



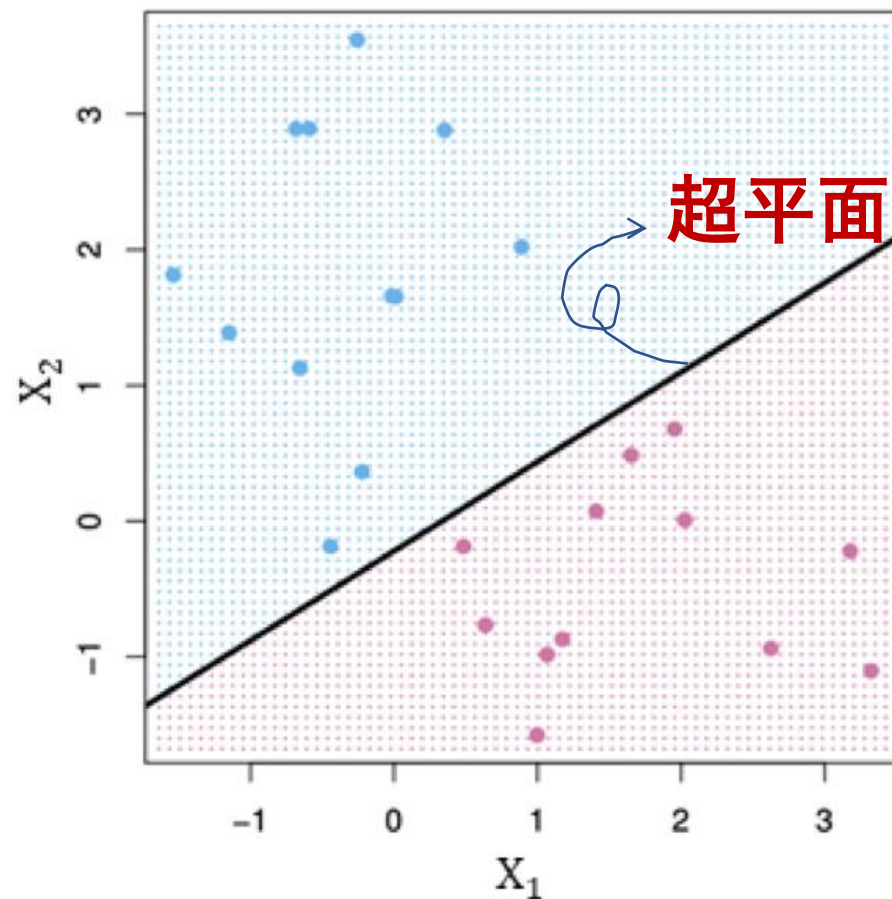
## 第九章 支持向量机

本章，我们先用一种直接的方式来处理二分类问题：

我们试图在特征空间中找到一个平面来分离类。例如：

如果做不到，我们就在两个方面发挥创意：

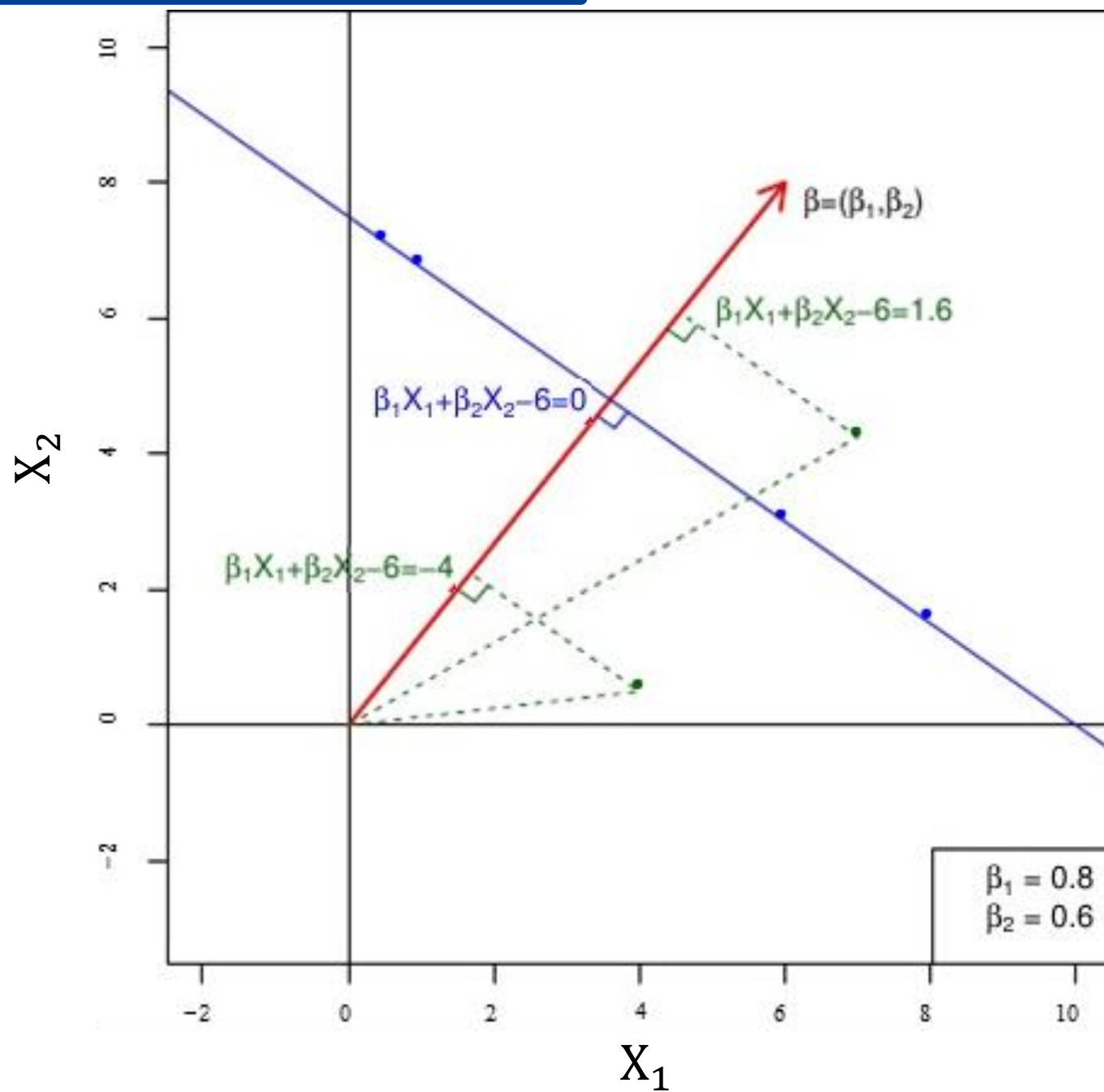
- 我们弱化了“分离”的含义；
- 我们丰富和扩大特征空间，以便分离成为可能。

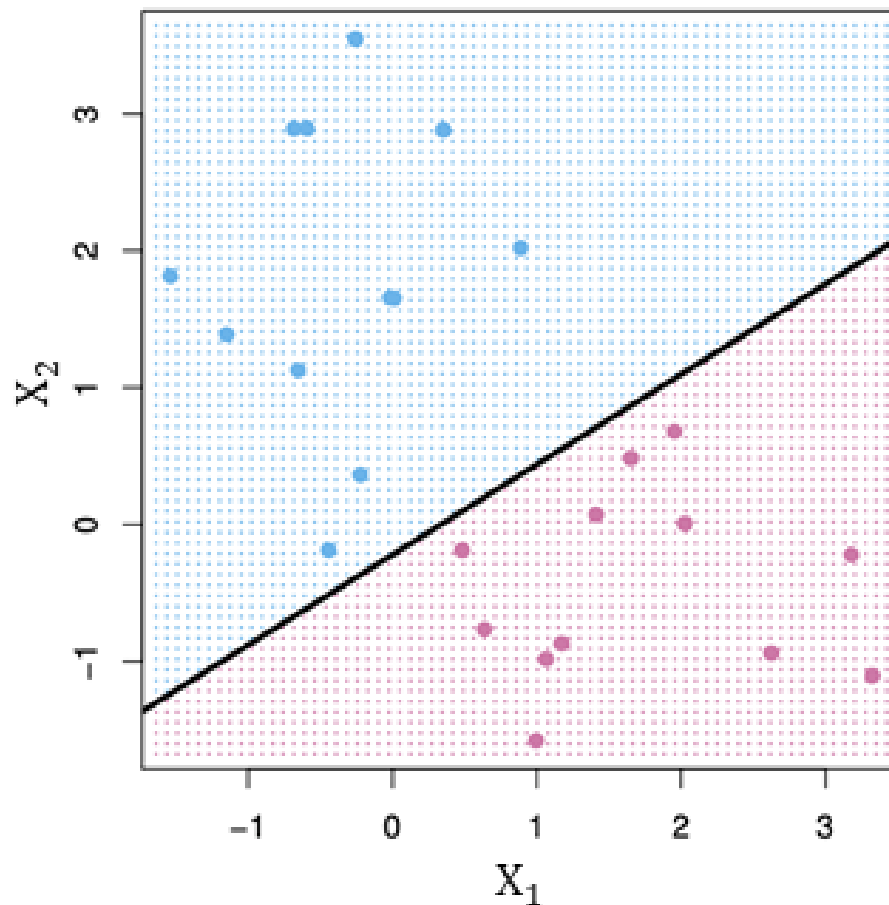
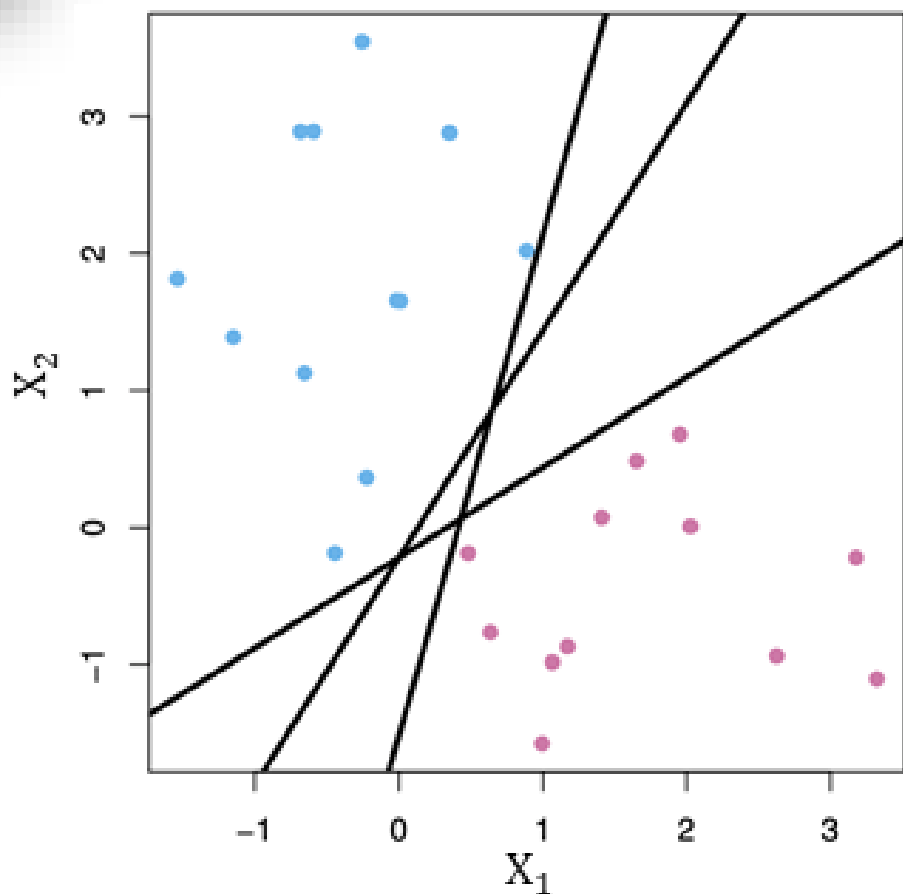


- $p$  维的超平面是  $p-1$  维的平面仿射子空间。
- 一般来说，超平面的方程有如下形式

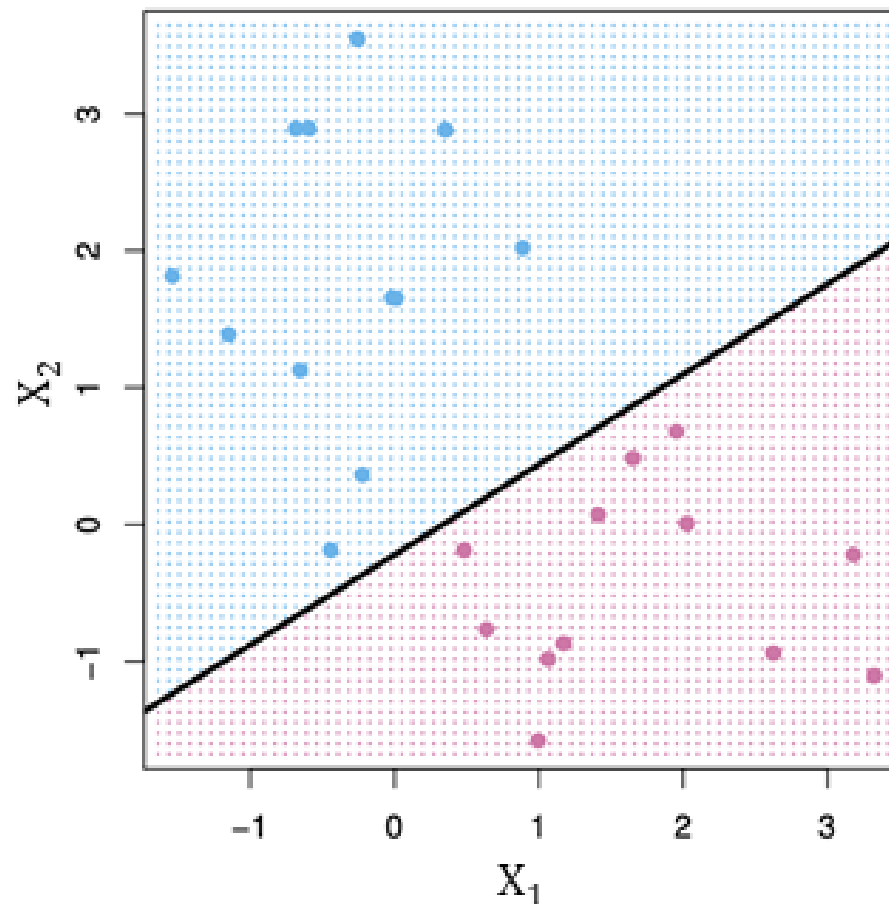
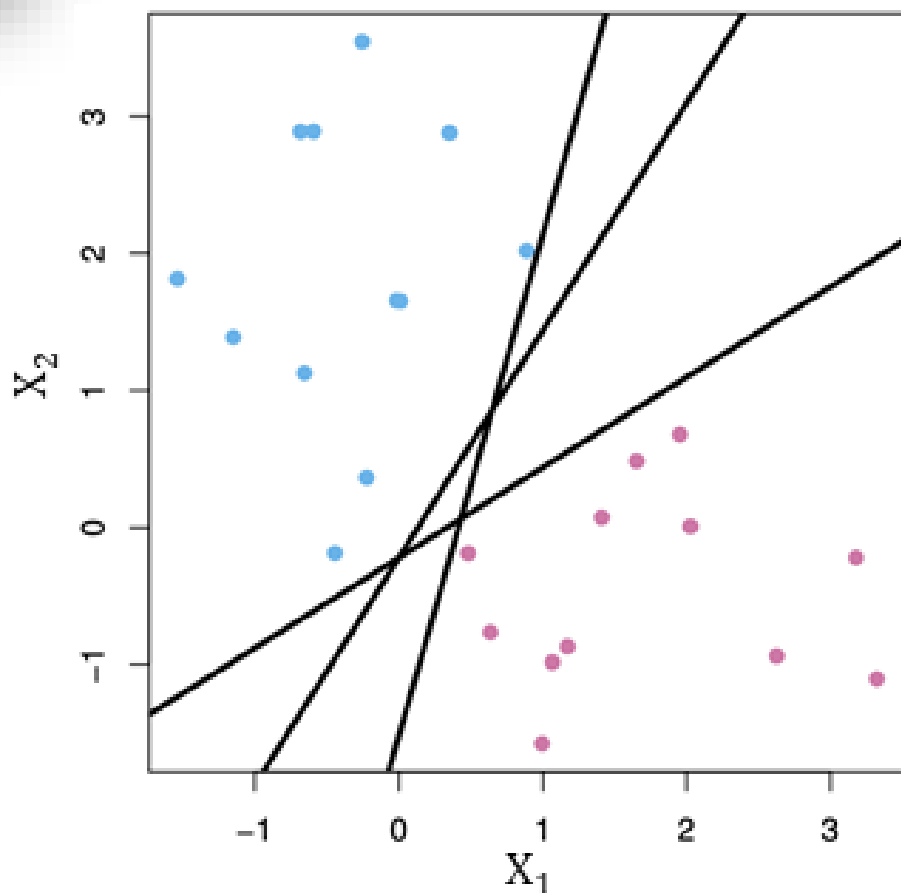
$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p = 0$$

- 在  $p = 2$  维中，超平面是一条线。
- 如果  $\beta_0 = 0$ ，超平面经过原点，否则不经过。
- 向量  $\beta = (\beta_1, \beta_2, \dots, \beta_p)$  称为法向量——它指向与超平面表面正交的方向。





- 如果  $f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$ ，则  $f(x) > 0$  表示超平面一侧的点， $f(x) < 0$  表示超平面另一侧的点。
- 如果我们将带颜色的点编码为  $y_i = +1$  表示蓝色，和  $y_i = -1$  代表淡紫色，那么如果  $y_i f(x_i) > 0$  对于所有  $i=1, \dots, n$ ， $f(x) = 0$  定义了一个**分割超平面**。

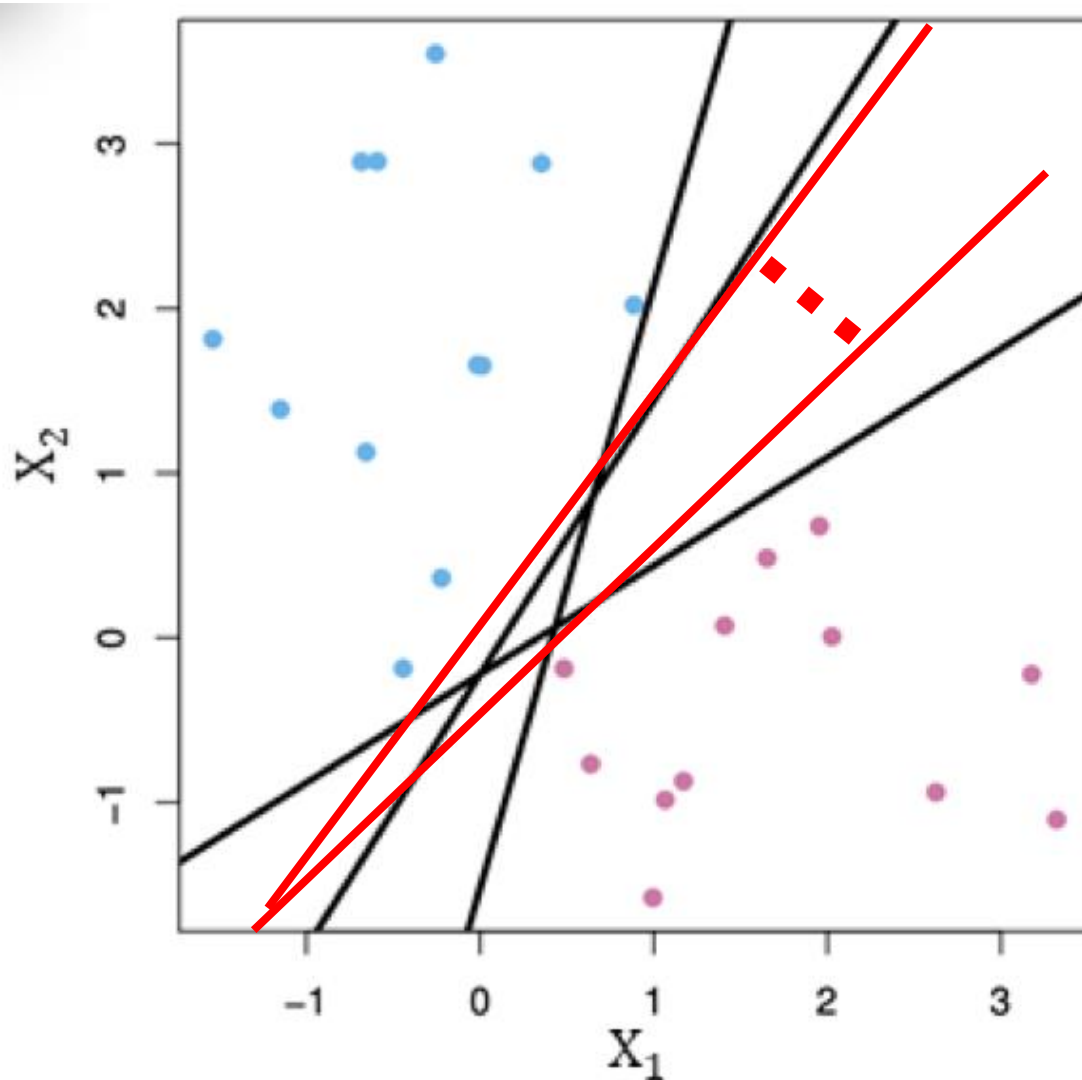


测试观测会被判定为哪个类完全取决于它落在分割超平面的哪一侧。即，根据 $f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \dots + \beta_p x_p^*$ 的符号分类。为正，分为1类，为负分为-1类。还可以根据 $f(x^*)$ 大小进行分类，若离0很远，即离分割超平面很远，能非常确信对 $x^*$ 的分类判断。



## (一) 最大间隔分类器

分割超平面若存在，则存在无穷多个。



用哪个好?



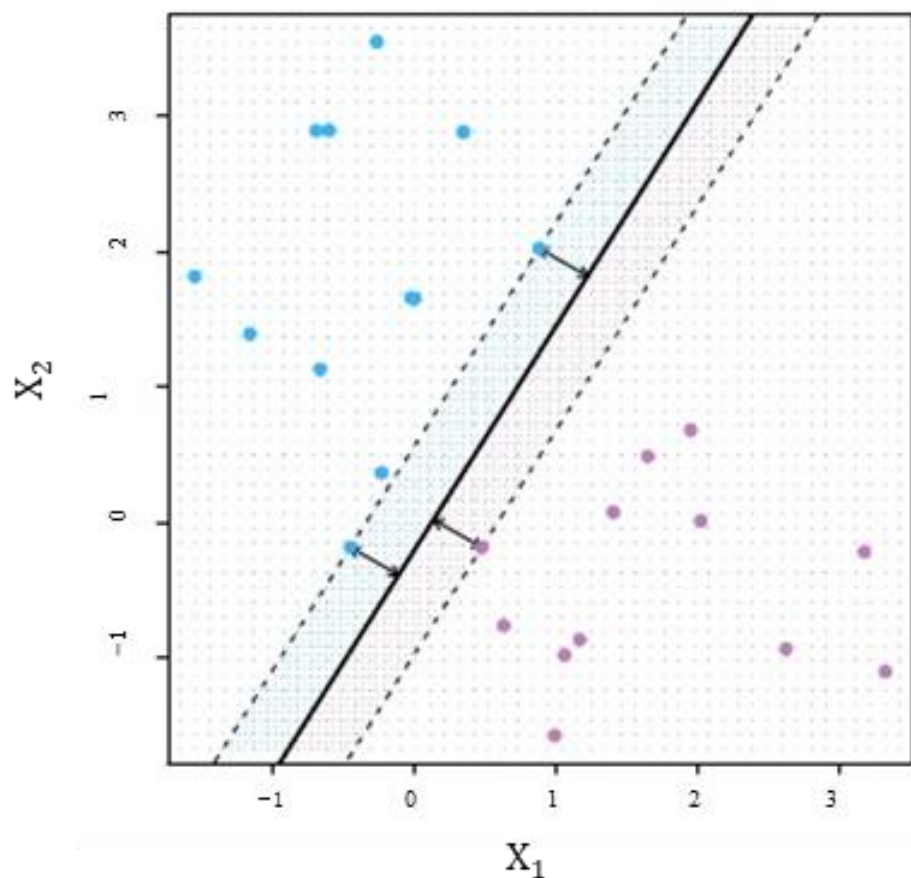
在所有分离超平面中，找出使两类之间的间隙或间隔最大的超平面。

首先，计算每个训练观测到一个分割超平面的距离，取最小值，称为**间隔**（margin）。这个间隔值最大的那个分割超平面即为最大间隔超平面。

然后，用此最大间隔超平面判断测试样本落在哪一侧，就可以分类。此为最大间隔分类器。

见下图。

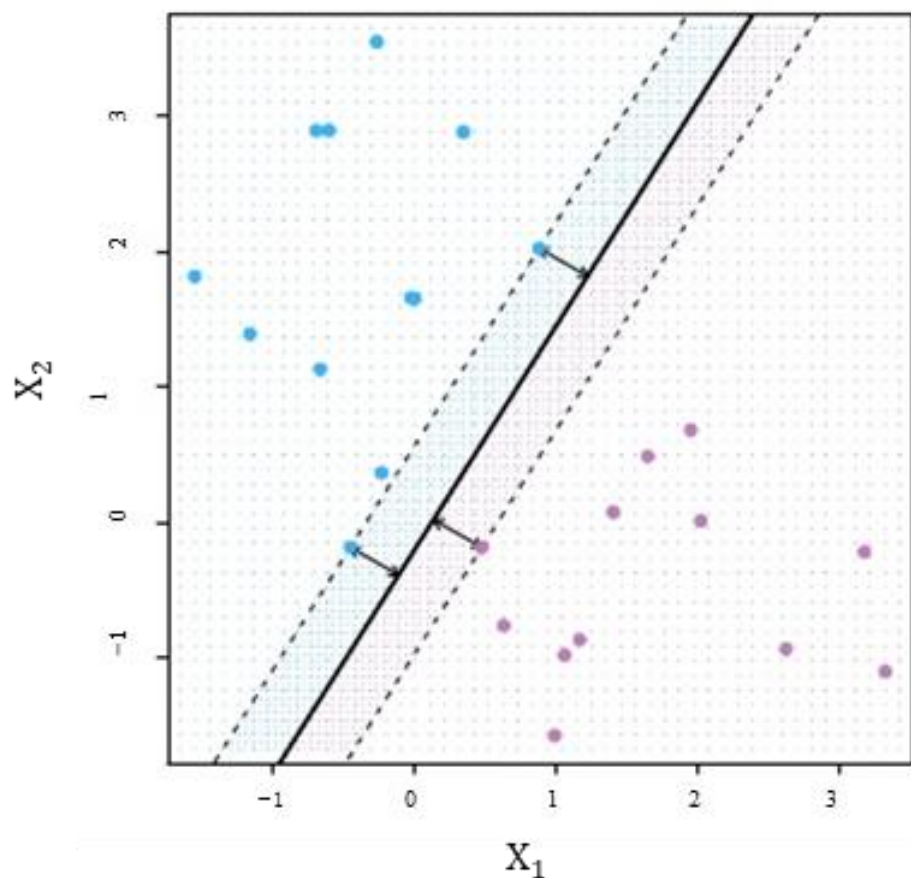
在所有分离超平面中，找出使两类之间的间隙或间隔最大的超平面。



此例中，有3个训练观测到最大间隔超平面的距离最小，它们被称为“**支持向量**”（Support Vector, SV）。

这3个点的改变会明显影响最大间隔超平面走向，其它点不会。这3个点“**支持**”着分类器的形成与分类。

在所有分离超平面中，找出使两类之间的间隙或间隔最大的超平面。如何构建？ -> **约束优化问题！**



$y_i$ 的可能取值为-1和1。

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

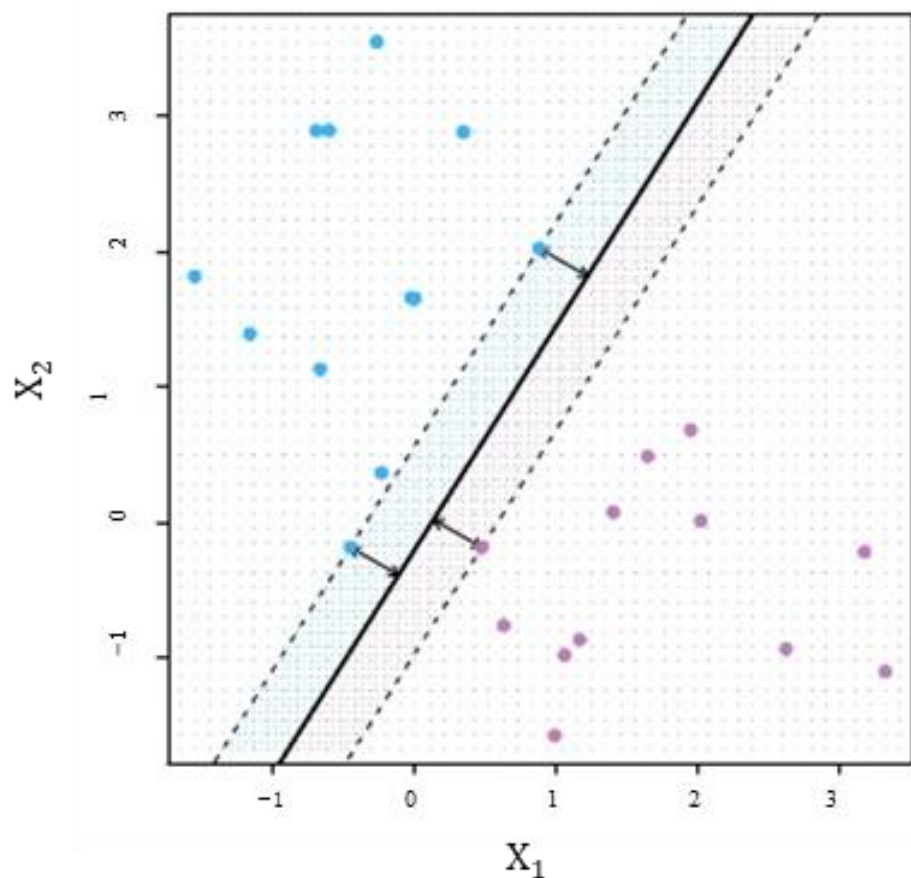
$$\text{subject to } \sum_{j=1}^p \beta_j^2 = 1,$$

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

for all  $i = 1, \dots, N$ .

约束条件保证了每个观测落在超平面正确的一侧！  
M就是间隔，找出最大化的M！

在所有分离超平面中，找出使两类之间的间隙或间隔最大的超平面。如何构建？ -> **约束优化问题！**



$y_i$ 的可能取值为-1和1。

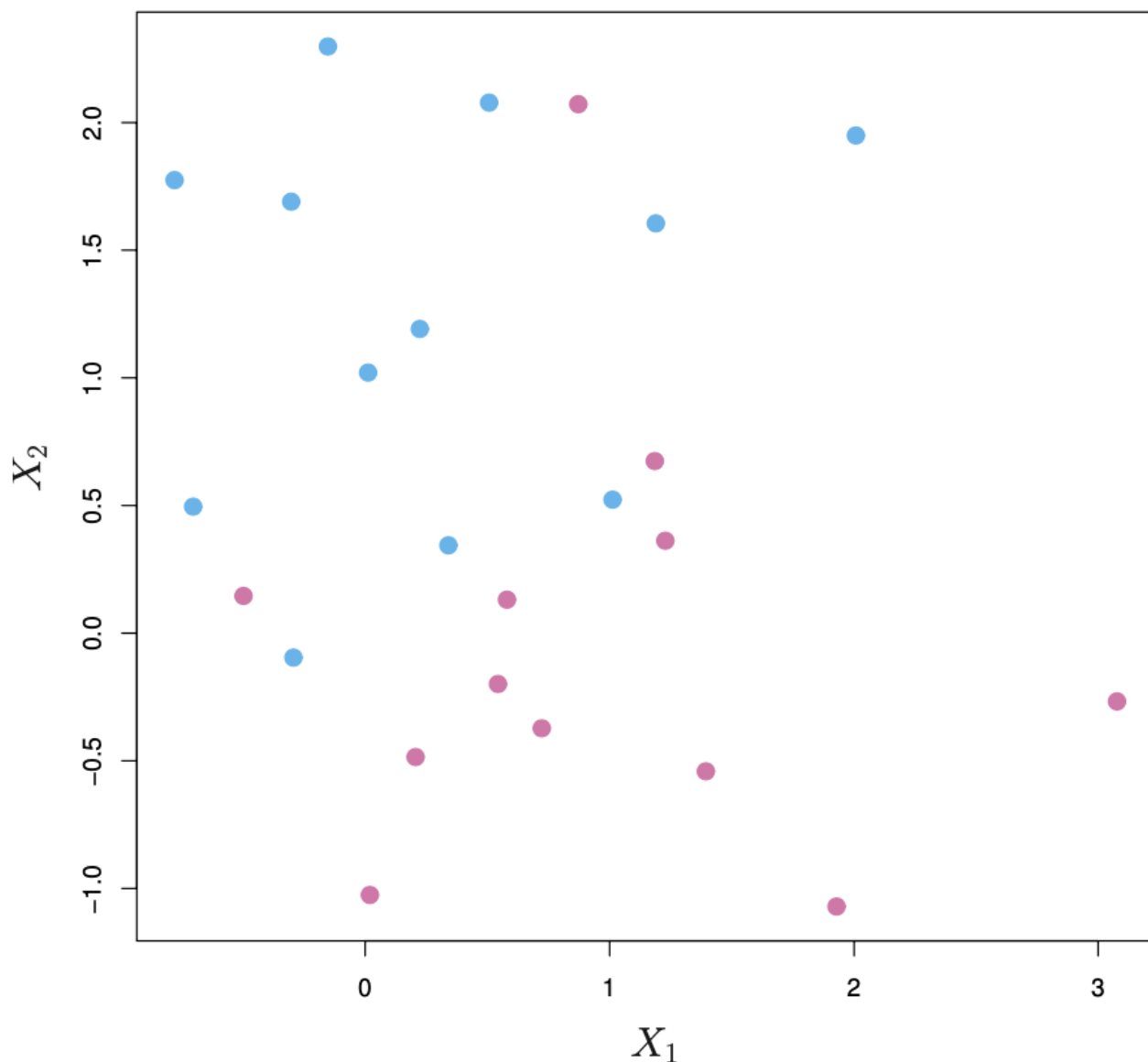
$$\text{maximize } M$$
$$\beta_0, \beta_1, \dots, \beta_p$$

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

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

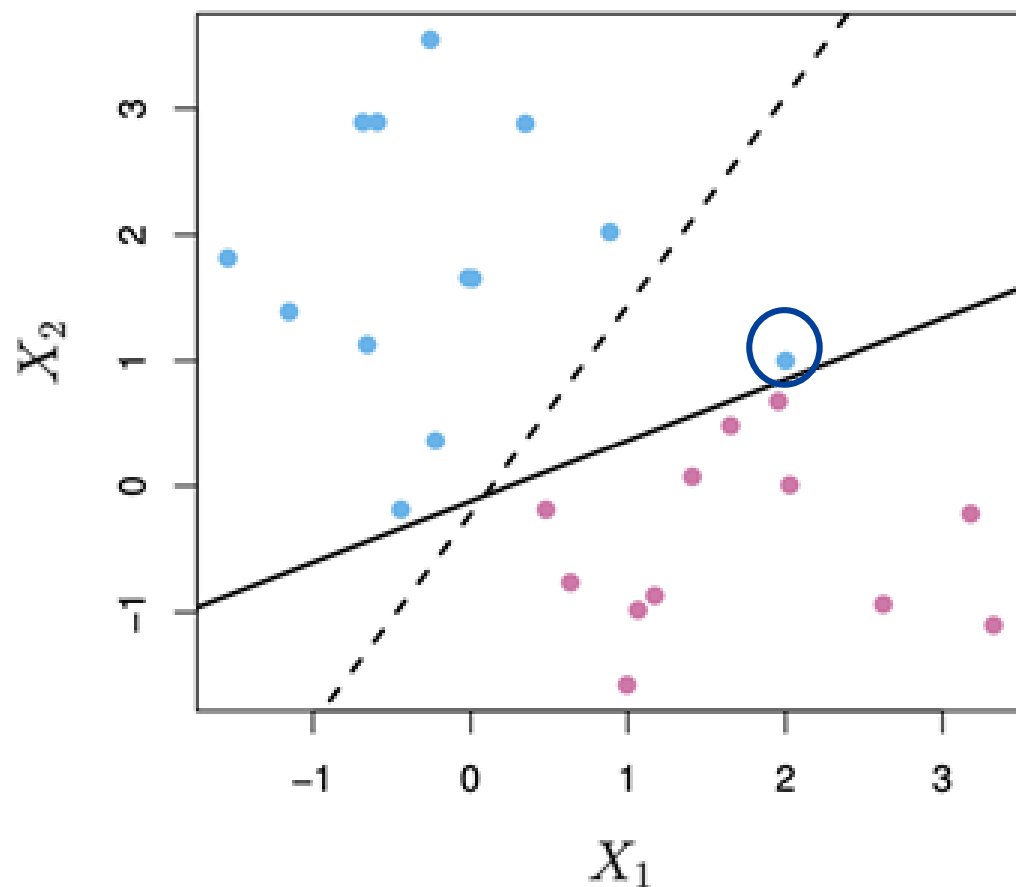
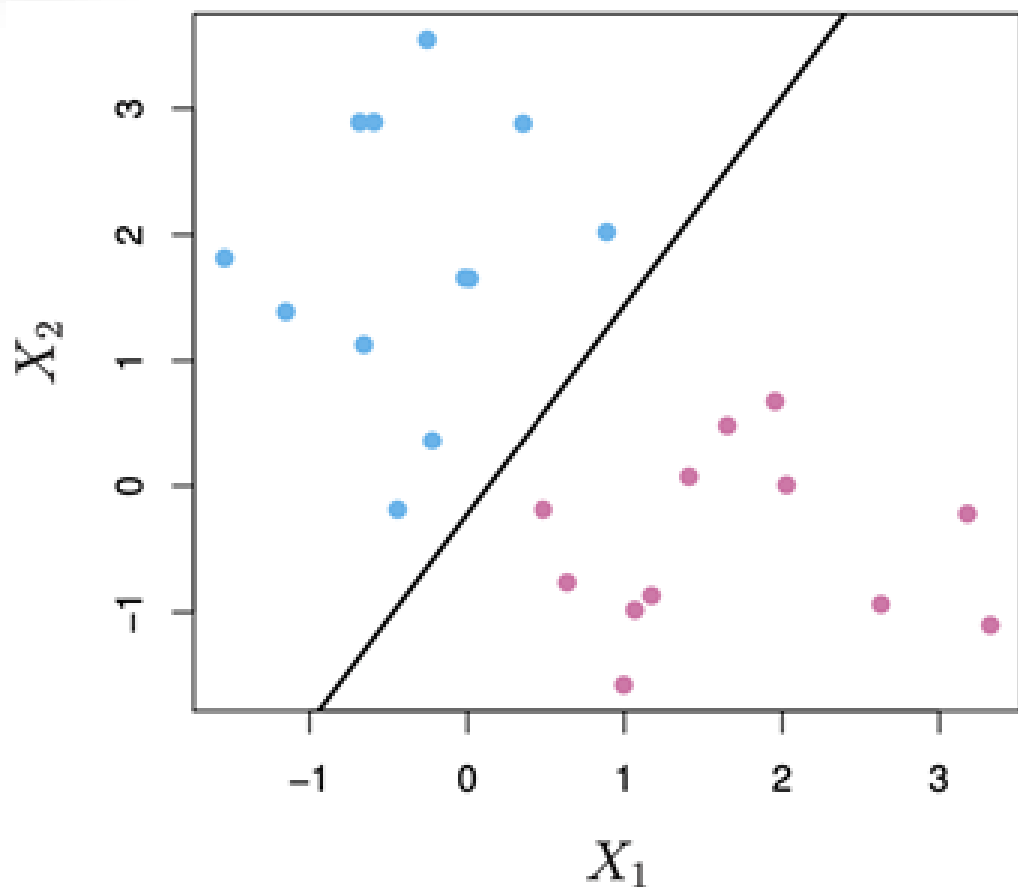
for all  $i = 1, \dots, N$ .

这可以重新表示为凸二次规划，然后有效地解出。  
包e1071中的函数svm()有效地解决了这个问题。



左边的数据不能被  
线性边界分离 ->  
几乎能分类开即可，  
**软间隔**

这是经常发生的情  
况，除非  $N < p$ 。

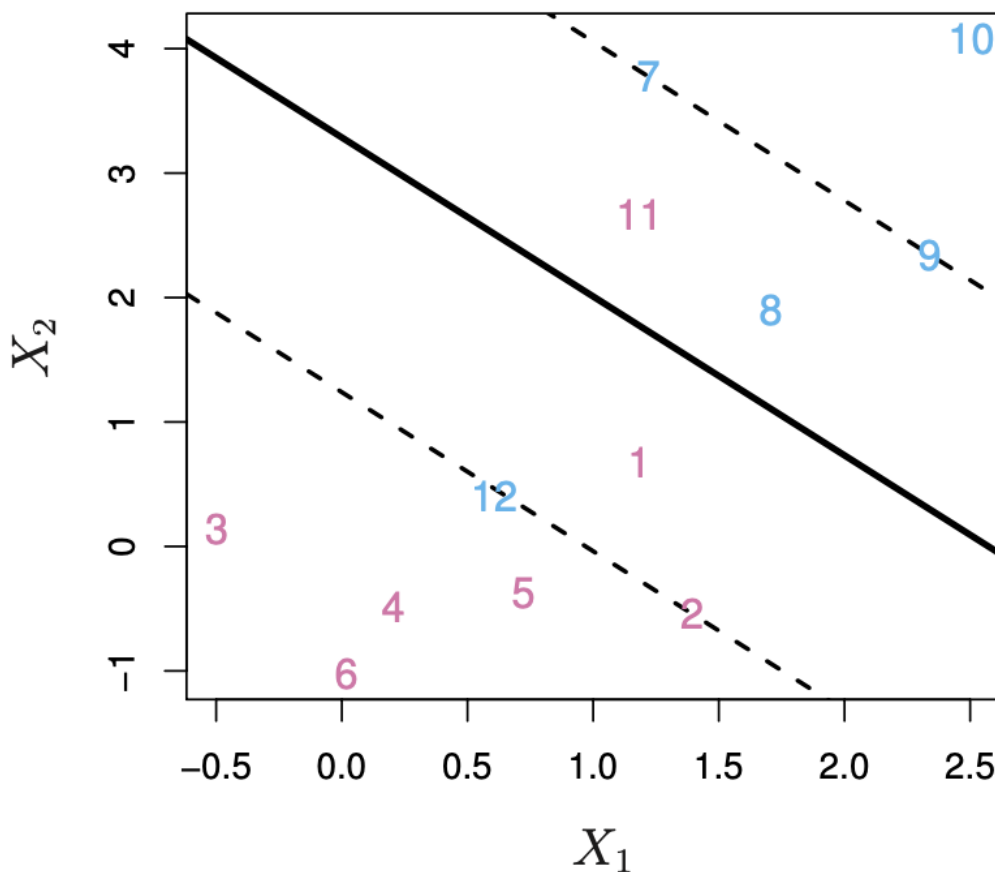
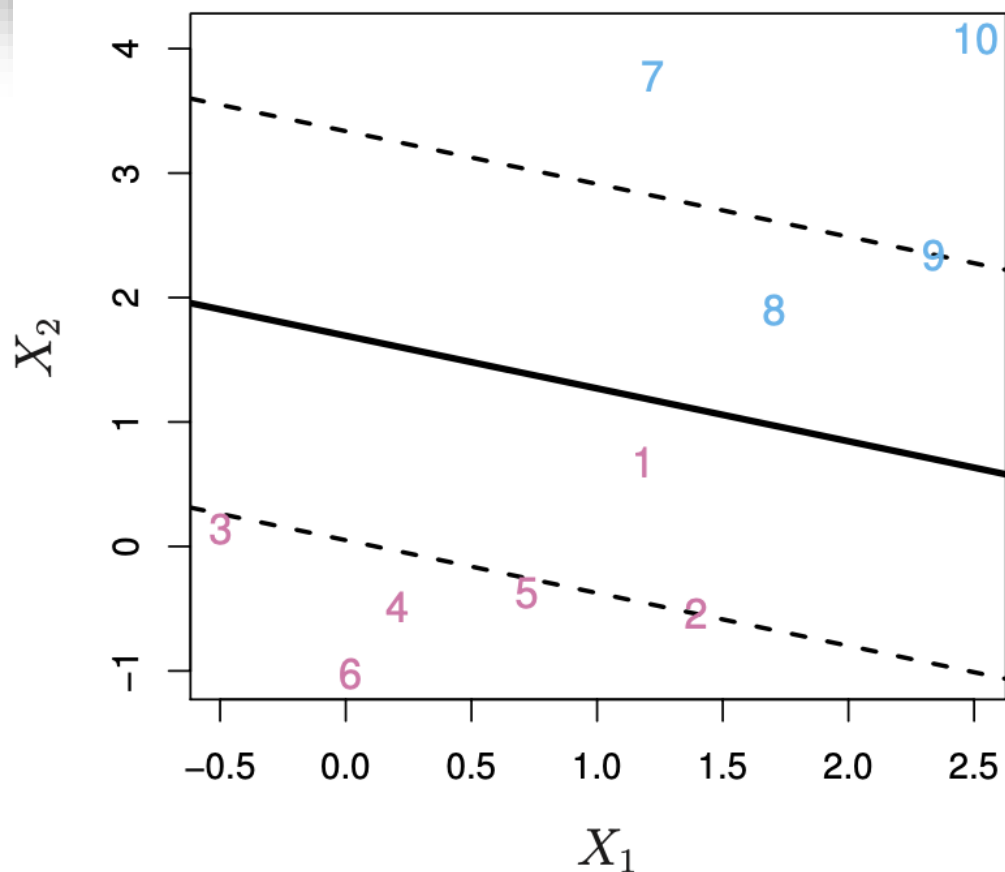


有时数据是可分离的，但噪声很大。这可能会导致最大间隔分类器的解决方案比较差。

只增加一个蓝点就让最大间隔超平面发生了巨大变化！而且间隔很小。对单个观测的变化极其敏感也说明可能过拟合了！

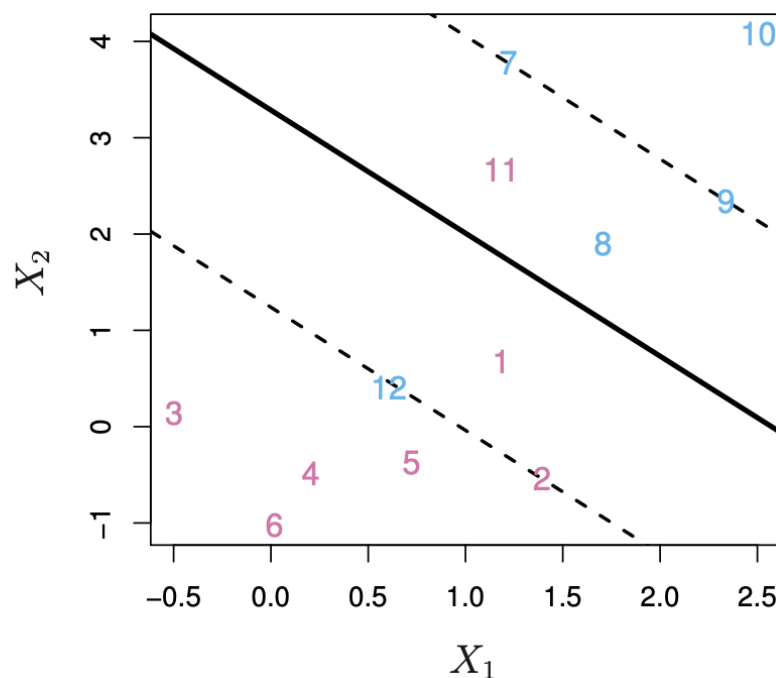
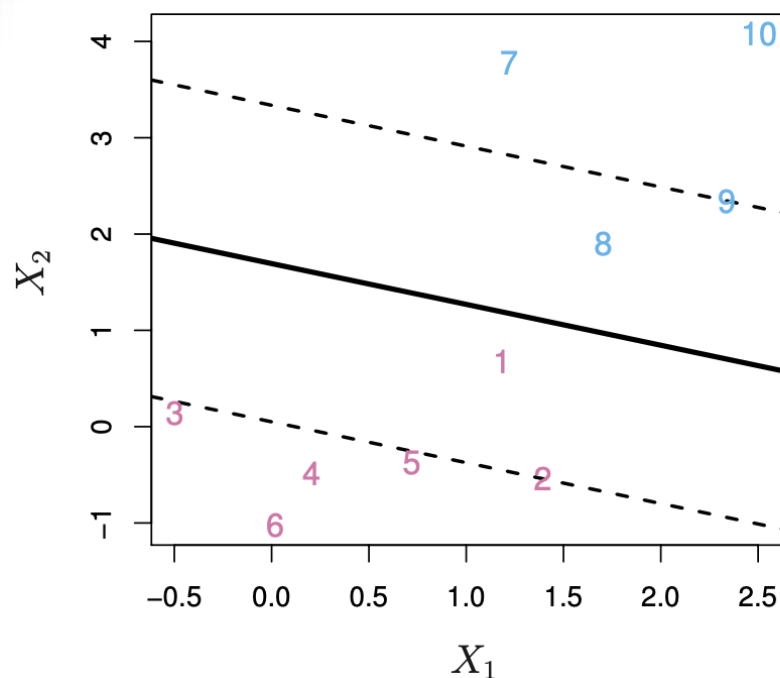


## (二) 支持向量分类器



支持向量分类器最大化了软间隔，即最大化允许了一些误分之后的间隔。→ 怎么软化？

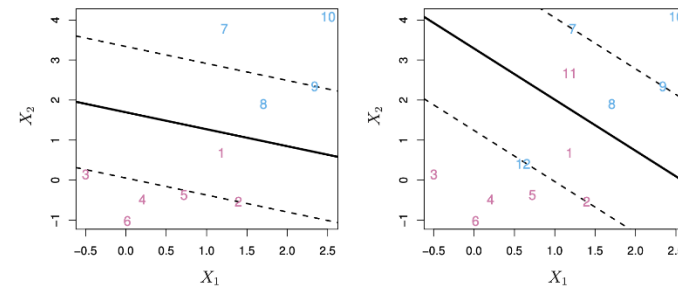




$$\underset{\beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n}{\text{maximize}} \quad M \quad \text{subject to} \quad \sum_{j=1}^p \beta_j^2 = 1,$$

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

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



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

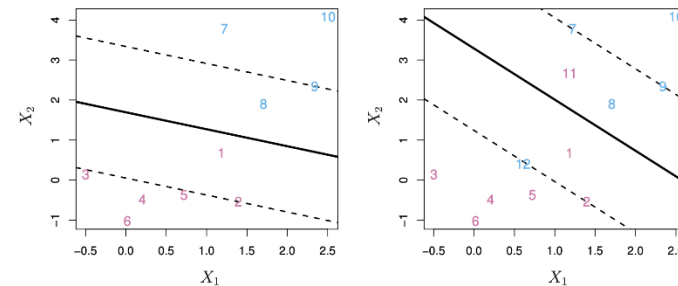
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq \underline{M(1 - \epsilon_i)},$$

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

松弛变量  $\epsilon_i$  的含义：允许间隔冲突，其值为0意味着什么？  
 $>0$ ， $>1$ 呢？

调节参数  $C$  的含义：能容忍的穿过间隔（以及超平面）  
 的观测数目的严重程度，其值为0意味着什么？ $>0$ 呢？

$C$ 通常通过交叉验证来选择。控制方差-偏差权衡。



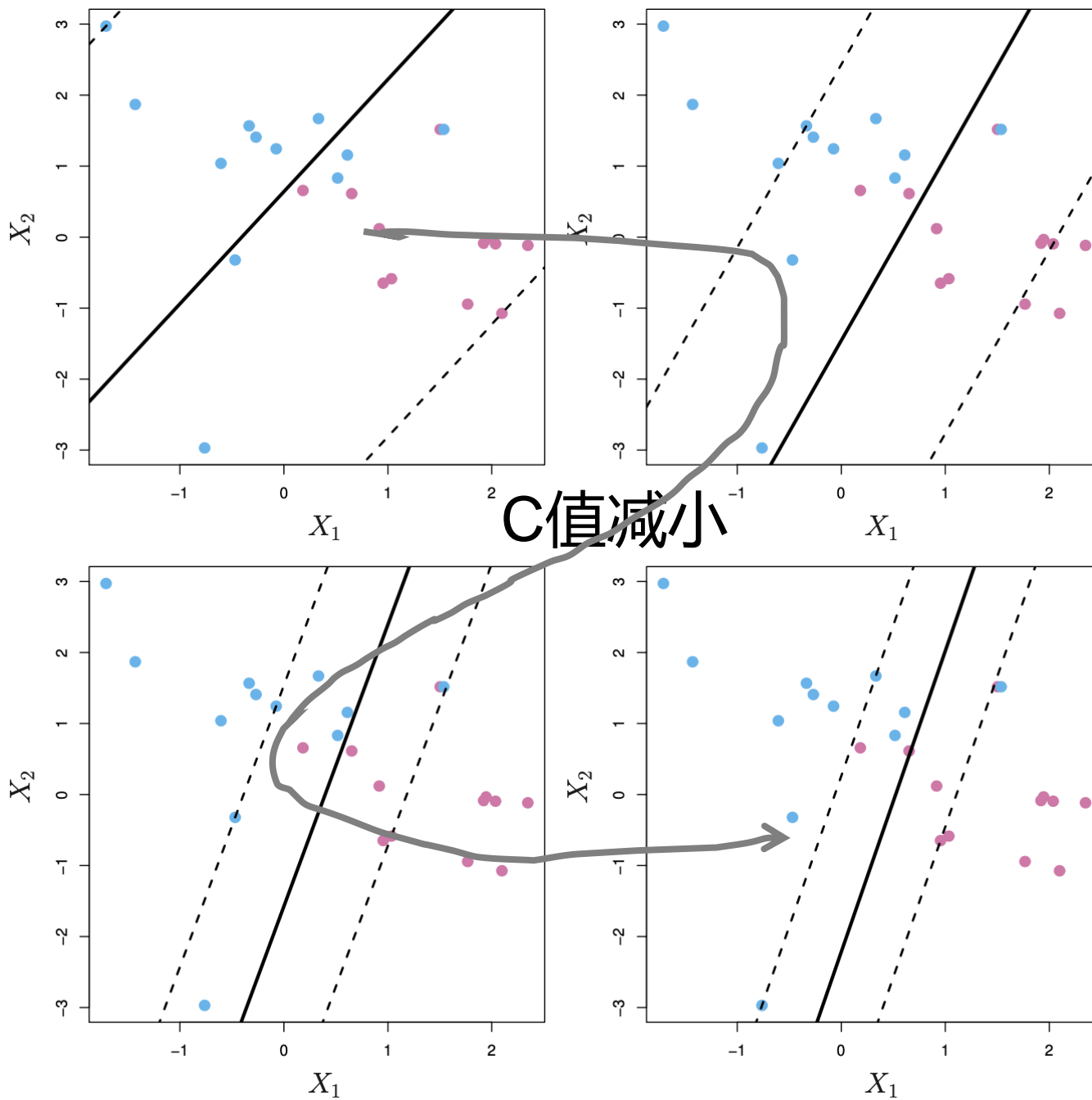
$$\underset{\beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n}{\text{maximize}} \quad M \quad \text{subject to} \quad \sum_{j=1}^p \beta_j^2 = 1,$$

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

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

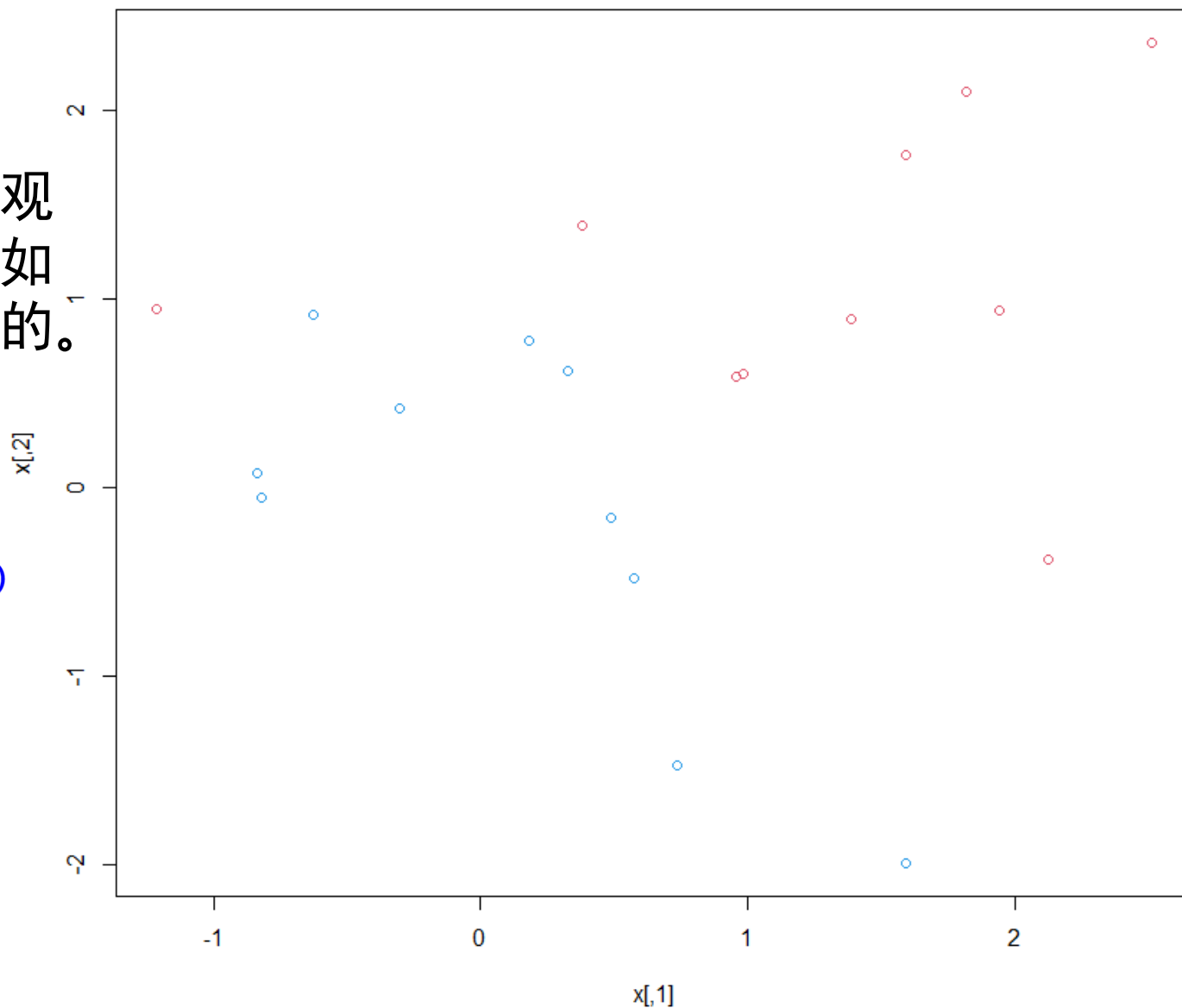
此优化问题的一个性质：只有落在间隔或穿过间隔的观测会影响超平面，根据这些观测就能得到分类器。落在正确侧正确间隔外的观测没有任何影响！

→刚好落在间隔和落在间隔错误一侧的观测叫**支持向量**！



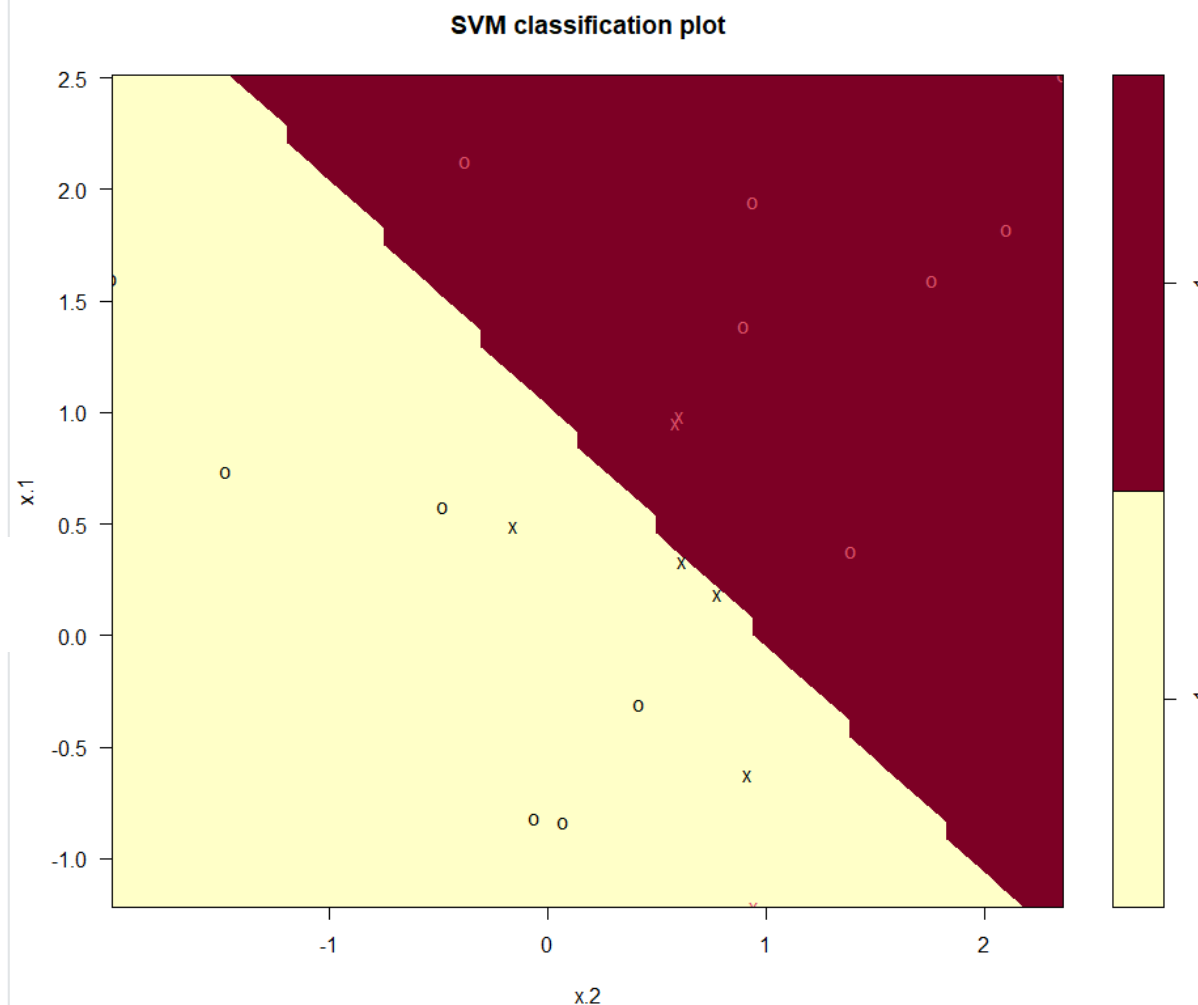
首先生成属于不同类别的两个观测并检测类别是否可分，结果如图所示：数据并不是线性可分的。

```
> set.seed(1)
> x=matrix(rnorm(20*2),ncol=2)
> y=c(rep(-1,10),rep(1,10))
> x[y==1,]=x[y==1,]+1
> plot(x,col=(3-y))
```



首先将响应向量转换成因子变量，并用svm()函数进行分类，然后画出得到的支持向量分类器，最后查看支持向量。

```
> dat=data.frame(x=x,y=as.factor(y))
> library(e1071)
> svmfit=svm(y~.,data=dat,kernel
="linear",cost=10,scale=FALSE)
> plot(svmfit,dat)
> svmfit$index
[1] 1 2 5 7 14 16 17
```



使用summary ()命令可以获得支持向量分类器拟合的一些基本信息:

```
> summary(svmfit)
```

Call:

```
svm(formula = y ~ ., data = dat,  
     kernel = "linear", cost = 10,  
     scale = FALSE)
```

Parameters:

```
SVM-Type: C-classification  
SVM-Kernel: linear  
cost: 10
```

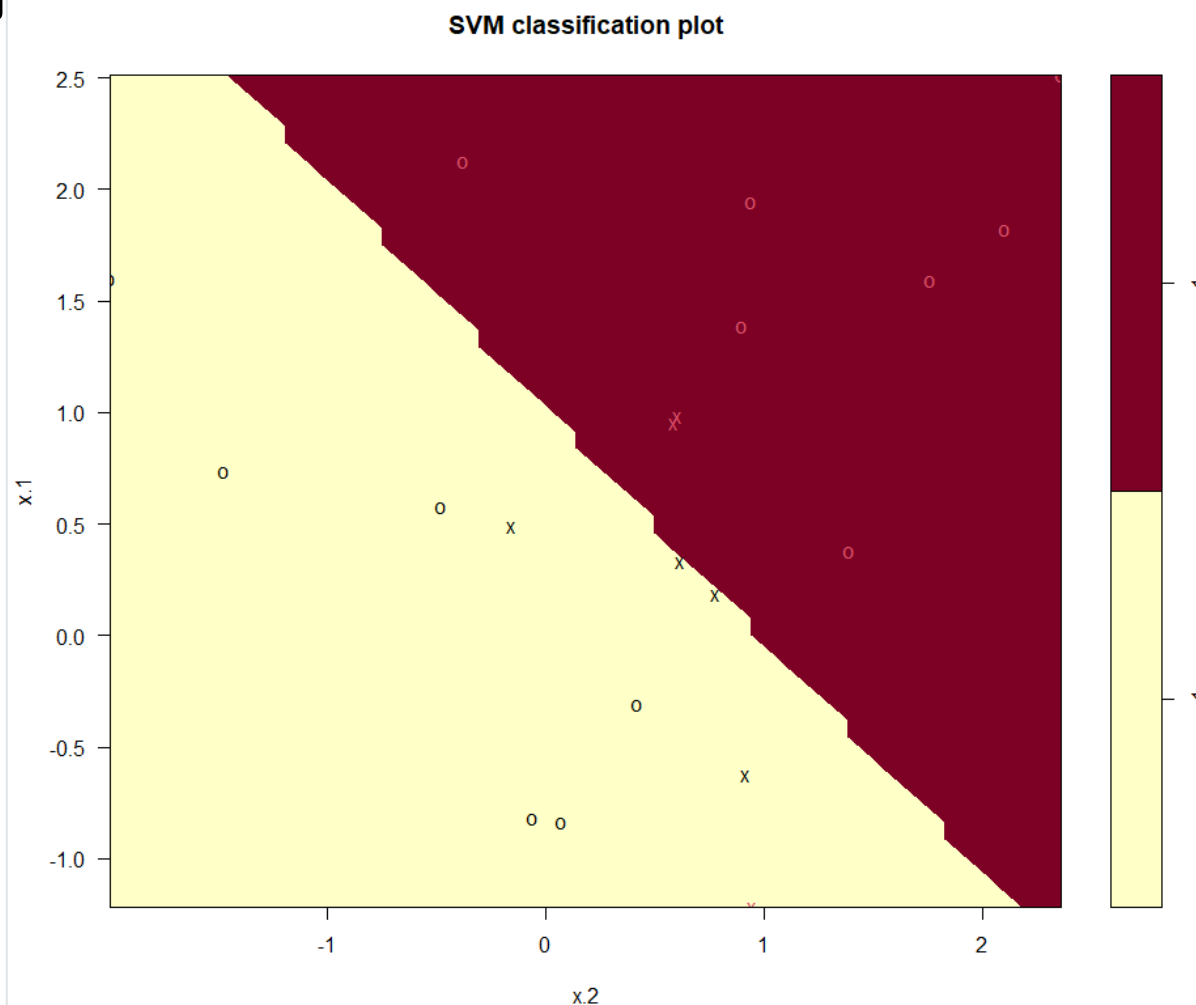
Number of Support Vectors: 7

```
( 4 3 )
```

Number of Classes: 2

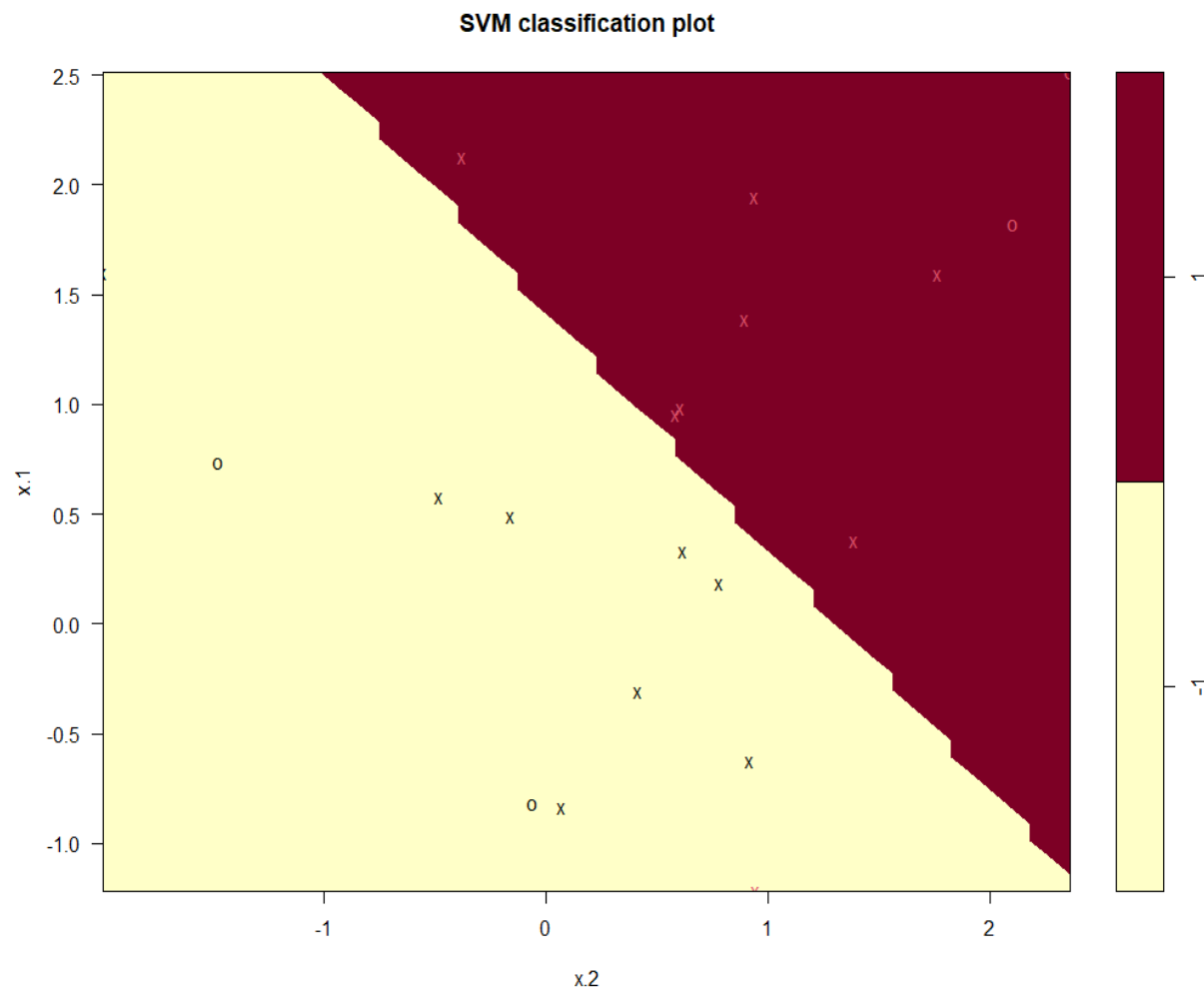
Levels:

```
-1 1
```

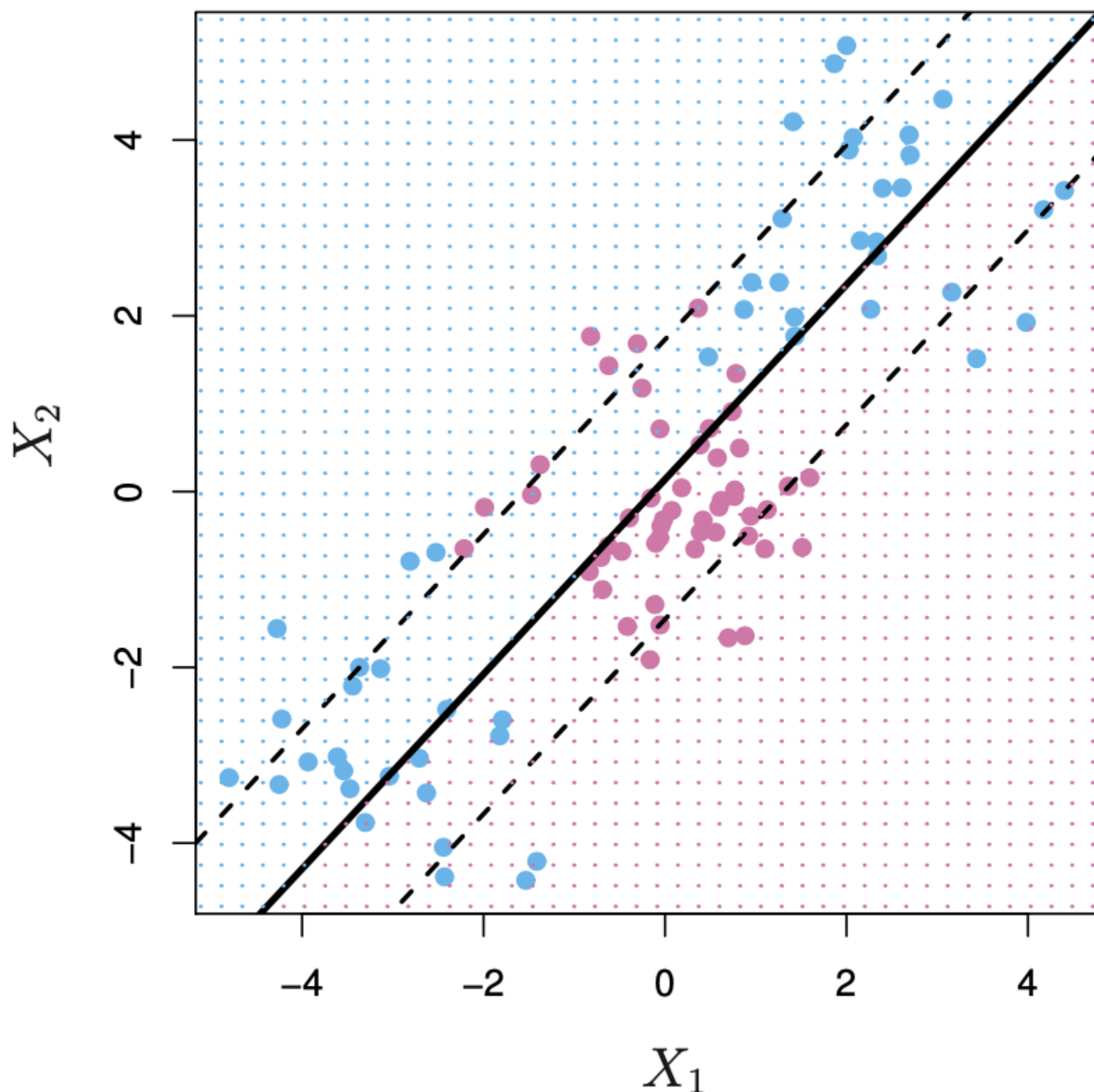


将cost值修改为0.1,  
得到更多支持向量

```
> svmfit=svm(y~.,data=dat,kernel  
="linear",cost=0.1,scale=FALSE)  
> plot(svmfit,dat)  
> svmfit$index  
[1] 1 2 3 4 5 7 9 10  
[9] 12 13 14 15 16 17 18 20
```







有时线性边界根本不起作用，不管C值是多少。

左边的例子就是这样一种情况。

怎么办？

- 通过把样本变换包含进来来扩大特征空间, 例如  $X_1^2, X_1^3, X_1X_2, X_1X_2^2, \dots$  因此从一个  $p$  维空间到一个  $M > p$  维空间。
- 在扩大的空间中拟合一个支持向量分类器。
- 这导致了在原始空间中的非线性决策边界。

- 通过把样本变换包含进来来扩大特征空间，例如  $X_1^2, X_1^3, X_1X_2, X_1X_2^2, \dots$  因此从一个  $p$  维空间到一个  $M > p$  维空间。
- 在扩大的空间中拟合一个支持向量分类器。
- 这导致了在原始空间中的非线性决策边界。

示例:

假设我们使用  $(X_1, X_2, X_1^2, X_2^2, X_1X_2)$  而不仅是  $(X_1, X_2)$ 。  
那么决策边界就会是这样的

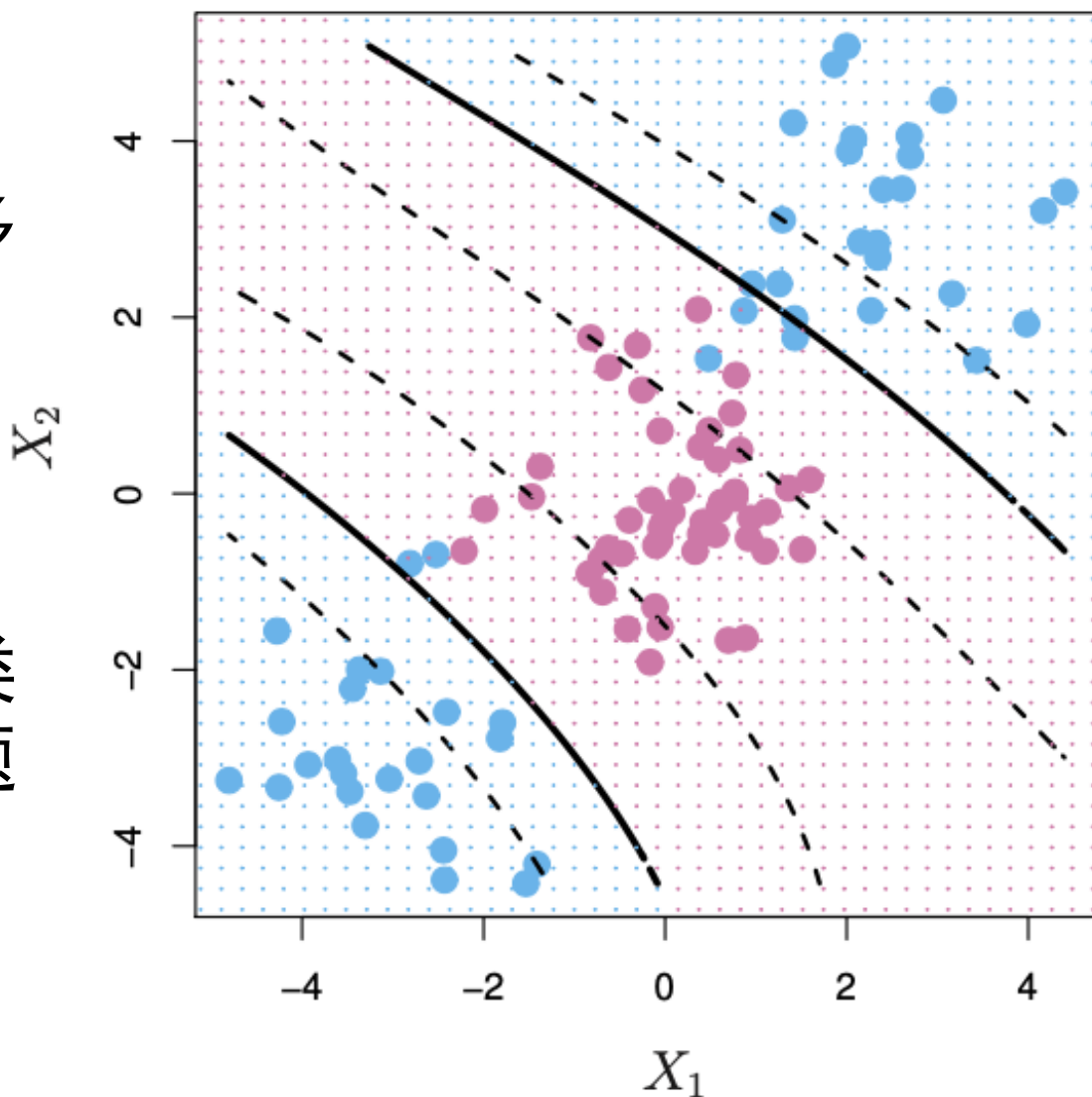
$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1 x_2 = 0$$

这导致了原始空间(二次圆锥截面)的非线性决策边界。

这里我们使用一组三次多项式的基展开

从2个变量到9个

扩大空间的支持向量分类器解决了低维空间的问题



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

- 多项式 (特别是高维的) 非常快地变得不可控。
- 有一个更优雅和可控的方法来引入非线性到支持向量分类器 - 通过核函数的使用。
- 在我们讨论这些之前, 我们必须了解内积在支持向量分类器中的作用。

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

向量内积

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

向量内积

- 线性支持向量分类器可以表示为

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

n个参数

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

向量内积

- 线性支持向量分类器可以表示为

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

n个参数

- 估计参数  $\alpha_1, \dots, \alpha_n$  和  $\beta_0$ , 我们只需要  $\binom{n}{2}$  个内积  $\langle x_i, x_{i'} \rangle$ , 在所有成对的训练观察之间。



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

向量内积

- 线性支持向量分类器可以表示为

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

n个参数

- 估计参数  $\alpha_1, \dots, \alpha_n$  和  $\beta_0$ ，我们只需要  $\binom{n}{2}$  个内积  $\langle x_i, x_{i'} \rangle$ ，在所有成对的训练观察之间。

- 其实，大部分训练观测的  $\hat{\alpha}_i$  为零，只有支持向量对应的非零：

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

$S$  是使  $\hat{\alpha}_i > 0$  的索引  $i$  的支持集。

能否一般化？



### (三) 核函数和支持向量机

- 如果我们计算观测值之间的内积，就可以拟合支持向量(SV)分类器。可以相当抽象！
- 一些特殊的函数 $K$ 可以为我们做到这一点 - **核函数 (kernel)**。  
如多项式核函数：

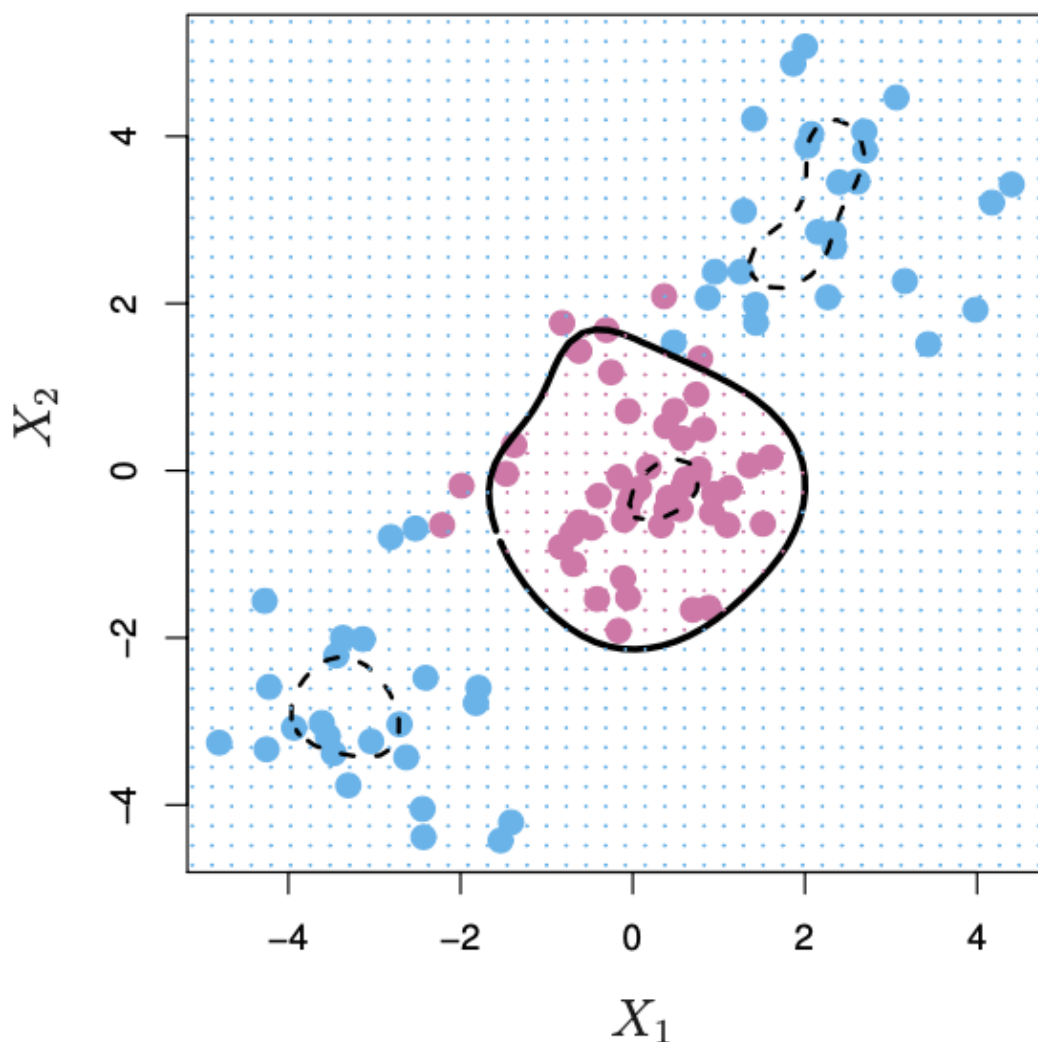
$$K(x_i, x_{i'}) = \left( 1 + \sum_{j=1}^p x_{ij} x_{i'j} \right)^d$$

计算  $d$  (正整数) 维多项式所需的内积 -  $\binom{p+d}{d}$  个基函数!  
对  $p=2$  和  $d=2$  试试看。

- 解有这样的形式 
$$f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i K(x, x_i)$$

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

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



如果测试观测距离训练观测非常远，核函数值会很小，对预测的影响就很小。

隐式特征空间。很高的维度。  
控制方差，通过极力压缩最多的维度。

$$K(x_i, x_{i'}) = \left( 1 + \sum_{j=1}^p x_{ij} x_{i'j} \right)^d \quad K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})^2)$$

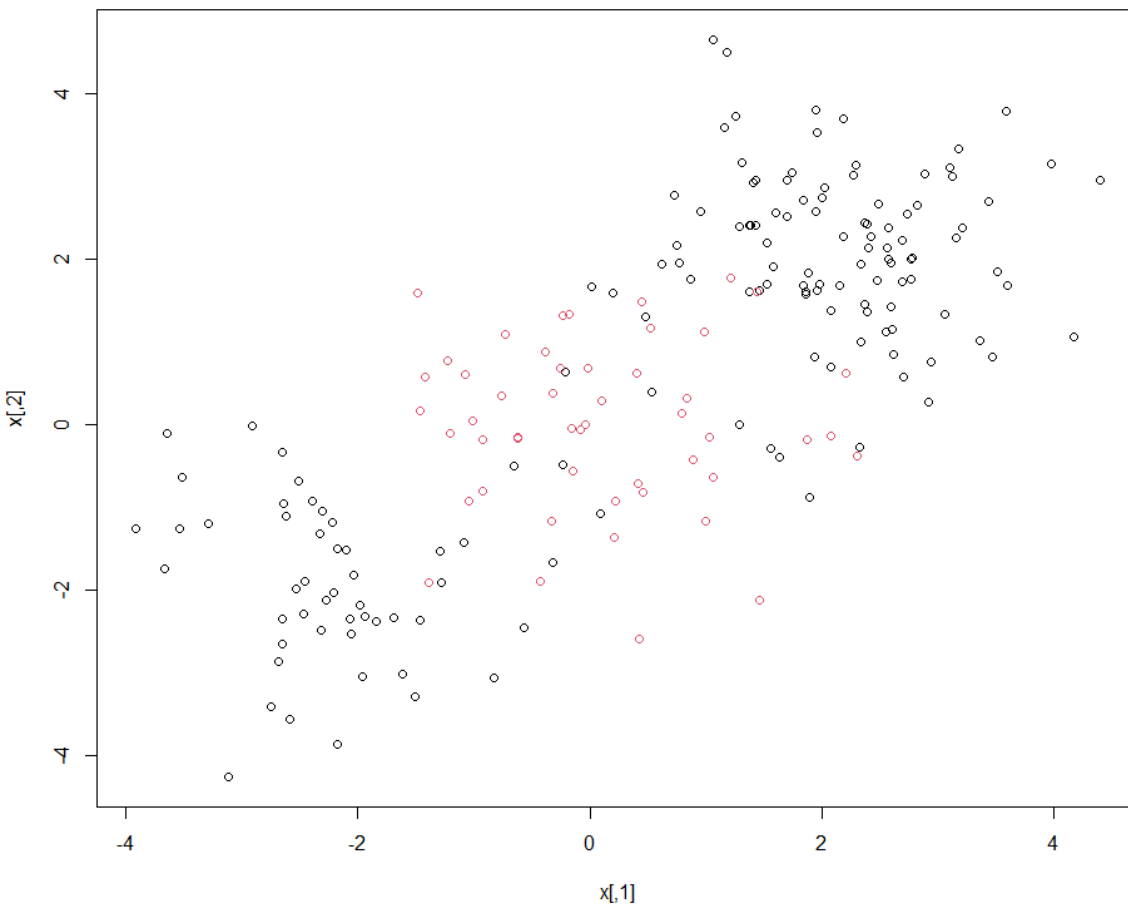
$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i)$$

使用核函数，只需为 $\binom{n}{2}$ 个不同样本配对计算核函数K

vs.

单纯的扩大特征空间的方法，没有明确的计算量！

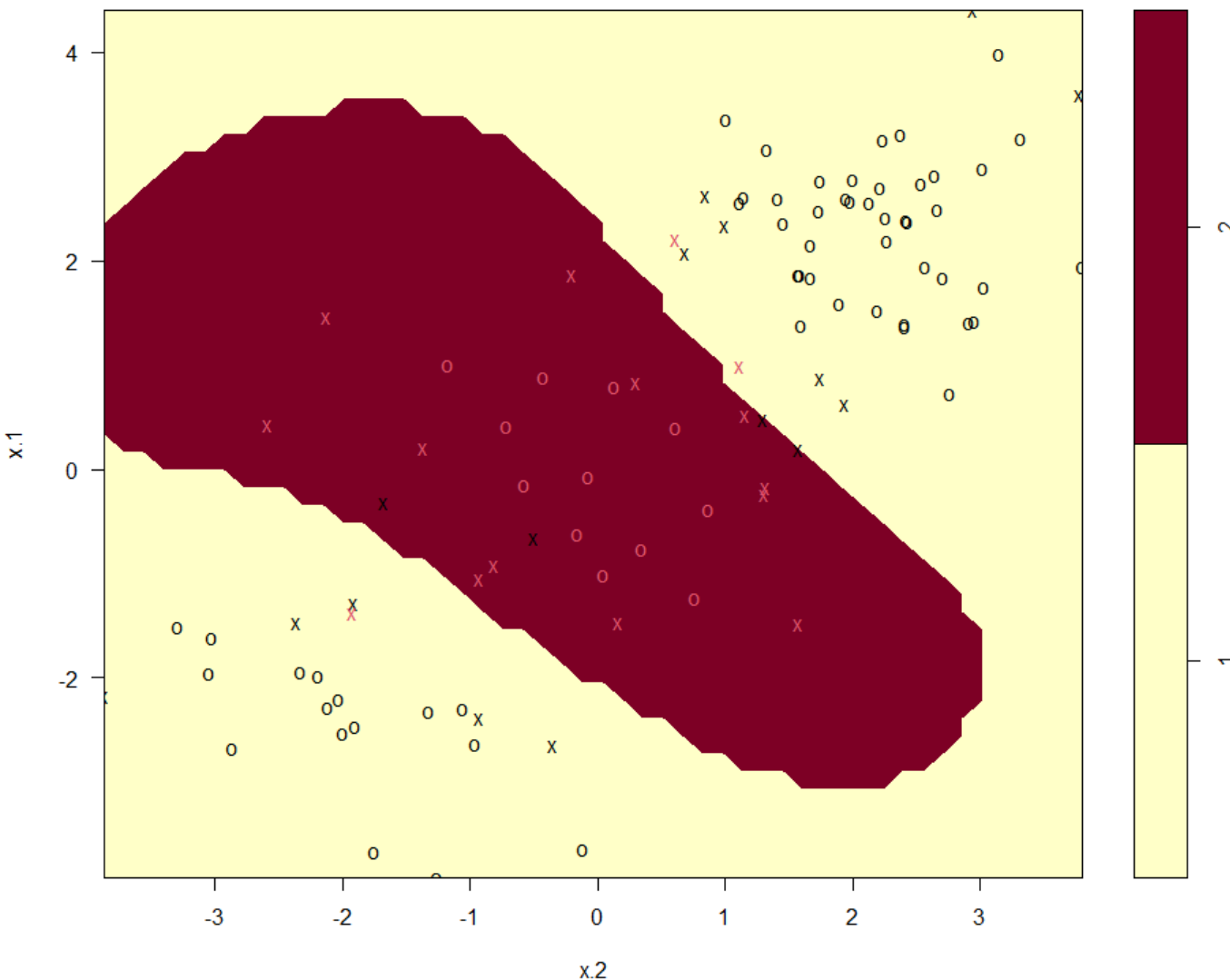
```
> set.seed(1)
> x=matrix(rnorm(200*2),ncol=2)
> x[1:100,]=x[1:100,]+2
> x[101:150,]=x[101:150,]-2
> y=c(rep(1,150),rep(2,50))
> dat=data.frame(x=x,y=as.factor(y))
> plot(x,col=y)
```



使用代码生成一些具有非线性类别边界的数据。  
得到的结果如图所示：类别边界是非线性的。

```
> train=sample(200,100)
> svmfit=svm(y~.,data=dat[train,],kernel="radial",gamma=1,cos
t=1)
> plot(svmfit,dat[train,])
```

SVM classification plot



使用 `svm()` 函数拟合非线性核函数的 SVM，其中 `kernel` 设置为 `radial` 拟合径向基核函数，并设置 `gamma` 的值为 1。

得到非线性的决策边界。

```
> summary(svmfit)
```

```
Call:
svm(formula = y ~ ., data = dat
[train,
      ], kernel = "radial",
      gamma = 1, cost = 1)
```

```
Parameters:
  SVM-Type:  C-classification
SVM-Kernel:  radial
      cost:  1
```

```
Number of Support Vectors:  31
```

```
( 16 15 )
```

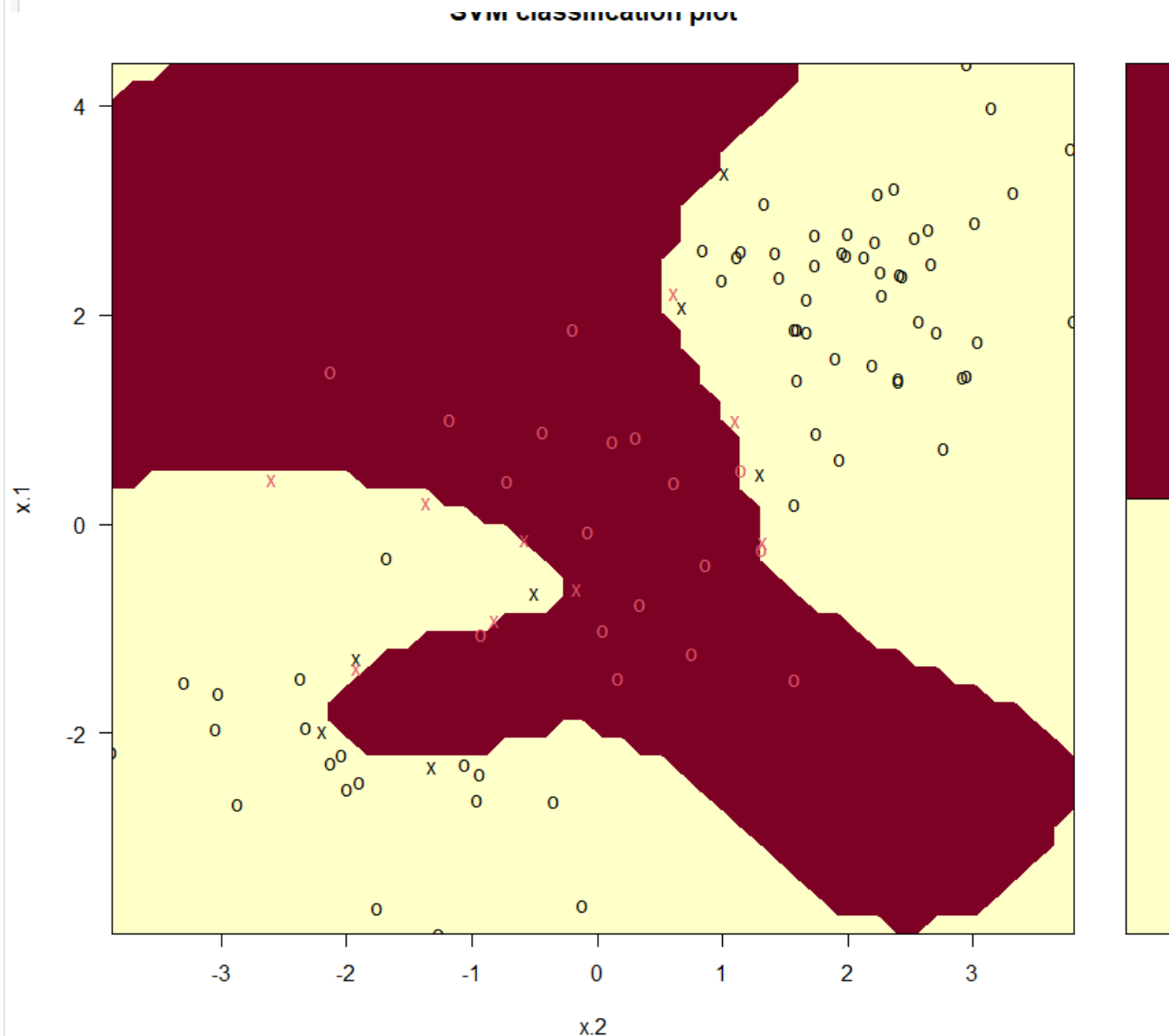
```
Number of Classes:  2
```

```
Levels:
 1  2
```

利用summary()函数  
查看svm拟合的信  
息。



```
> svmfit=svm(y~.,data=dat[trai  
n,],kernel="radial",gamma=1,cost  
=1e5)
```



增大cost的值可以减小误差，但是会有过拟合的风险

得到非线性的决策边界更为不规则。

```
> set.seed(1)
> tune.out=tune(svm,y~.,data=dat[train,],kernel
="radial",ranges=list(cost=c(0.1,1,10,100,100),
gamma=c(0.5,1,2,3,4)))
> summary(tune.out)
```

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:

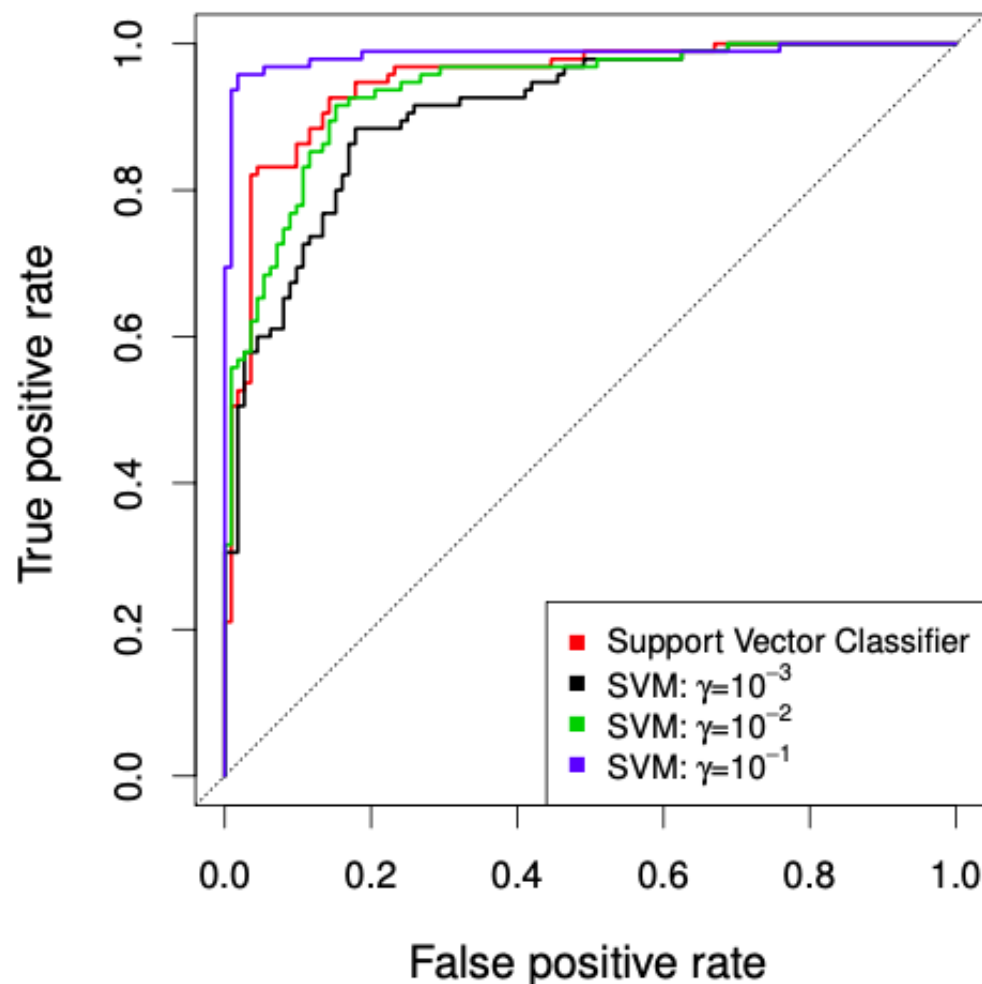
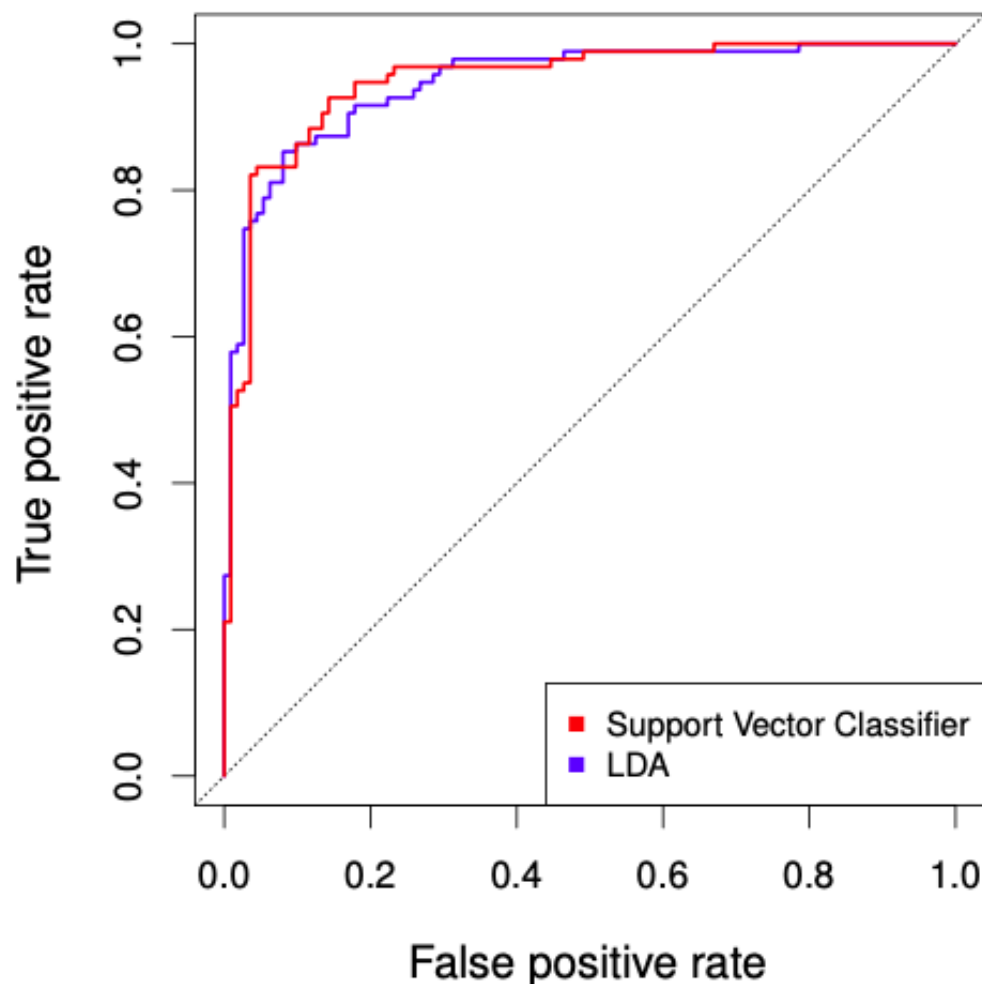
cost	gamma
1	0.5

- best performance: 0.07

- Detailed performance results:

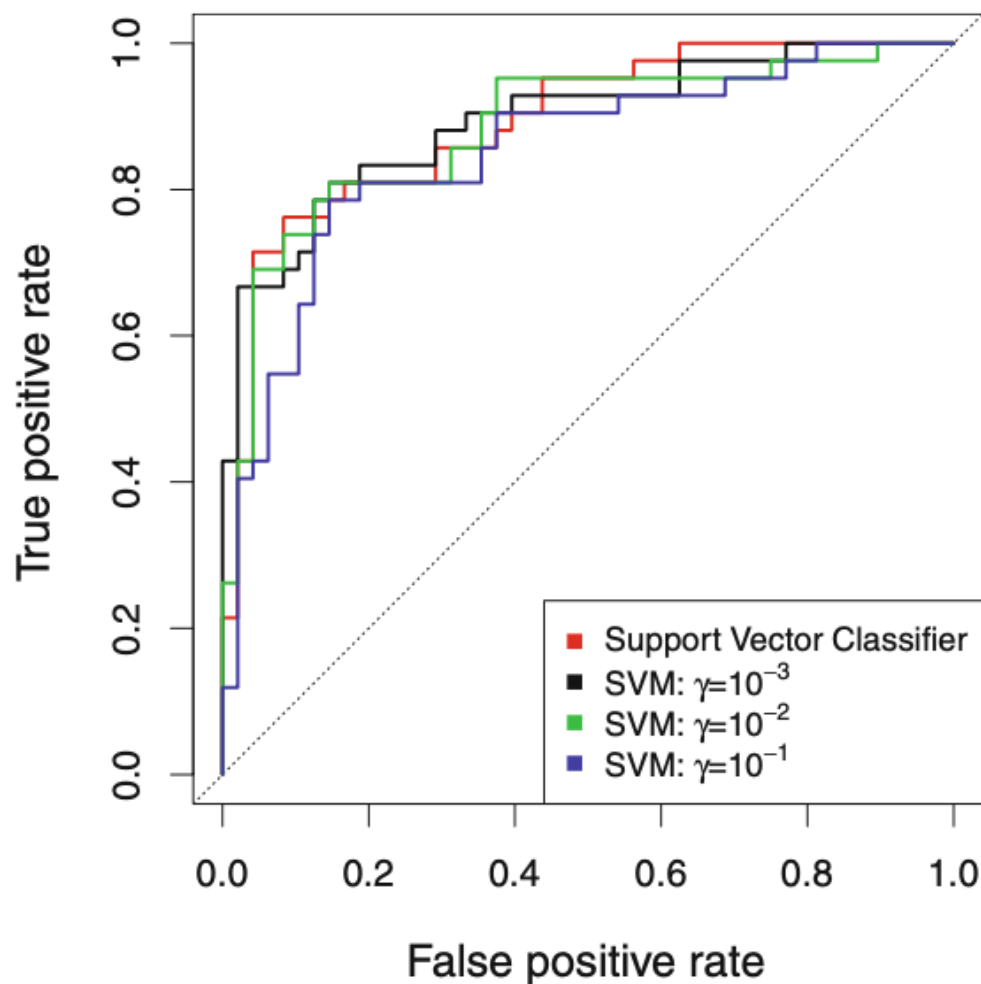
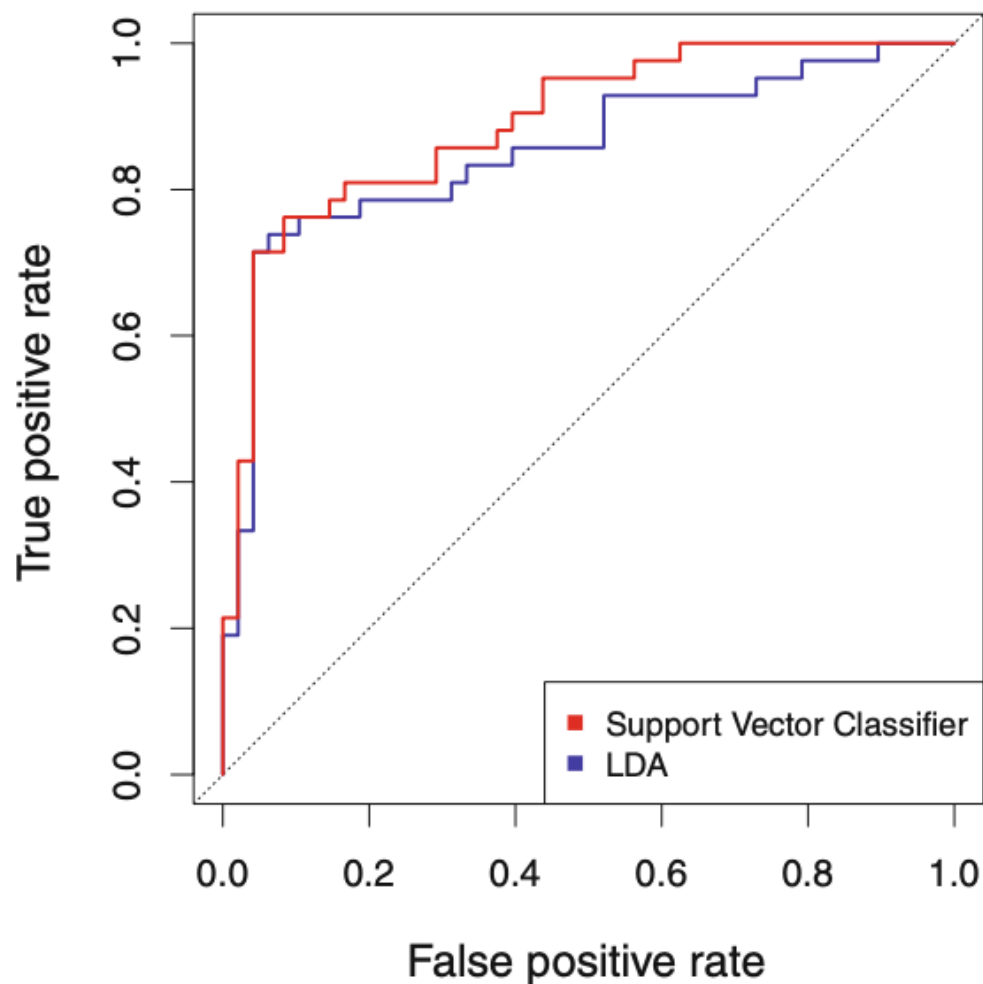
	cost	gamma	error
1	0.1	0.5	0.26
2	1.0	0.5	0.07

使用tune()函数选择  
径向核函数的最优  
gamma值以及cost值,  
结果如图所示为  
cost=1,gamma=0.5。



训练集ROC曲线。

左：与LDA比较；右：SVC与SVM比较。  
SVM选的径向核函数，参数  $\gamma$  如图示。



测试集ROC曲线。

左：与LDA比较；右：SVC与SVM比较。  
SVM选的径向核函数，参数  $\gamma$  如图示。

```
> library(ROCR)
> rocplot=function(pred, truth, ...){
+   predob = prediction(pred, truth)
+   perf = performance(predob, "tpr", "fpr")
+   plot(perf,...)}
```

做一个用预测值和真实值画图的函数

```
> svmfit.opt=svm(y~., data=dat[train,], kernel="radial",
+               gamma=2, cost=1, decision.values=T)
> fitted=attributes(predict(svmfit.opt, dat[train,], decision.
+                           values=TRUE))$decision.values
```

SVM拟合、预测

```
> par(mfrow=c(1,2))
> rocplot(fitted, dat[train, "y"], main="Training Data")
```

调用前面定义的函数画图

```
> svmfit.flex=svm(y~., data=dat[train,], kernel="radial",
+                 gamma=50, cost=1, decision.values=T)
> fitted=attributes(predict(svmfit.flex, dat[train,], decision.
+                           values=T))$decision.values
> rocplot(fitted, dat[train, "y"], add=T, col="red")
```

增加gamma，再画图

```
> fitted=attributes(predict(svmfit.opt, dat[-train,], decision.
+                           values=T))$decision.values
> rocplot(fitted, dat[-train, "y"], main="Test Data")
> fitted=attributes(predict(svmfit.flex, dat[-train,], decision.
+                           values=T))$decision.values
> rocplot(fitted, dat[-train, "y"], add=T, col="red")
```

测试集画图



---

## (四) 多分类

---

- 按定义SVM适用于  $K = 2$  分类。那  $K > 2$  分类呢?

- 按定义SVM适用于  $K = 2$  分类。那  $K > 2$  分类呢?

**OVA** One Vs. All, 1对所有。也可写作One Vs. Rest (OVR)  
拟合  $K$  个不同的2分类SVM分类器  $\hat{f}_k(x)$ ,  $k = 1, \dots, K$ ; 然后各类对其余。  $x^*$  分类到  $\hat{f}_k(x^*)$  最大的那个类。



- 按定义SVM适用于  $K = 2$  分类。那  $K > 2$  分类呢?

**OVA** One Vs. All, 1对所有。也可写作One Vs. Rest (OVR)  
拟合  $K$  个不同的2分类SVM分类器  $\hat{f}_k(x)$ ,  $k = 1, \dots, K$ ; 然后各类对其余。  $x^*$  分类到  $\hat{f}_k(x^*)$  最大的那个类。

**OVO** One Vs. One, 1对1。

拟合所有  $\binom{K}{2}$  成对分类器  $\hat{f}_{kl}(x)$ 。  $x^*$  分类到赢得最多两两对决的类。

•按定义SVM适用于  $K = 2$  分类。那  $K > 2$  分类呢?

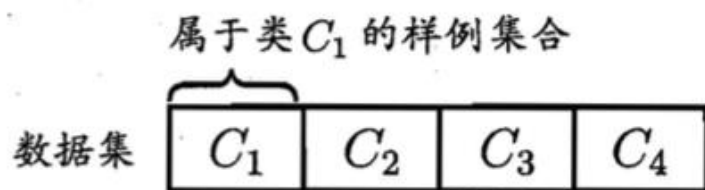
**OVA** One Vs. All, 1对所有。也可写作One Vs. Rest (OVR)  
拟合  $K$  个不同的2分类SVM分类器  $\hat{f}_k(x)$ ,  $k = 1, \dots, K$ ; 然后各类对其余。  $x^*$  分类到  $\hat{f}_k(x^*)$  最大的那个类。

**OVO** One Vs. One, 1对1。

拟合所有  $\binom{K}{2}$  成对分类器  $\hat{f}_{kl}(x)$ 。  $x^*$  分类到赢得最多两两对决的类。

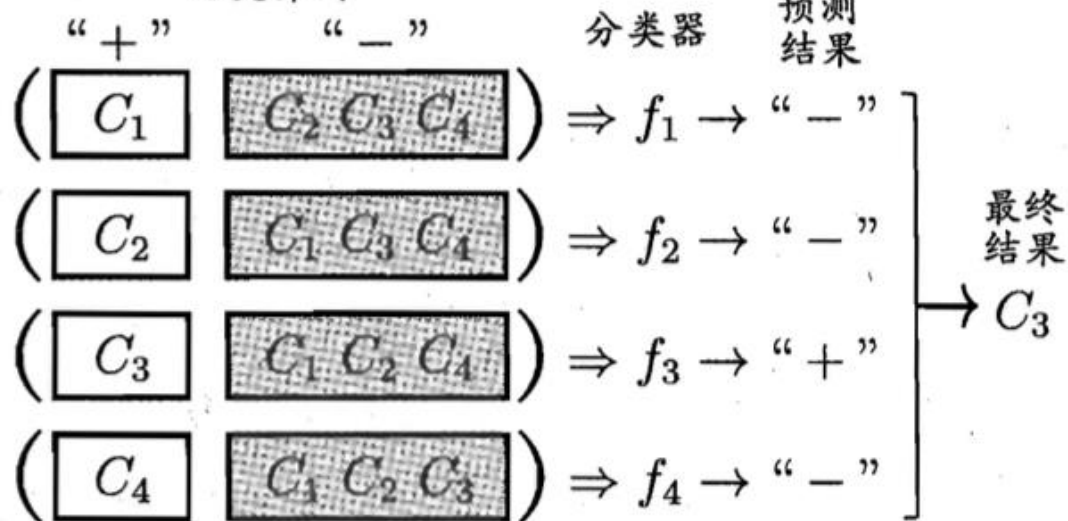
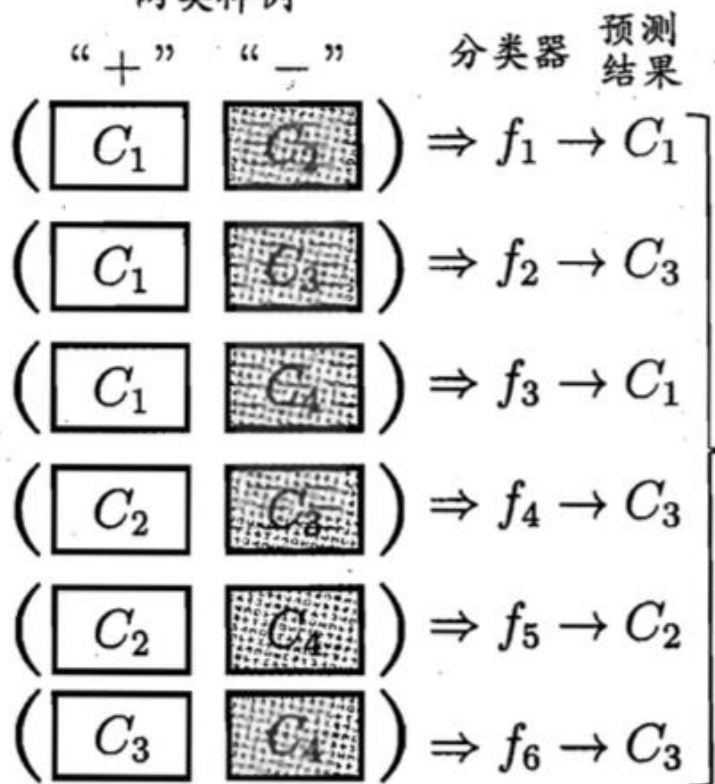
选择哪一个? 如果  $K$  不是太大, 就用OVO。

## OVO与OVA (OVR) 示意图



OvO

OvR

用于训练的  
两类样例用于训练的  
两类样例

	$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	海明距离 ↓	欧氏距离 ↓
$C_1 \rightarrow$	-1	+1	-1	+1	+1	3	$2\sqrt{3}$
$C_2 \rightarrow$	+1	-1	-1	+1	-1	4	4
$C_3 \rightarrow$	-1	+1	+1	-1	+1	1	2
$C_4 \rightarrow$	-1	-1	+1	+1	-1	2	$2\sqrt{2}$
测试示例 $\rightarrow$	-1	-1	+1	-1	+1	↑	↑

(a) 二元 ECOC 码

	$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$	$f_7$	海明距离 ↓	欧氏距离 ↓
$C_1 \rightarrow$	-1	-1	+1	+1	-1	+1	+1	4	4
$C_2 \rightarrow$	-1				+1	-1		2	2
$C_3 \rightarrow$	+1	+1	-1	-1	-1	+1	-1	5	$2\sqrt{5}$
$C_4 \rightarrow$	-1	+1		+1	-1		+1	3	$\sqrt{10}$
测试示例 $\rightarrow$	-1	+1	+1	-1	+1	-1	+1	↑	↑

(b) 三元 ECOC 码

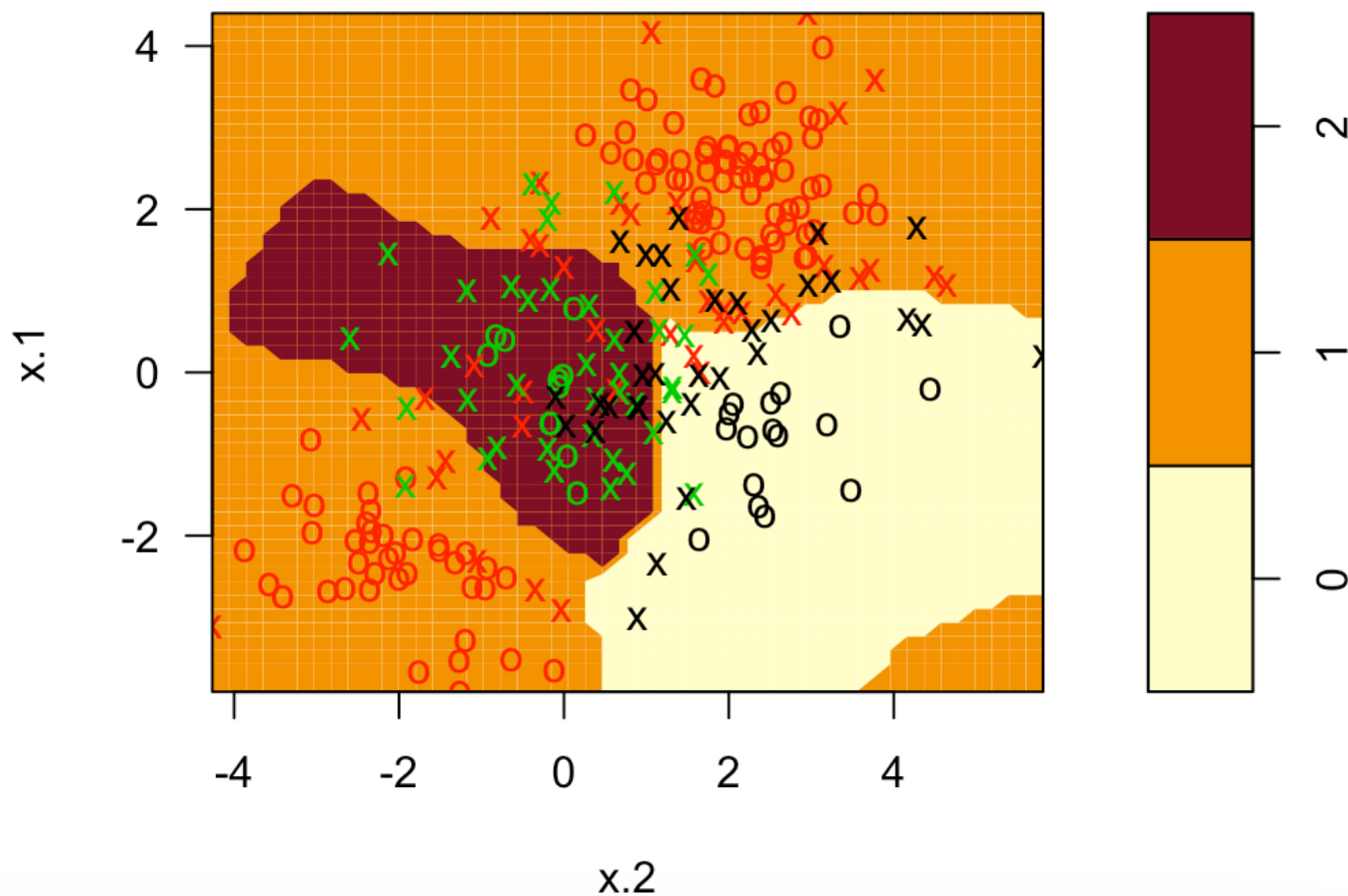
使用ECOC编码（Error Correcting Output Codes, 纠错输出码, 1995年提出）

- 1) 编码：对N个类别做M次划分，每次划分将一部分类划为正类，一部分划为反类，形成二分类训练集，训练出M个分类器。
- 2) 解码：对测试样本分类，返回距离最小的作为预测结果。

```
> set.seed(1)
> x=rbind(x, matrix(rnorm(50*2), ncol=2))
> y=c(y, rep(0,50))
> x[y==0,2]=x[y==0,2]+2
> dat=data.frame(x=x, y=as.factor(y))
> par(mfrow=c(1,1))
> plot(x,col=(y+1))
```

```
> svmfit=svm(y~., data=dat, kernel="radial", cost=10, gamma=1)
> plot(svmfit, dat)
```

**SVM classification plot**



svm()使用OVO,  
3分类



## (五) 与逻辑斯谛回归的关系

- 用  $f(X) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p$  能将支持向量分类器优化再表达为

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

**$\lambda$ 大小变化的含义?**

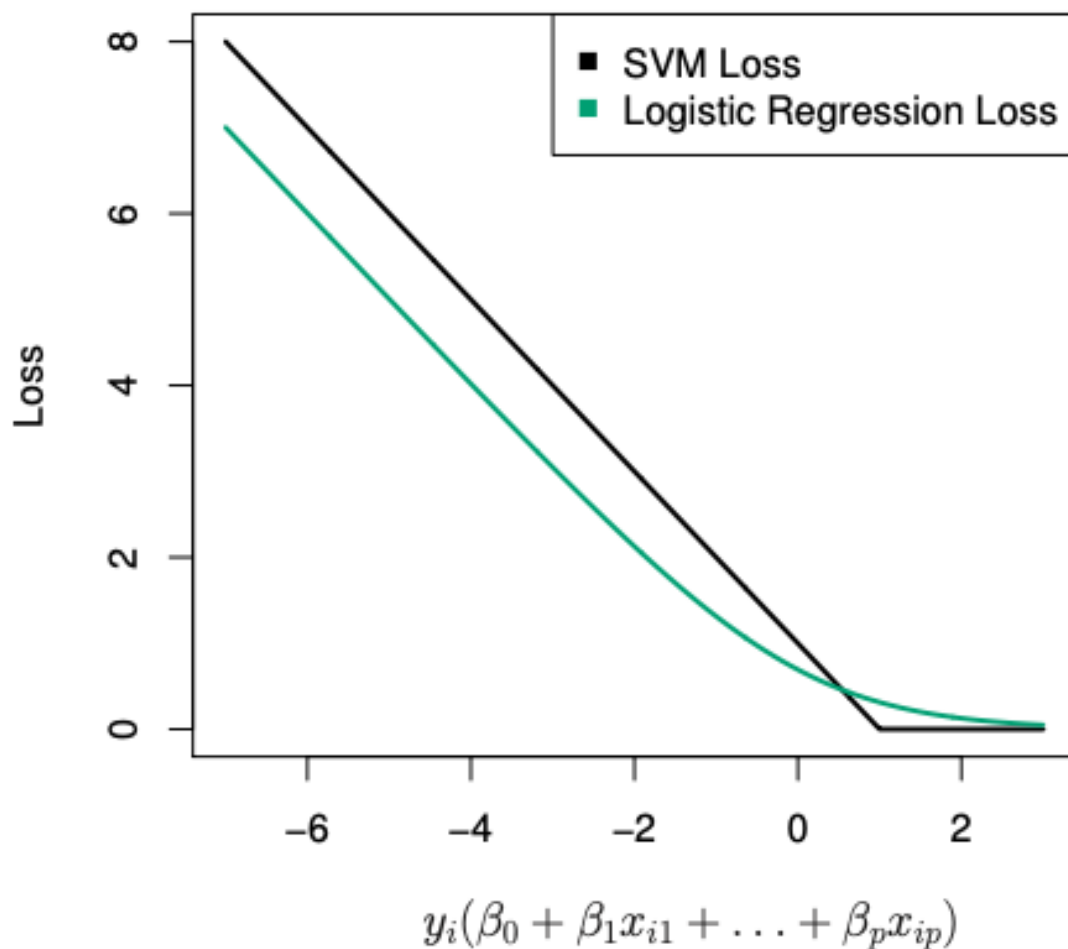
形式为：损失函数 + 惩罚项

$$L(\mathbf{X}, \mathbf{y}, \beta) = \sum_{i=1}^n \max [0, 1 - y_i (\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip})]$$

这个损失函数的形式为Hinge损失(铰链)。非常类似于logistic回归中的“损失”(负对数似然)。

- 用  $f(X) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p$  能将支持向量分类器优化再表达为

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





- 当类别(几乎)可分离时, SVM比LR做得更好。LDA也是如此。
- 不可分离的时候, LR比较合适。
- 如果你想估计概率, 选择LR。
- 对于非线性边界, 核支持向量机很受欢迎。也可以使用核函数连同LR和LDA, 但计算开销更大。

e1071 库能够实现许多统计学习方法。特别是，如果在参数设置中使用 `kernel="linear"`，`svm()` 函数可以用来拟合支持向量分类器，其中的 `cost` 参数用来设置观测穿过间隔的成本。

如果 `cost` 参数值较小，那么间隔就会很宽，许多支持向量会落在间隔上或者穿过间隔。如果 `cost` 参数值较大，那么间隔就会很窄，更少支持向量会落在间隔上或者穿过间隔。