Lecture10
支持向量机

刘晨辉
邮箱：chenhuiliu@hnu.edu.cn
办公室：土木楼A422
2023.04.25

目录

1. 最大边际分类器(Maximal Margin Classifier)

在1个p维的空间中，一个超平面（Hyperplane）是一个$(p-1)$维的扁平映射子空间。
（1）在一个二维空间，一个超平面就是一个一维扁平子空间，也就是一条线。
（2）在一个三维空间，一个超平面就是一个两维扁平子空间，也就是一个平面。
（3）在一个$p>3$的多维空间，很难直接展示超平面，但其定义不变。

对于一个二维空间，其超平面可以用以下数学公式定义：

$$\beta_0+\beta_1X_1+\beta_2X_2=0 \quad (9.1)$$

也就是说，任何满足公式9.1的数据$X=(X_1,X_2)^T$都是超平面上一点。

在一个p维空间，公式9.1可以扩展为：

$$\beta_0+\beta_1X_1+\beta_2X_2+\cdots+\beta_pX_p=0 \quad (9.2)$$

同样，任何满足公式9.2的数据$X=\left(X_1,X_2,\ldots,X_p\right)^T$都是超平面上一点。

1. 最大边际分类器(Maximal Margin Classifier)

假设某数据X不满足公式9.2，则有两种情况：

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0 \quad (9.3)$$

数据位于超平面外一边。

或者

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0 \quad (9.4)$$

数据位于超平面另外一边。

对于一个二维空间，其超平面为如右图所示。

也就是说，我们可以认为该超平面将p维空间分割为了两部分。因此，通过计算$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$的正负号，即可判断数据点位于超平面的哪一边。

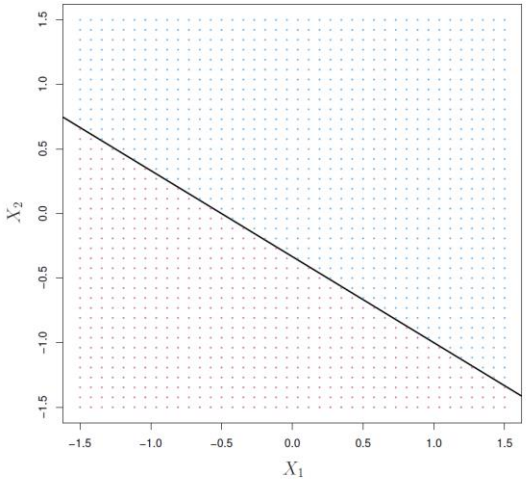


FIGURE 9.1. *The hyperplane $1 + 2X_1 + 3X_2 = 0$ is shown. The blue region is the set of points for which $1 + 2X_1 + 3X_2 > 0$, and the purple region is the set of points for which $1 + 2X_1 + 3X_2 < 0$.*

1. 最大边际分类器(Maximal Margin Classifier)

假设有一个$n*p$的训练数据矩阵X，即包含n个p维特征值的训练数据：$X=\begin{pmatrix} x_{11} & \cdots & x_{n1} \\ \vdots & \vdots & \vdots \\ x_{1p} & \cdots & x_{np} \end{pmatrix}$。

这些数据被划分为两类：$y_1,\ldots,y_n \in \{-1,1\}$，其中$-1$代表一类，而1代表另一类。

假设有一个检验数据集$x^*=(x_1^*,\ldots,x_p^*)^T$。如何基于训练数据开发一个分类模型，从而利用特征测量值正确预测检验数据？我们可以利用一个分割超平面(Separating Hyperplane)来对数据进行分类。

如右图所示，假设蓝色点$y_i=1$，紫色点$y_i=-1$。则该分割超平面应该具有以下特性：

$$\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_p x_{ip}>0 \text{ if } y_i=1 \quad (9.6)$$

$$\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_p x_{ip}<0 \text{ if } y_i=-1 \quad (9.7)$$

或者也可以写成

$$y_i(\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_p x_{ip})>0$$

for all $i=1,\ldots,n$。

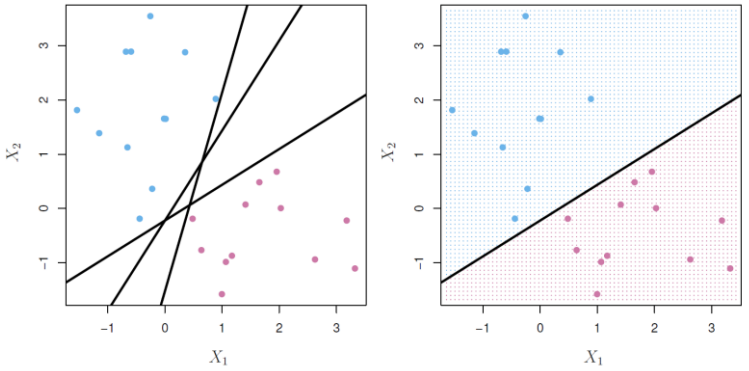


FIGURE 9.2. Left: *There are two classes of observations, shown in blue and in purple, each of which has measurements on two variables. Three separating hyperplanes, out of many possible, are shown in black.* Right: *A separating hyperplane is shown in black. The blue and purple grid indicates the decision rule made by a classifier based on this separating hyperplane: a test observation that falls in the blue portion of the grid will be assigned to the blue class, and a test observation that falls into the purple portion of the grid will be assigned to the purple class.*

对于x^*，

（1）可以通过判断$f(x^*)=\beta_0+\beta_1 x_1^*+\beta_2 x_2^*+\cdots+\beta_p x_p^*$的符号，来对其进行分类。

（2）$f(x^*)$大小可以体现可靠性。如果$f(x^*)$离0较远，则x^*离超平面较远，我们对分类结果就比较有信心；如果$f(x^*)$接近0，则x^*靠近超平面，分类结果的不确定性就更大。实际上，这个分割超平面就是一个线性边界。

1. 最大边际分类器(Maximal Margin Classifier)

如右图所示，对于一个数据集，通过上下移动，我们可能有无数个可能的分割超平面，到底选择哪一个？

我们选用的最大边际超平面(Maximal Margin Hyperplane, or Optimal separating hyperplane)，也就是离训练数据集数据最远的那个超平面。

也就是说，对于每一个给定的分割超平面，我们都可以计算每个训练数据与其之间的垂直距离，其中所有距离值中最小的距离被称为边际(Margin)。最大边际超平面就是边际最大的那个分割超平面，也就是与所有训练数据最小距离中最大的那个分割超平面。
接着，对于检验数据，我们就可以根据它们落在最大边际超平面哪边，判断它们属于哪一类。这就是所谓的最大边际分类器。

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

对于右图与图9.2的右图，可以发现，这个分割超平面的最小距离明显更大。

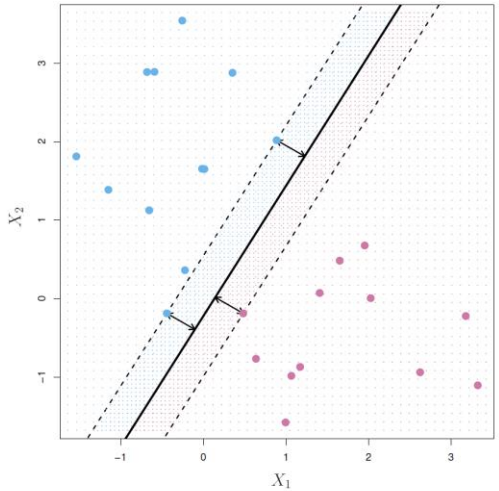


FIGURE 9.3. *There are two classes of observations, shown in blue and in purple. The maximal margin hyperplane is shown as a solid line. The margin is the distance from the solid line to either of the dashed lines. The two blue points and the purple point that lie on the dashed lines are the support vectors, and the distance from those points to the hyperplane is indicated by arrows. The purple and blue grid indicates the decision rule made by a classifier based on this separating hyperplane.*

上图有三个点距离最大边际超平面的距离一样，这三个点就被称为支持向量(Support Vectors)，因为它们是p维(p=2)空间向量，且支持最大边际超平面。也就是说，如果这些点有些许移动，那么最大边际超平面也需要进行移动。

1. 最大边际分类器(Maximal Margin Classifier)

如何构筑最大边际超平面?
简要来说，最大边际超平面就是一个优化问题的解:

$$\max_{\beta_0,\beta_1,\ldots,\beta_p,M} \quad M \qquad (9.9)$$

$Subject\ to$

$$\sum_{j=1}^{p} {\beta_j}^2 = 1 \qquad (9.10)$$

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M \ \ \forall \ i = 1, \ldots, n \qquad (9.11)$$

其中，$M > 0$。
可以证明约束条件(9.10)使得第i个观测值到超平面的垂直距离是$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right)$。
The constraints (9.10) and (9.11) ensure that each observation is on the correct side of the hyperplane and at least a distance M from the hyperplane.
因此，M就是超平面的边际值(margin)，上述优化公式的目标就是选择能够最大化M的各参数值。

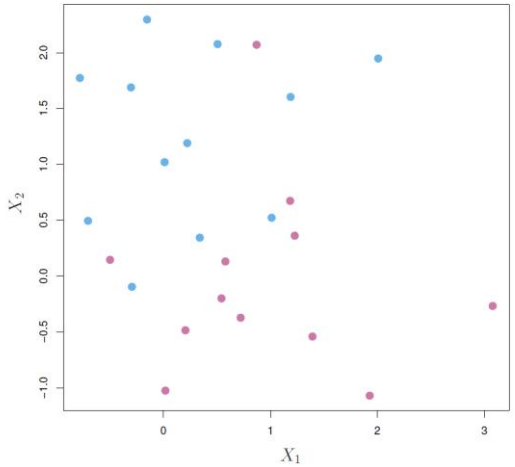


FIGURE 9.4. *There are two classes of observations, shown in blue and in purple. In this case, the two classes are not separable by a hyperplane, and so the maximal margin classifier cannot be used.*

如果分割超平面存在，可以利用最大边际分类器完成分类。但很多情况下，上述优化问题是无解的，也就是不存在分割超平面，也就是不可分开的例子(Non-separable case)。在这种情况下，我们可以使用所谓的软边际(soft margin)来对数据进行分类。由最大边际分类器延伸至不可分割的例子就称为支持向量分类器(support vector classifier)。

2. 支持向量分类器(Support Vector Classifier)

很多时候，最大边际分类器的分类表现并不能让人满意，如右边第二个图所示，因为边际值很小。最大边际超平面对单个观测值非常敏感，导致它很容易出现过拟合的情况。

支持向量分类器，又称为软边际分类器，可以解决这一问题。支持向量分类器并不是寻找最大边际，而是需按照软边际，也就是说它允许部分训练数据被错误的分类。支持向量分类器对应的优化问题是：

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M \qquad (9.12)$$

Subject to

$$\sum_{j=1}^{p} \beta_j{}^2 = 1 \qquad (9.13)$$

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M(1 - \epsilon_i) \quad (9.15)$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i{}^2 \leq C \qquad (9.15)$$

式中，C是一个非负调节参数；

M是边际宽度；

$\epsilon_1, \ldots, \epsilon_n$是松弛变量。

同样，对于检验数据x^*，我们可以基于$f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*$的符号对数据进行分类。

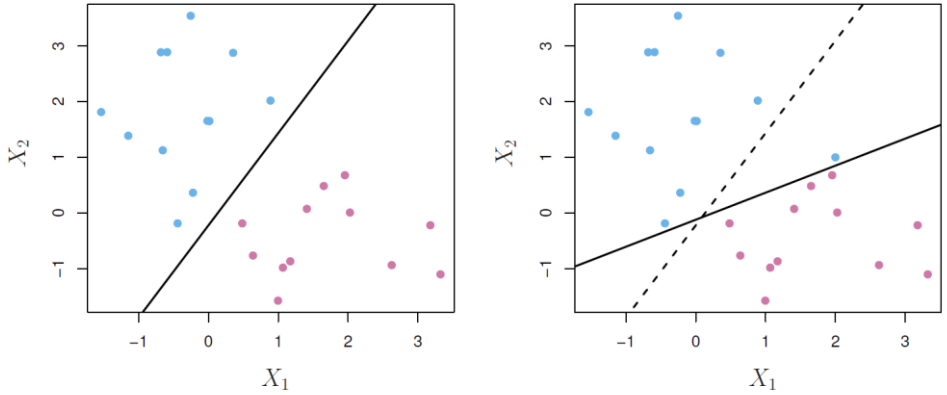


FIGURE 9.5. Left: *Two classes of observations are shown in blue and in purple, along with the maximal margin hyperplane.* Right: *An additional blue observation has been added, leading to a dramatic shift in the maximal margin hyperplane shown as a solid line. The dashed line indicates the maximal margin hyperplane that was obtained in the absence of this additional point.*

2. 支持向量分类器(Support Vector Classifier)

支持向量分类器对应的优化问题是：

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \; M \quad (9.12)$$

$$Subject\ to$$

$$\sum_{j=1}^{p} {\beta_j}^2 = 1 \quad (9.13)$$

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M(1-\epsilon_i) \quad (9.15)$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} {\epsilon_i}^2 \leq C \quad (9.15)$$

式中，C是一个非负调节参数；M是边际宽度；$\epsilon_1, \ldots, \epsilon_n$是松弛变量。

可以发现，只有落在边际上或者违反边际的数据才会影响超平面，它们被称为支持向量，直接影响支持模型构建。这使得模型对于异常数据非常的鲁棒性。

C决定了我们能够容忍的违反边际或超平面情况的数目与严重程度。

- 当$C = 0$时，公式就变成了最大边界分类器模型；
- 当$C > 0$时，有不超过C个数据会被划分错误；
- 随着C的增大，模型对于违背边际的情况容忍度更大，也就是边际更宽；
- 随着C的减小，模型对于违背边际的情况容忍度变小，也就是边际更窄。

在实际应用中，C值可以利用交叉验证来确定。

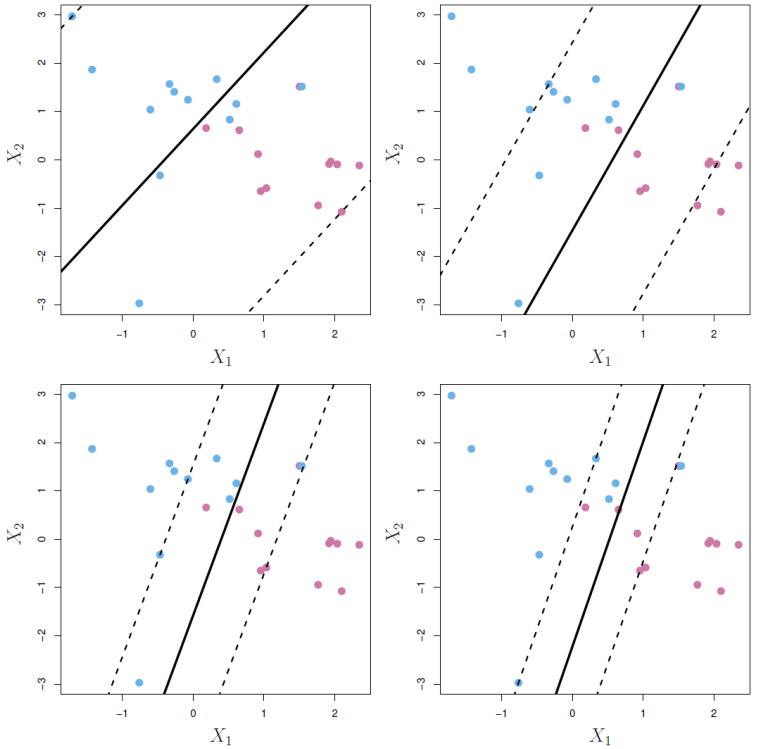


FIGURE 9.7. *A support vector classifier was fit using four different values of the tuning parameter C in (9.12)–(9.15). The largest value of C was used in the top left panel, and smaller values were used in the top right, bottom left, and bottom right panels. When C is large, then there is a high tolerance for observations being on the wrong side of the margin, and so the margin will be large. As C decreases, the tolerance for observations being on the wrong side of the margin decreases, and the margin narrows.*

2. 支持向量分类器(Support Vector Classifier)

SVM例子1：Carseats数据集

```
library(e1071)
library(ISLR2)

##### 1. Classification: Carseats#####
### 1.1 创造新变量：必须转换成factor变量
Carseats$High<- ifelse(Carseats$Sales<=8,"No","Yes")
Carseats$High<- factor(Carseats$High)
# 分割训练和检验数据集
set.seed(2)
train<- sample(1:nrow(Carseats),nrow(Carseats)/2)
train_Carseats<- Carseats[train,]
test_Carseats<- Carseats[-train,]

### 1.2 建立SVM模型：Linear
svm1_Carseats<- svm(High~.-Sales,
                    data = train_Carseats,
                    kernel = "linear",
                    cost = 1)
print(summary(svm1_Carseats))

# 预测检验数据集
test_Carseats$pred_svm1<- predict(svm1_Carseats,
                                  newdata = test_Carseats,
                                  type = "class")
# 计算混淆矩阵
cm5<- table(test_Carseats$pred_svm1,test_Carseats$High)
print(cm5)
# 计算准确率
accuracy5<- sum(diag(cm5))/sum(cm5)
print(accuracy5)
```

```
> print(cm5)

      No Yes
  No  110  10
  Yes   7  73
> # 计算准确率
> accuracy5<- sum(diag(cm5))/sum(cm5)
> print(accuracy5)
[1] 0.915
```

Model	CART	Bagging	RF	Boosting	SVM-Linear
Accuracy	0.77	0.835	0.83	0.865	0.915

3. 支持向量机(Support Vector Machines)

对于二分类问题，如果两类数据之间存在线性边界的话，那么使用支持向量分类器即可。然而实际应用中，不同类别数据之间经常存在非线性的边界。怎么办?

对于各特征变量，我们可以利用他们的多项式来识别非线性边界。比如，我们可以利用$2p$个预测变量：
$X_1, X_1^2, X_2, X_1^2, \dots, X_p, X_p^2.$
支持向量机对应的优化问题是：

$$\underset{\beta_0,\beta_1,\dots,\beta_p,\epsilon_1,\dots,\epsilon_n,M}{\text{maximize}} \quad M \qquad (9.16)$$

$s.t.$

$$\begin{cases} y_i\left(\beta_0 + \sum_{j=1}^{p}\beta_{j1}x_{ij} + \sum_{j=1}^{p}\beta_{j2}{x_{ij}}^2\right) \geq M(1-\epsilon_i) \\ \epsilon_i \geq 0 \\ \sum_{i=1}^{n}\epsilon_i \leq C \\ \sum_{j=1}^{p}\sum_{k=1}^{2}{\beta_{jk}}^2 = 1 \end{cases} \qquad (9.15)$$

式中，C是一个非负调节参数；M是边际宽度；$\epsilon_1, \dots, \epsilon_n$是松弛变量。

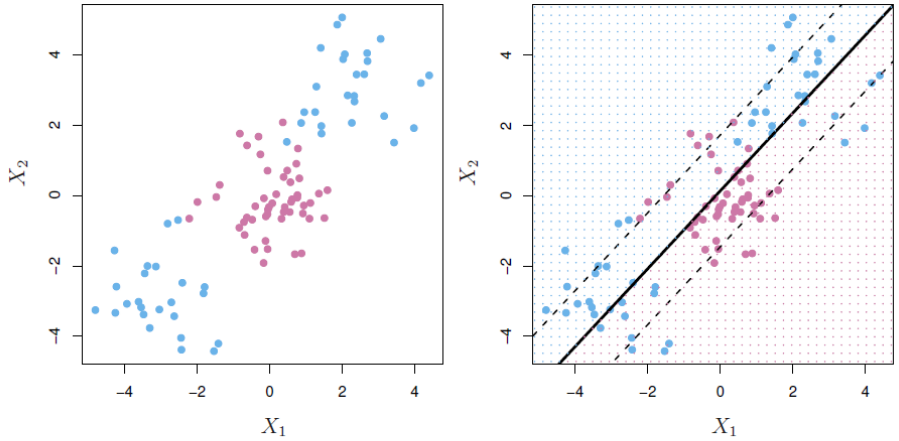


FIGURE 9.8. Left: *The observations fall into two classes, with a non-linear boundary between them.* Right: *The support vector classifier seeks a linear boundary, and consequently performs very poorly.*

可以发现，能够实现non-linear的特征空间很多，比如更高次多项式，交互效应，其他公式等。显然，特征空间越复杂，求解也会越困难。如何选择一个计算高效的的支持向量机分类器?

3. 支持向量机(Support Vector Machines)

支持向量机是支持向量分类器的一个特殊形式，其采用核(kernel)来扩大特征空间，以识别不同类别之间的非线性边界。

支持向量分类器公式(9.12)-(9.15)只涉及到观测数据之间的内积(Inner product)。
对于两个r维向量a与b，其内积为$\langle a,b\rangle=\sum_{i=1}^{r}a_ib_i$。因此，两个观测数据$x_i$，$x_{i'}$的内积为

$$\langle x_i,x_{i'}\rangle=\sum_{j=1}^{p}x_{ij}x_{i'j} \quad (9.17)$$

可以发现，线性支持向量分类器可以写为

$$f(x)=\beta_0+\sum_{i=1}^{n}a_i\langle x,x_i\rangle \quad (9.18)$$

式中有n个参数a_i，$i=1,\dots,n$，n为训练数据个数。

为了估计参数a_1，…，a_n，以及β_0，我们需要所有训练数据两两之间的内积$\langle x_i,x_{i'}\rangle$，也就是$\binom{n}{2}=\frac{n(n-1)}{2}$个内积。注意在公式(9.18)中，为了估计公式$f(x)$，我们需要计算新数据点$x$与每个训练数据点$x_i$之间的内积。

可以发现，当某个训练数据不是支持向量时，$a_i=0$；当其是支持向量时，$a_i\neq 0$。因此，如果S是所有支持点的索引集合，我们可以把公式(9.18)的解写为

$$f(x)=\beta_0+\sum_{i\in S}a_i\langle x,x_i\rangle \quad (9.19)$$

也就是为了计算线性分类器$f(x)$的回归系数，我们只需要计算内积即可。

3. 支持向量机(Support Vector Machines)

那么，我们现在用一个通用的内积表示方式

$$K(x_i, x_{i'}) \quad (9.20)$$

式中，K是称为核(kernel)的某种公式。一个核就是量化两个观测数据相似性的公式。

比如，我们可以简单地用

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j} \quad (9.21)$$

公式(9.21)也被称为线性核，因为支持向量分类器的特征是现行的。线性核使用的Pearson关联系数来量化两组观测值的相似性。

我们也可以使用其他形式，比如

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d \quad (9.22)$$

公式(9.22)也被称为自由度为d的多项式核，其中d为正整数。

当支持向量分类器与类似公式(9.22)这种非线性核结合在一起时，这种分类器就被称为支持向量机。在这种情况下，非线性函数的形式为

$$f(x) = \beta_0 + \sum_{i \in S} a_i K(x_i, x_{i'})$$

一个区分非线性数据的多项式核的支持向量机

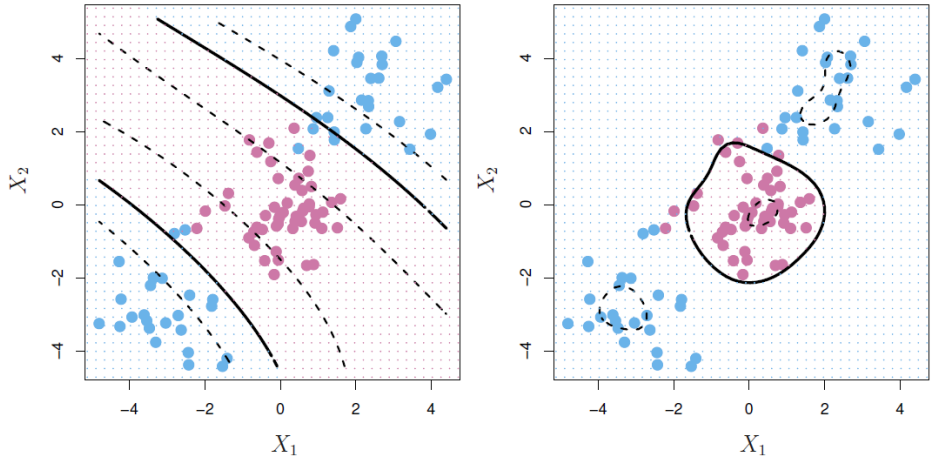


FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

3. 支持向量机(Support Vector Machines)

另外一种常用的核函数为径向核(radial kernel)，

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \quad (9.24)$$

式中，γ是一个正常数。

径向核函数的工作原理：如果一个新的检验数据$x^* = (x_1^*, \dots, x_p^*)^T$离一个训练数据$x_i$的欧氏距离(Euclidean distance)较远，那么$\sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$就偏大，$\exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$就偏小。这就意味着，$x_i$对$f(x^*)$值的影响很小。也就是说，所有离检验数据$x^*$比较远的训练数据在预测$x^*$的类别方面，几乎不会起到什么作用。因此，径向核有很好的本地行为，只有检验数据周围的训练数据才会对检验数据的预测结果有影响。

$$f(x) = \beta_0 + \sum_{i \in S} a_i K(x_i, x_{i'})$$

使用核比简单扩大特征空间的一个优势就是计算。对于核函数来说，我们只需要计算任意一对数据i和i'的内积$K(x_i, x_{i'})$，一共$\binom{n}{2}$次。在很多SVM应用中，特征空间太大会导致无法完成计算活动。

一个区分非线性数据的多项式核的支持向量机

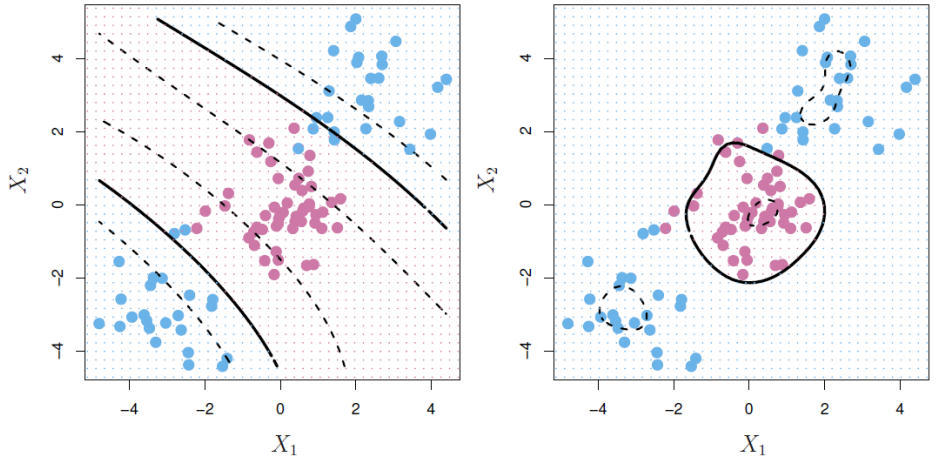


FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

3. 支持向量机(Support Vector Machines)

SVM例子1：Carseats数据集 -Radial

```
### 1.3 建立SVM模型：Radial
svm2_Carseats<- svm(High~.-Sales,
                    data = train_Carseats,
                    kernel = "radial",
                    gamma = 0.001,
                    cost = 1)
print(summary(svm2_Carseats))

# 预测检验数据集
test_Carseats$pred_svm2<- predict(svm2_Carseats,
                                  newdata = test_Carseats,
                                  type = "class")
# 计算混淆矩阵
cm6<- table(test_Carseats$pred_svm2,test_Carseats$High)
print(cm6)
# 计算准确率
accuracy6<- sum(diag(cm6))/sum(cm6)
print(accuracy6)
```

```
> print(cm6)

      No Yes
  No  117  83
  Yes   0   0
> # 计算准确率
> accuracy6<- sum(diag(cm6))/sum(cm6)
> print(accuracy6)
[1] 0.585
```

```
### 1.4 建立SVM模型：Radial，利用cross-validation选择最优参数
set.seed(1)
tune.out<- tune(svm,
                High~.-Sales,
                data = train_Carseats,
                kernel = "radial",
                ranges = list(
                  cost = c(0.1,1,10,100),
                  gamma = c(0.0001,0.001,0.01,0.1,1)
                ))
print(summary(tune.out))
```

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross v

- best parameters:
 cost gamma
  100 0.001

- best performance: 0.155

- Detailed performance results:
    cost gamma error dispersion
1    0.1 1e-04 0.405 0.11890706
2    1.0 1e-04 0.405 0.11890706
3   10.0 1e-04 0.405 0.11890706
4  100.0 1e-04 0.225 0.14953632
5    0.1 1e-03 0.405 0.11890706
6    1.0 1e-03 0.405 0.11890706
7   10.0 1e-03 0.225 0.14953632
8  100.0 1e-03 0.155 0.10658851
9    0.1 1e-02 0.405 0.11890706
10   1.0 1e-02 0.230 0.14944341
11  10.0 1e-02 0.160 0.09067647
12 100.0 1e-02 0.190 0.10749677
13   0.1 1e-01 0.405 0.11890706
14   1.0 1e-01 0.220 0.13984118
15  10.0 1e-01 0.230 0.11352924
16 100.0 1e-01 0.245 0.11654756
17   0.1 1e+00 0.405 0.11890706
18   1.0 1e+00 0.415 0.12703893
19  10.0 1e+00 0.395 0.12349089
20 100.0 1e+00 0.395 0.12349089
```

```
### 1.5 建立SVM模型：Radial，调整参数
svm2_Carseats<- svm(High~.-Sales,
                    data = train_Carseats,
                    kernel = "radial",
                    gamma = 0.001,
                    cost = 100)
print(summary(svm2_Carseats))

# 预测检验数据集
test_Carseats$pred_svm2<- predict(svm2_Carseats,
                                  newdata = test_Carseats,
                                  type = "class")
# 计算混淆矩阵
cm6<- table(test_Carseats$pred_svm2,test_Carseats$High)
print(cm6)
# 计算准确率
accuracy6<- sum(diag(cm6))/sum(cm6)
print(accuracy6)
```

```
> print(cm6)

      No Yes
  No  112  14
  Yes   5  69
> # 计算准确率
> accuracy6<- sum(diag(cm6))/sum(cm6)
> print(accuracy6)
[1] 0.905
```

3. 支持向量机(Support Vector Machines)

我们之前的案例中，目标变量均为二分类数据。针对多分类数据($K > 2$)，如何处理?

- One-versus-One classification：首先，针对任意两类数据，构建1个SVM模型，总共构建$\binom{K}{2}$个SVM模型。其次，对于任意一个检验数据，用每一个SVM模型进行预测，确定其分类，总共得出$\binom{K}{2}$个预测结果。接着，计算这个检验数据属于K个类别中每个分类的次数。最后，将这个检验数据归为次数出现最多的那个分类。

- One-versus-All classification：我们一共建立K个SVM模型，每个模型对比任一类别数据与其他$K-1$类的数据。 建设对于第k类数据，建立的SVM模型估计出来的参数为$\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$。针对一个检验数据$x^*$，将其归为$\beta_{0k} + \beta_{1k}x_1^* + \beta_{2k}x_2^* + \cdots + \beta_{pk}x_p^*$值最大的那类，因为这个数据能够代表我们对于数据属于第k类数据的信心。

3. 支持向量机(Support Vector Machines)

我们之前的案例中，目标变量均为二分类数据。针对多分类数据($K > 2$)，如何处理?

- One-versus-One classification：首先，针对任意两类数据，构建1个SVM模型，总共构建$\binom{K}{2}$个SVM模型。其次，对于任意一个检验数据，用每一个SVM模型进行预测，确定其分类，总共得出$\binom{K}{2}$个预测结果。接着，计算这个检验数据属于K个类别中每个分类的次数。最后，将这个检验数据归为次数出现最多的那个分类。

- One-versus-All classification：我们一共建立K个SVM模型，每个模型对比任一类别数据与其他$K-1$类的数据。 建设对于第k类数据，建立的SVM模型估计出来的参数为$\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$。针对一个检验数据$x^*$，将其归为$\beta_{0k} + \beta_{1k}x_1^* + \beta_{2k}x_2^* + \cdots + \beta_{pk}x_p^*$值最大的那类，因为这个数据能够代表我们对于数据属于第k类数据的信心。

4. ROC曲线

混淆矩阵 (Confusion matrix)		真实值	
		Positive (P)	Negative (N)
预测值	Positive (P)	TP	FP
	Negative (N)	FN	TN
	Total	P	N

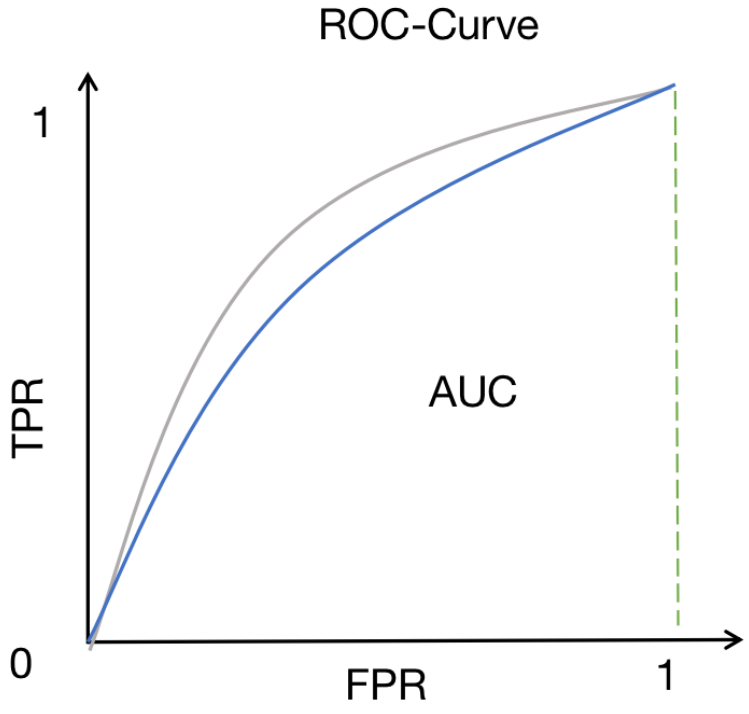
指标	公式	含义
准确率 (Accuracy)	$A = \frac{TP + TN}{P + N}$	正确预测的结果占所有观测值的比例
灵敏度 (Recall, Sensitivity)	$R = \frac{TP}{TP + FN}$	所有真实值是Positive的数据中，模型预测正确的比例
精确率 (Precision)	$P = \frac{TP}{TP + FP}$	所有预测值是Positive的数据中，模型预测正确的比例
特异度 (Specificity)	$S = \frac{TN}{TN + FP}$	所有真实值是Negative的数据中，模型预测正确的比例
F1 Score	$F1 = \frac{2PR}{P + R}$	取值范围为0到1，1代表模型输出最好，0代表最差。

4. ROC曲线

受试者工作特征曲线(Receiver operating characteristic curve, ROC curve)

A ROC curve is a graph that shows a classification model performance at all classification thresholds. It is a probability curve that plots two parameters, the True Positive Rate (TPR) against the False Positive Rate (FPR), at different threshold values and separates a so-called 'signal' from the 'noise.'

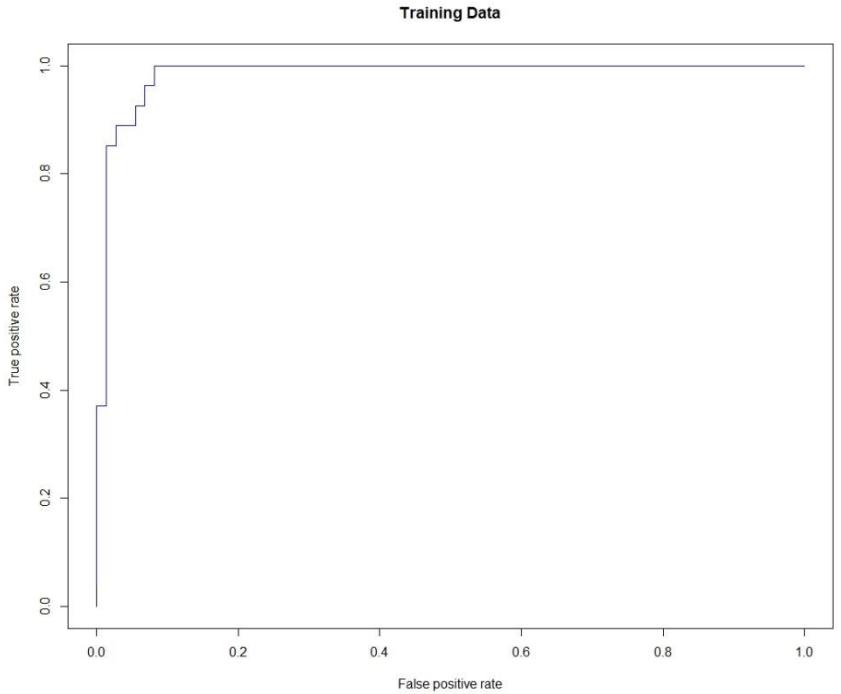
指标	公式	含义
灵敏度(Recall, Sensitivity)	$R = \frac{TP}{TP + FN}$	所有真实值是Positive的数据中，模型预测正确的比例
特异度(Specificity)	$S = \frac{TN}{TN + FP}$	所有真实值是Negative的数据中，模型预测正确的比例



4. ROC曲线

```
library(ROCR)

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))
train<- sample(200,100)

rocplot<- function(pred,truth,...){
  predob<- prediction(pred,truth)
  perf<- performance(predob,"tpr","fpr")
  plot(perf,...)
}
```

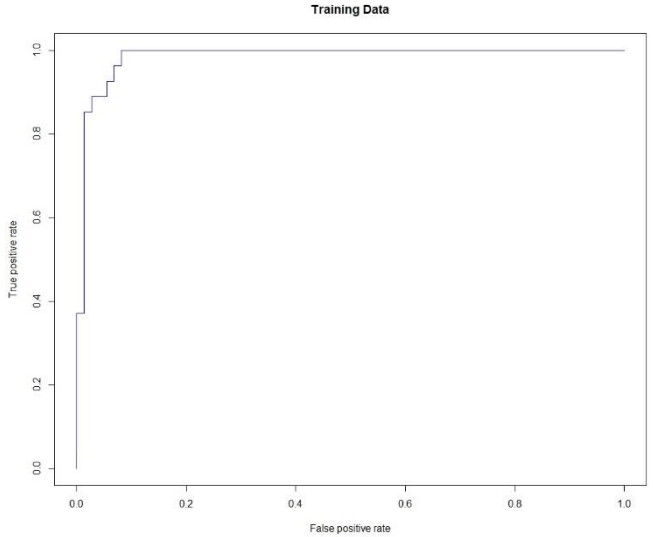
```
### 1. SVM1 - Training Data
svmfit.opt<- svm(y~.,
                 data = dat[train,],
                 kernel = "radial",
                 gamma = 2,
                 cost = 1,
                 decision.values = T)

fitted<- attributes(predict(svmfit.opt,
                            dat[train,],
                            decision.values = TRUE))$decision.values

rocplot(-fitted,dat[train,"y"],main ="Training Data")
```



4. ROC曲线

```
### 手动计算ROC Curve
roc1<- data.frame(attributes(predict(svmfit.opt,
                          dat[train,],
                          decision.values = TRUE)))
roc1$truth<- dat[train,"y"]
roc1<- roc1[order(roc1$X1.2),]
roc1$T1<- 0
roc1$T2<- 0
for(i in 1:nrow(roc1)){
  roc1[i,"T1"]<- sum(roc1[1:i,"truth"]=="1")
  roc1[i,"T2"]<- sum(roc1[1:i,"truth"]=="2")
}
roc1$TPR<- roc1$T1/sum(roc1[,"truth"]=="1")
roc1$FPR<- roc1$T2/sum(roc1[,"truth"]=="2")
lines(roc1$TPR,roc1$FPR,col="blue")
```

```
> head(roc1)
    levels  class names      X1.2 truth T1 T2 TPR         FPR
188      1 factor   188 -1.491965     2  0  1   0 0.03703704
181      1 factor   181 -1.430744     2  0  2   0 0.07407407
200      2 factor   200 -1.422113     2  0  3   0 0.11111111
198      1 factor   198 -1.290352     2  0  4   0 0.14814815
158      1 factor   158 -1.275544     2  0  5   0 0.18518519
174      2 factor   174 -1.254779     2  0  6   0 0.22222222
```



当x1.2等于-1.491965时，
TP=0，TPR=0/73=0
FP=2，FPR=0/27=0

谢 谢!