldsymbol{\beta}\right)\right]_+, \quad i = 1, 2, \cdots, n$$

- Then Objective function:

$$\frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\xi_i \geq \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\left[1 - y_i\left(\beta_0 + \boldsymbol{x}_i^\top\beta\right)\right]_+$$

  If and only if $\xi_i = \left[1 - y_i\left(\beta_0 + \boldsymbol{x}_i^\top\boldsymbol{\beta}\right)\right]_+$, the above inequality takes "=".

The SVM as a Penalization Method

- Then the solution of β_0, β in (1) is the same as the solution of the following optimization problem:

$$\min_{\beta_0,\beta} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n}\left[1 - y_i\left(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}\right)\right]_+$$

- Let $\lambda = 1/C$, The above optimization problem is equivalent to

$$\min_{\beta_0,\boldsymbol{\beta}} \sum_{i=1}^{n}\left[1 - y_i\left(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}\right)\right]_+ + \frac{\lambda}{2}\|\boldsymbol{\beta}\|^2 \qquad (2)$$

- This has the form "loss + penalty ", which is a familiar paradigm in function estimation.

Exercise:Prove It

Implication of the Hinge Loss

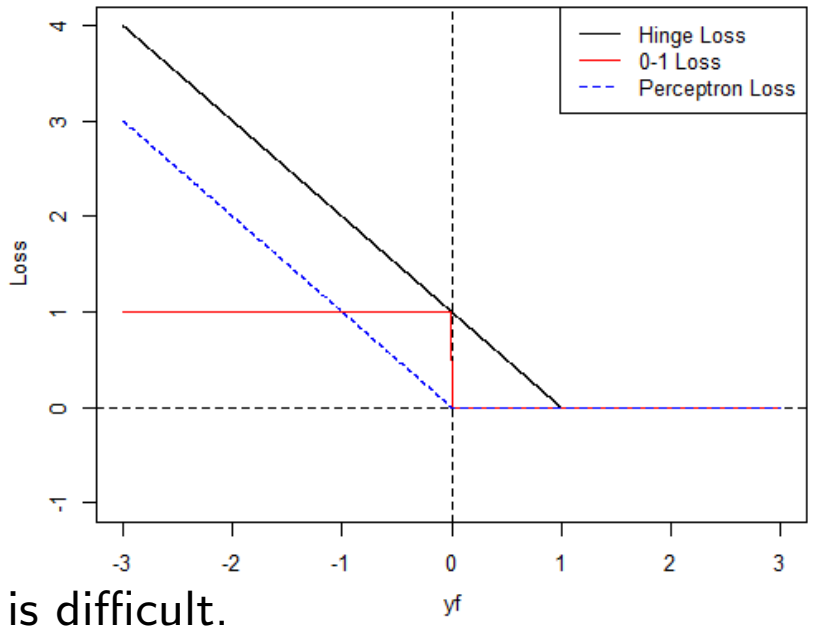
- Consider the expectation of the loss function:

$$E(\ell(yf)|x) = \ell(f)\Pr(y=1|x) + \ell(-f)\Pr(y=-1|x)$$

- Minimize expected loss when $\ell(.)$ is hinge loss or 0-1 loss, obtain:

$$f(x) = \text{sign}\left(\Pr(y=1|x) - \frac{1}{2}\right)$$

- 0-1 loss is the true loss function in binary classification.



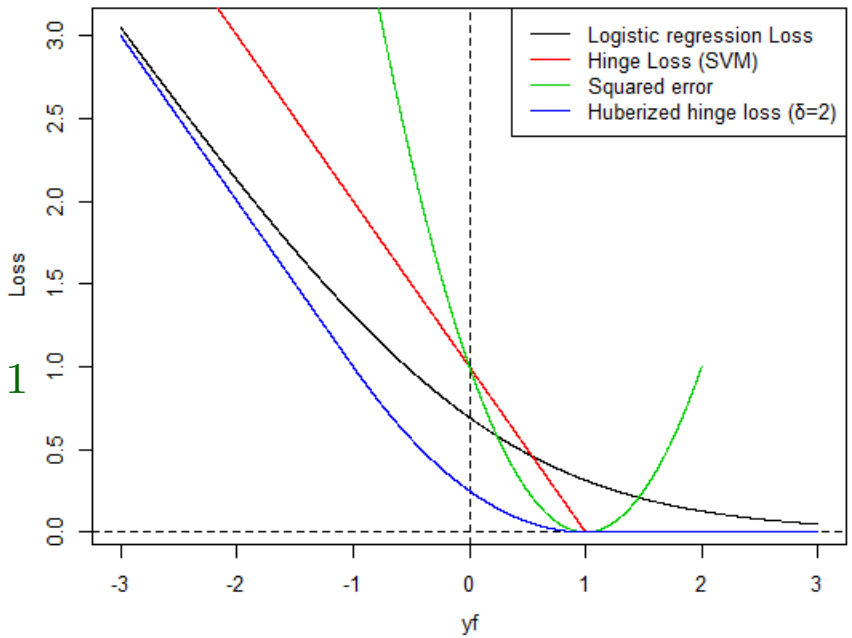
- But the following optimization problem is difficult.

$$\min_f \frac{1}{n}\sum_{i=1}^{n} I_{[y_i f(x_i)<0]}$$

- Since the solution to minimize the expected loss is the same, the hinge loss is also called the surrogate loss function.
- Compared with the perceptron loss, the hinge loss is not only classified correctly, but also the loss is only 0 when the confidence is high enough, that is, the hinge loss has higher requirements for learning.

Compare with other commonly used loss functions

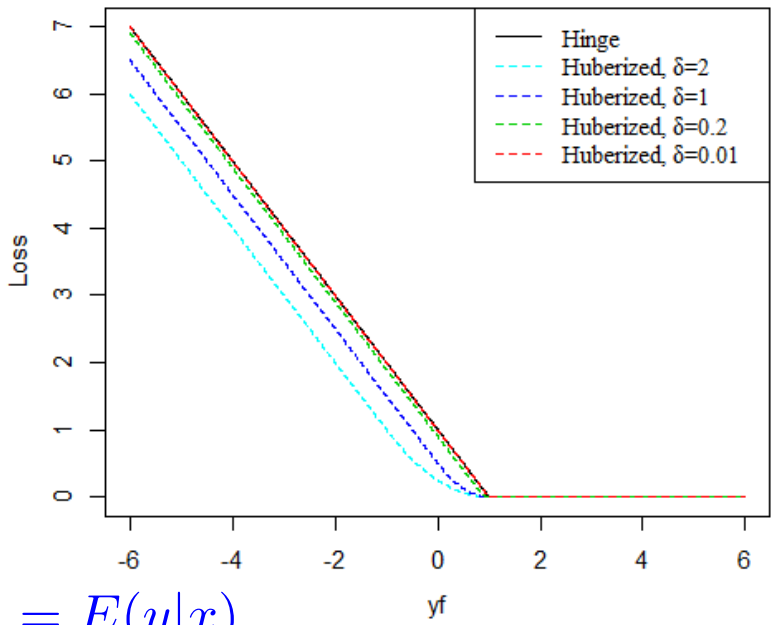
- SVM Hinge Loss: $[1 - yf(x)]_+$
- Logistic Regression Loss:
$$\log[1 + e^{-yf(x)}]$$
- Squared Error:
$$[y - f(x)]^2 = [1 - yf(x)]^2$$
- Huberized Hinge Loss:
$$\ell(yf) = \begin{cases} 0, & yf > 1 \\ (1 - yf)^2/(2\delta), & 1 - \delta < yf \leq 1 \\ 1 - t - \delta/2, & yf \leq 1 - \delta. \end{cases}$$
where $\delta > 0$ is a pre-specfied constant.



- Examination of the hinge loss function shows that it is reasonable for two-class classification, when compared to Logistic loss and Squared error.
  - Logistic loss has similar tails as the SVM loss, but there is never zero penalty for points well inside their margin.
  - Squared error gives a quadratic penalty, and points well inside their own margin have a strong influence on the model as well.
- Rosset and Zhu (2007) proposed a Huberized hinge loss, which is very similar to hinge loss in shape.

Hinge Loss and Huberized Loss

- As the δ decreases, the hubrized loss gets closer to the hinge loss. In fact, when $\delta \to 0$, the limit of hubrized loss is the hinge loss.
- Unlike the hinge loss, the huberized loss function is differentiable everywhere.
- It is easy to prove that the huberized loss is satisfied



$$\arg\min_f E[\ell(yf)|x] = 2\Pr(y=1|x) - 1 = E(y|x)$$

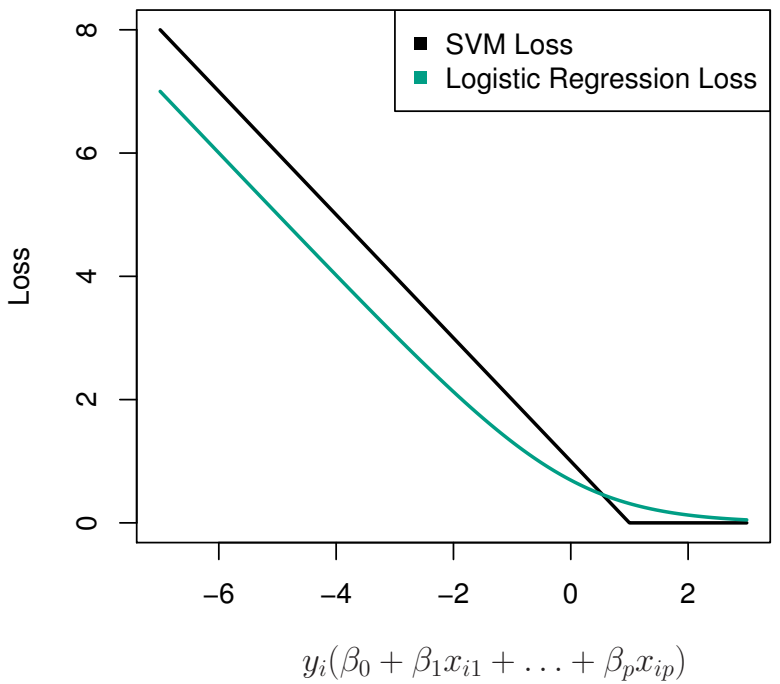
$$E(y|x) > 0 \Leftrightarrow \text{sign}[\Pr(y=1|x) - \frac{1}{2}] > 0.$$

- Rosset and Ji Zhu(2007): In the model with logistic loss, huberized loss, and square hinge loss with ℓ_1-penalty, the impact of outliers on the received huberized loss model is minimal.

Support Vector versus Logistic Regression?

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

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



This has the form *loss plus penalty*.

The loss is known as the *hinge loss*.

Very similar to "loss" in logistic regression (negative log-likelihood).

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.

l_1-SVM

- The penalty of SVM is called the ridge penalty. It is well known that this shrinkage has the effect of controlling the variances of $\hat{\beta}$;
- Hence possibly improves the fitted models prediction accuracy, especially when there are many highly correlated features.
- The SVM uses all variables in classification, which could be a great disadvantage in high-dimensional classification.
- Apply lasso penalty (l_1-norm) to the SVM, Consider the optimization problem(l_1-SVM)

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{n}\sum_{i=1}^{n}\left[1-y_i\left(\beta_0+\boldsymbol{x}_i^{\top}\boldsymbol{\beta}\right)\right]_+ + \lambda\|\boldsymbol{\beta}\|_1$$

  - the l_1-SVM is able to automatically discard irrelevant features.
  - In the presence of many noise variables, the l_1-SVM has signifi- cant advantages over the standard SVM.

l_1-SVM

- The optimization problem of l_1-SVM is a linear program with many constraints, and ecient algorithms can be complex.
- The solution paths can have many jumps, and show many discontinuities. For this reason, some authors prefer to replace the usual hinge loss with other smooth loss function, such as square hinge loss, huberized hinge loss, etc. (Zhu, Rosset, Hastie, 2004).
- The number of variables selected by the l_1-norm penalty is upper bounded by the sample size n.
- For highly correlated and relevant variables, the l_1-norm penalty tends to select only one or a few of them.

The hybrid huberized SVM(HHSVM)

- Li Wang, Ji Zhu and Hui Zou(2007) apply the elasticnet penalty to the SVM and propose the HHSVM:

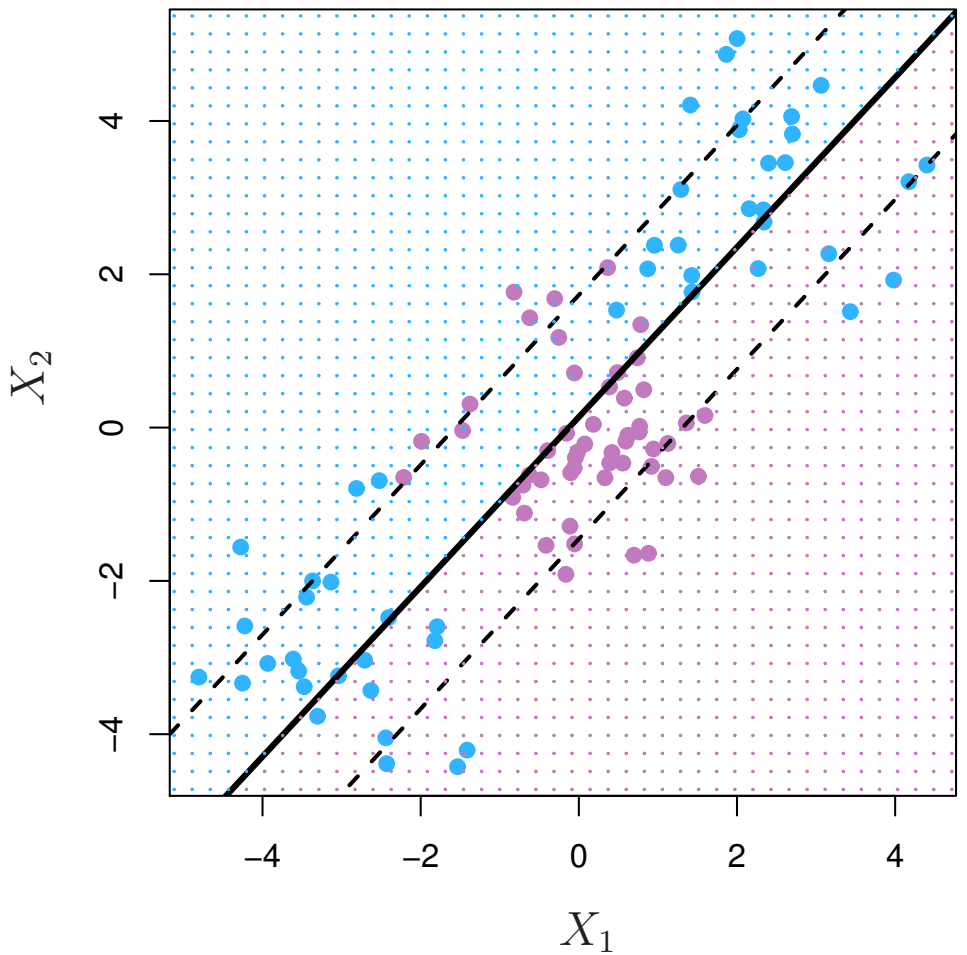
$$\min_{(\beta_0,\boldsymbol{\beta})} \sum_{i=1}^{n} \ell\left(y_i\left(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}\right)\right) + \lambda_1 \|\boldsymbol{\beta}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\beta}\|_2^2 \qquad (3)$$

Where the loss function is huberized hinge loss:

$$\ell(yf) = \begin{cases} 0, & yf > 1 \\ (1-yf)^2/(2\delta), & 1-\delta < yf \le 1 \\ 1 - yf - \delta/2, & yf \le 1-\delta \end{cases}$$

and $\delta > 0$ is a pre-specified constant. The delfault choice for δ is 2 unless specified otherwise.

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?

前面刻画非线性，引入2次项，3次项，交叉项等

Feature Expansion

- Enlarge the space of features by including transformations; e.g. X_1^2, X_1^3, X_1X_2, $X_1X_2^2$,.... 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 non-linear decision boundaries in the original space.

Feature Expansion

- Enlarge the space of features by including transformations; e.g. X_1^2, X_1^3, X_1X_2, $X_1X_2^2$,.... 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 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

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 = 0$$

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

hyperplane divid p-dimensional space into two halves

we could instead fit a support vector classifier using $2p$ features

$$X_1, X_1^2, X_2, X_2^2, \cdots, X_p, X_p^2$$

Then (9.12)-(9.15) would become

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

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

$$\sum_{i=1}^{n} \epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1 \tag{9.16}$$

Example:

spam {ElemStatLearn}

Spam Detection

- data from 4601 emails sent to an individual (named George, at HP labs, before 2000). Each is labeled as *spam* or *email*.
- goal: build a customized spam filter.
- input features: relative frequencies of 57 of the most commonly occurring words and punctuation marks in these email messages.

	george	you	hp	free	!	edu	remove
spam	0.00	2.26	0.02	0.52	0.51	0.01	0.28
email	1.27	1.27	0.90	0.07	0.11	0.29	0.01

Average percentage of words or characters in an email message equal to the indicated word or character. We have chosen the words and characters showing the largest difference between spam *and* email.

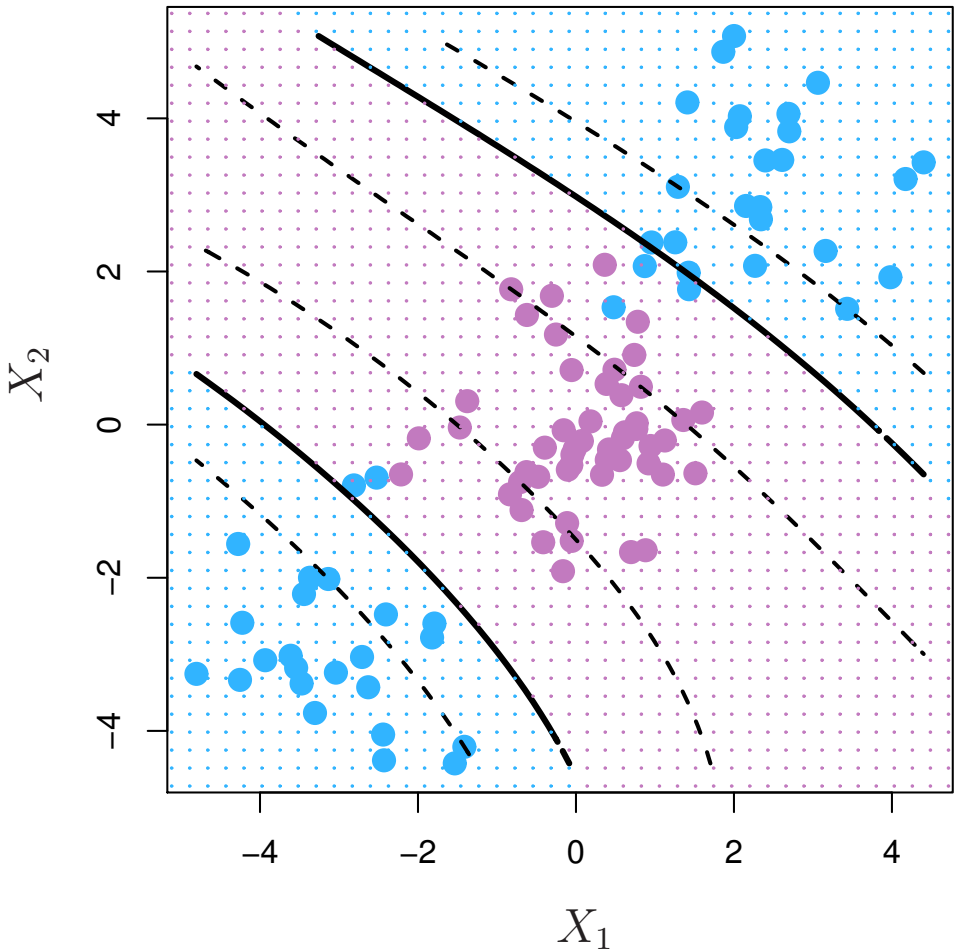
Decision tree; randomforest; boosting; linear SVM
Using measurement: sensitivity specificity ROC AUC

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

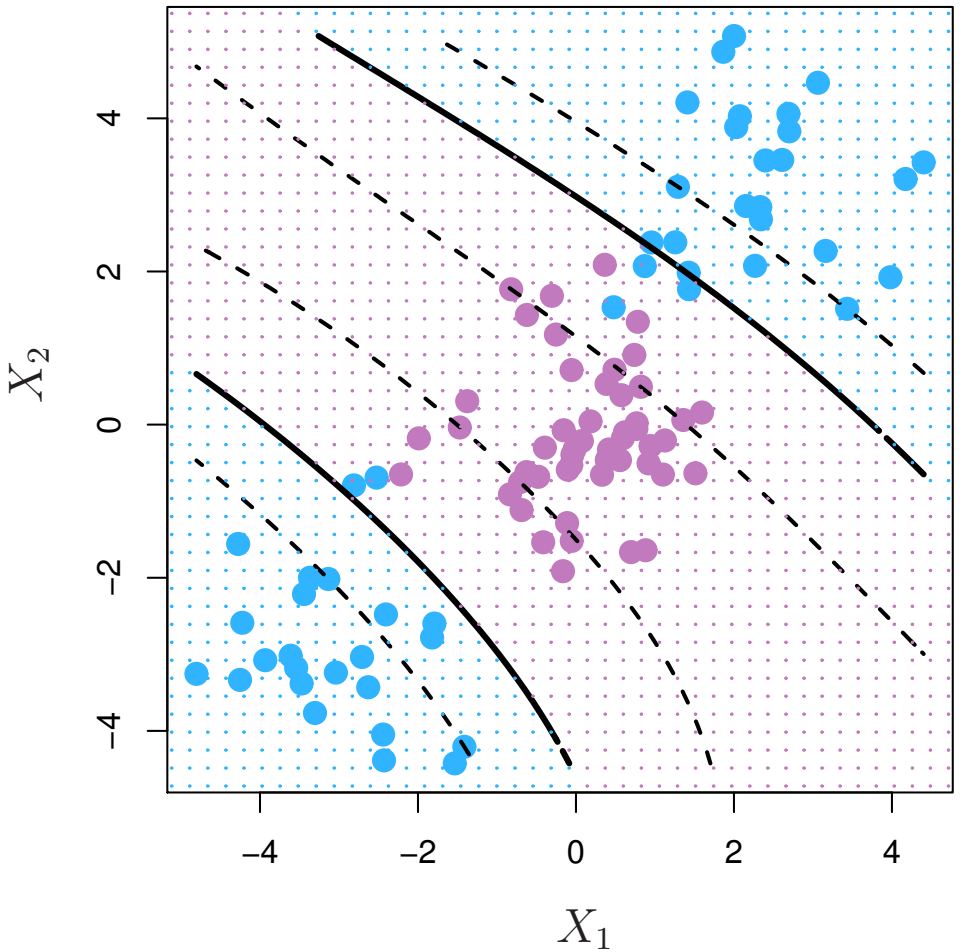


Cubic Polynomials

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



$$\beta_0+\beta_1X_1+\beta_2X_2+\beta_3X_1^2+\beta_4X_2^2+\beta_5X_1X_2+\beta_6X_1^3+\beta_7X_2^3+\beta_8X_1X_2^2+\beta_9X_1^2X_2=0$$

Nonlinearities and Kernels

- Polynomials (especially high-dimensional ones) get wild rather fast.
- There is a more elegant and controlled way to introduce nonlinearities in support-vector classifiers — through the use of *kernels*.
- Before we discuss these, we must understand the role of *inner products* in support-vector classifiers.

Inner products and support vectors

xi是一个p维向量

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j} \quad \text{— } \textit{inner product between vectors}$$

Inner products and support vectors

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j} \quad \text{— } \textit{inner product between vectors}$$

- The linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle \quad \text{— } \textit{n parameters}$$

Inner products and support vectors

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j} \quad \text{— } \textit{inner product between vectors}$$

- The linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle \quad \text{— } \textit{n parameters}$$

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

Inner products and support vectors

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j} \quad \text{— } \textit{inner product between vectors}$$

- The linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle \quad \text{— } n \textit{ parameters}$$

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

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

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

$\mathcal{S}$ is the *support set* of indices i such that $\hat{\alpha}_i > 0$. [see slide 8]

Kernels and Support Vector Machines

- If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!

Kernels 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!

Kernels 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 $d = 2$.

Kernels 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!

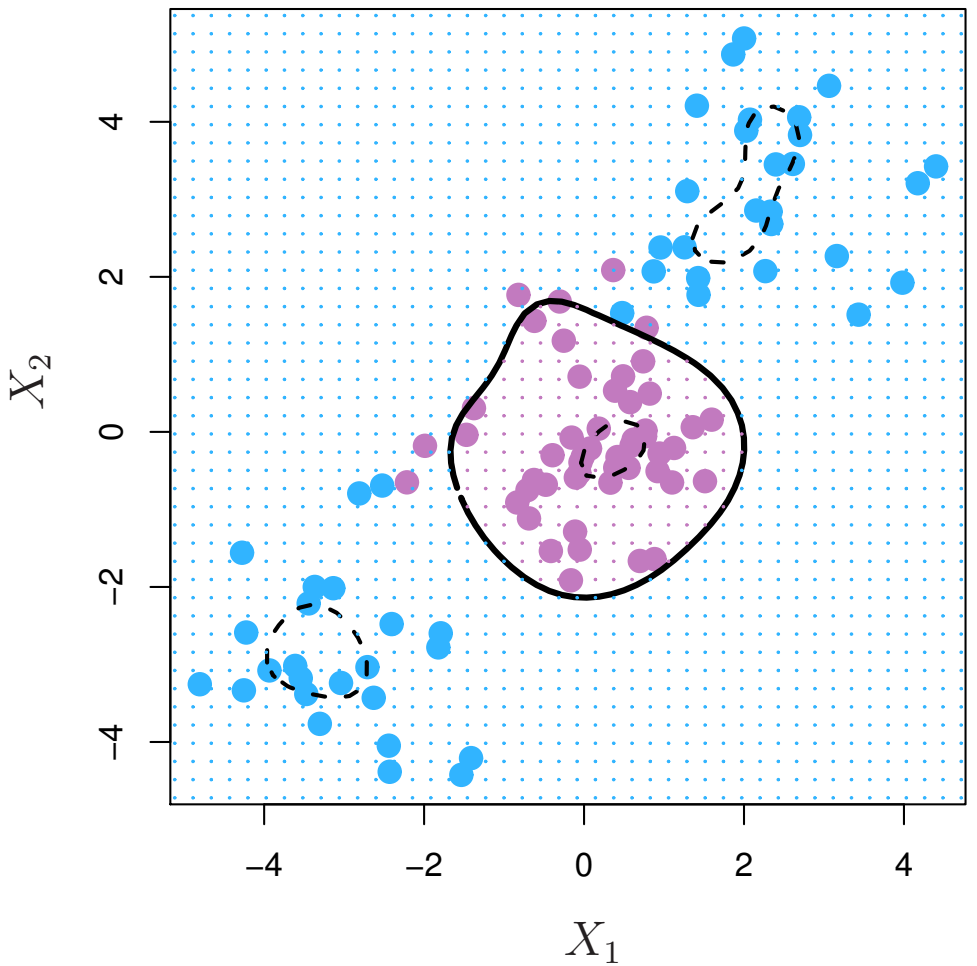
Try it for $p = 2$ and $d = 2$.

- The solution has the form

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

Radial Kernel

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

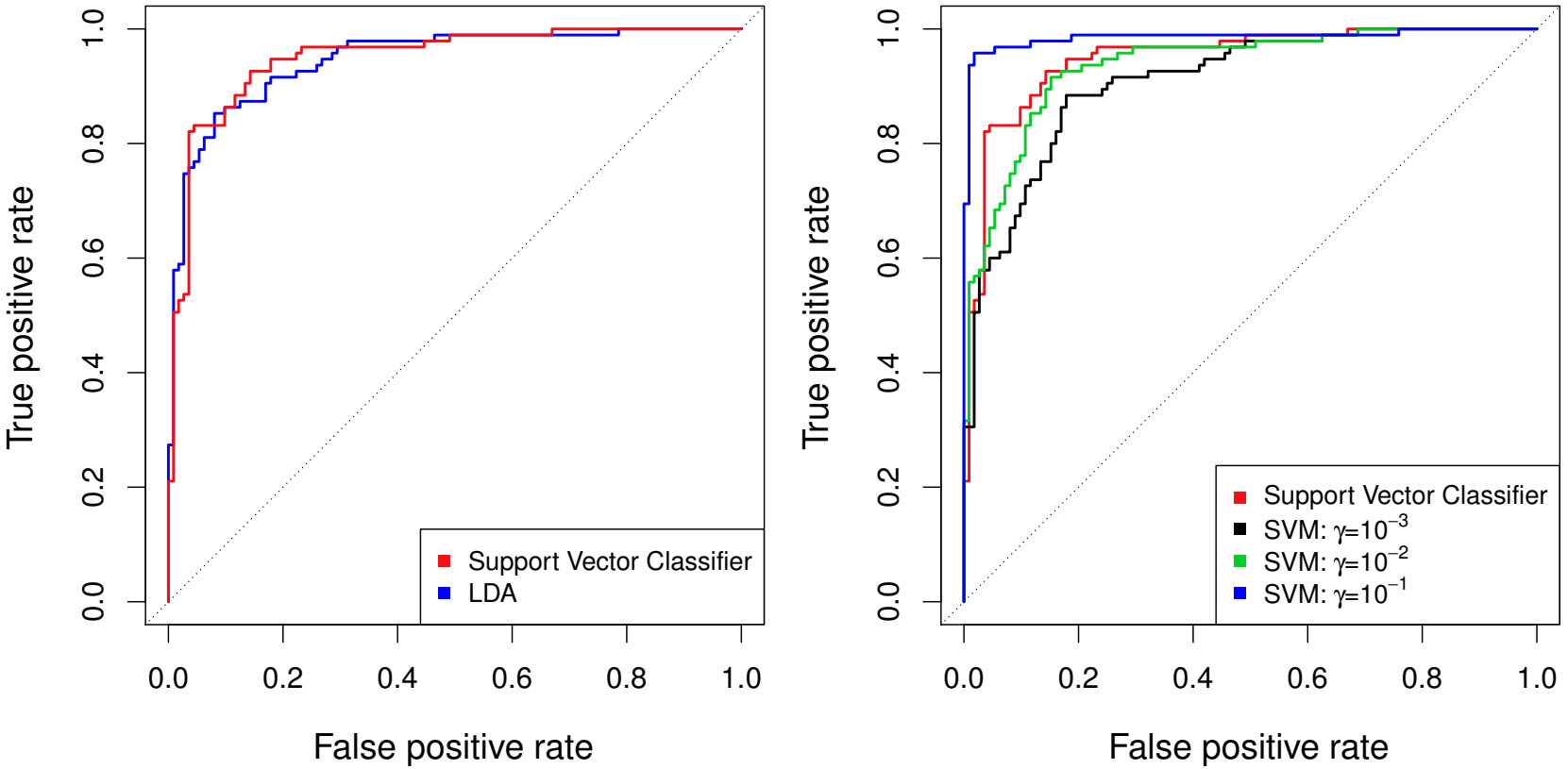


$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i)$$

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.

SVMs: more than 2 classes?

The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?

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.

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.

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.

Support vector regression

In this section we show how SVMs can be adapted for regression with a quantitative response, in ways that inherit some of the properties of the SVM classifier. We first discuss the linear regression model

$$f(x) = x^T\beta + \beta_0 \tag{12.35}$$

And then handle nonlinear generalizations. To estimate β, we consider minimization of

$$H\left(\beta, \beta_0\right) = \sum_{i=1}^{N} V\left(y_i - f\left(x_i\right)\right) + \frac{\lambda}{2}\|\beta\|^2 \tag{12.36}$$

Support vector regression

$$V_\epsilon(r) = \begin{cases} 0 & \text{if } |r| < \epsilon \\ |r| - \epsilon, & \text{otherwise} \end{cases} \tag{12.37}$$

This is an "ϵ-insensitive" error measure, ignoring errors of size less than ϵ (left panel of Figure 12.8) There is a rough analogy with the support vector classification setup, where points on the correct side of the decision boundary and far away from it, are ignored in the optimization. In regression, these "low error" points are the ones with small residuals. It is interesting to contrast this with error measures used in robust regression in statistics. The most popular, due to Huber (1964), has the form

$$V_H(r) = \begin{cases} r^2/2 & \text{if} |r| < c \\ c|r| - c^2/2, & \text{if} |r| > c \end{cases} \tag{12.38}$$

shown in the right panel of Figure 12.8. This function reduces from quadratic to linear the contributions of observations with absolute residual greater than a prechosen constant c. This makes the fitting less sensitive to outliers. The support vector error measure (12.37) also has linear tails (beyond ϵ), but in addition it flattens the contributions of those cases with small residuals.

Support vector regression

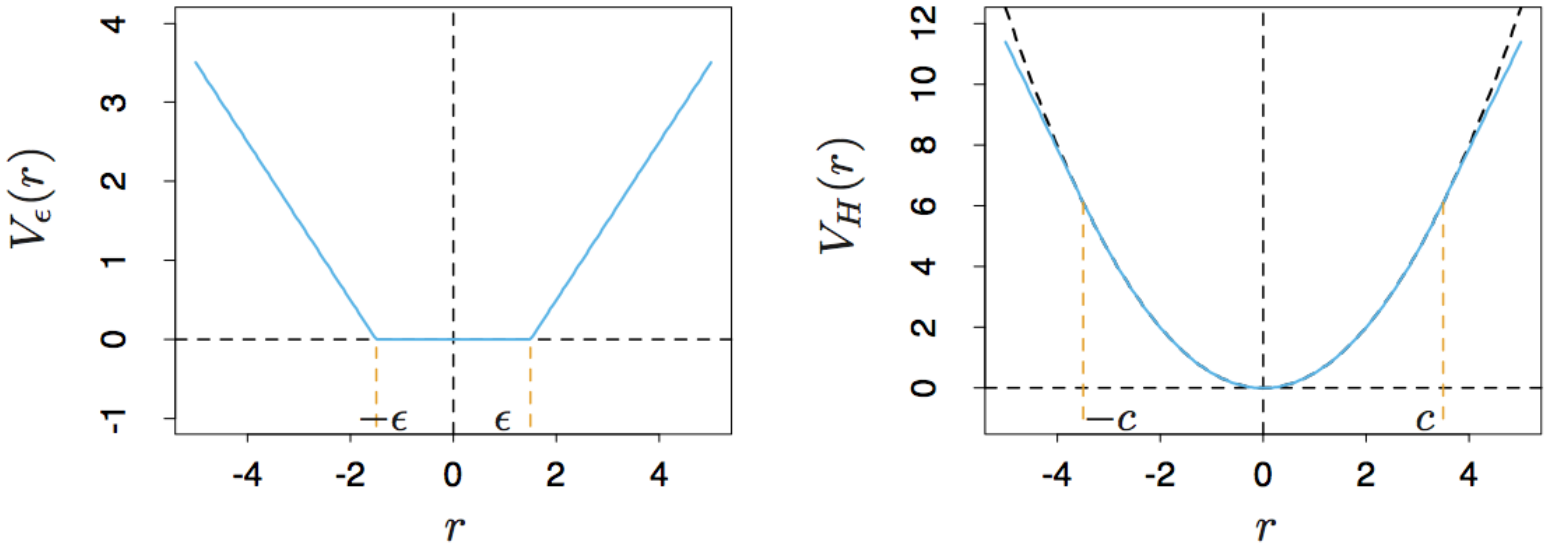


FIGURE 12.8. *The left panel shows the ϵ-insensitive error function used by the support vector regression machine. The right panel shows the error function used in Huber's robust regression (blue curve). Beyond $|c|$, the function changes from quadratic to linear.*

Support vector regression

As discussed in Section 12.3.3, this kernel property is not unique to support vector machines. Suppose we consider approximation of the regression function in terms of a set of basis functions $\{h_m(x)\}$, $m = 1, 2, \cdots, M$:

$$f(x) = \sum_{m=1}^{M} \beta_m h_m(x) + \beta_0 \tag{12.42}$$

To estimate β and β_0 we minimize

$$H(\beta, \beta_0) = \sum_{i=1}^{N} V(y_i - f(x_i)) + \frac{\lambda}{2} \sum \beta_m^2 \tag{12.43}$$

for some general error measure $V(r)$. For any choice of $V(r)$, the solution $\hat{f}(x) = \sum \hat{\beta}_m h_m(x) + \hat{\beta}_0$ has the form

$$\hat{f}(x) = \sum_{i=1}^{N} \hat{\alpha}_i K(x, x_i) \tag{12.44}$$

Support vector regression

with $K(x,y) = \sum_{m=1}^{M} h_m(x)h_m(y)$. Notice that this has the same form as both the radial basis function expansion and a regularization estimate, discussed in Chapters 5 and 6.

For concreteness, let's work out the case $V(r) = r^2$. Let $\mathbf{H}$ be the $N \times M$ basis matrix with imth element $h_m(x_i)$, and suppose that $M > N$ is large. For simplicity we assume that $\beta_0 = 0$, or that the constant is absorbed in h; see Exercise 12.3 for an alternative.

We estimate β by minimizing the penalized least squares criterion

$$H(\beta) = (\mathbf{y} - \mathbf{H}\beta)^T (\mathbf{y} - \mathbf{H}\beta) + \lambda \|\beta\|^2 \tag{12.45}$$

The solution is

$$\hat{\mathbf{y}} = \mathbf{H}\hat{\beta} \tag{12.46}$$

with $\hat{\beta}$ determined by

$$-\mathbf{H}^T \left(\mathbf{y} - \mathbf{H}\hat{\beta}\right) + \lambda\hat{\beta} = 0 \tag{12.47}$$

Support vector regression

From this it appears that we need to evaluate the $M \times M$ matrix of inner products in the transformed space. However, we can premultiply by $\mathbf{H}$ to give

$$\boldsymbol{H}\hat{\beta} = \left(\boldsymbol{H}\boldsymbol{H}^T + \lambda \boldsymbol{I}\right)^{-1} \mathbf{H}\mathbf{H}^T\mathbf{y} \tag{12.48}$$

The $N \times N$matrix $\mathbf{H}\mathbf{H}^T$ consists of inner products between pairs of observations i, i'; that is, the evaluation of an inner product kernel $\{\mathbf{H}\mathbf{H}^T\}_{i,i'} = K(x_i, x_{i'})$. It is easy to show (12.44) directly in this case, that the predicted values at an arbitrary x satisfy

$$\begin{aligned} \hat{f}(x) &= h(x)^T\hat{\beta} \\ &= \sum_{i=1}^{N} \hat{\alpha}_i K(x, x_i) \end{aligned} \tag{12.49}$$

where $\hat{\alpha} = (\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I})^{-1}\mathbf{y}$. As in the support vector machine, we need not specify or evaluate the large set of functions $h_1(x), h_2(x), \cdots, h_m(x)$. Only the inner product kernel $K(x_i, x_{i'})$ need be evaluated, at the N training points for each i, i' and at points x for predictions there. Careful choice of h_m (such as the eigenfunctions of particular, easy-to-evaluate kernels K) means, for example, that $\mathbf{H}\mathbf{H}^T$ can be computed at a cost of $N^2/2$ evaluations of K, rather than the direct cost N^2M.