还可以对 SVM 运用 tune() 函数，即使用交叉验证来选择径向核函数最优的 γ 值以及 cost 的值：

```
> set.seed(1)
> tune.out=tune(svm, y~., data=dat[train,], kernel="radial",
    ranges=list(cost=c(0.1,1,10,100,1000),
    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     2
- best performance: 0.12
- Detailed performance results:
    cost gamma error dispersion
1  1e-01   0.5  0.27     0.1160
2  1e+00   0.5  0.13     0.0823
3  1e+01   0.5  0.15     0.0707
4  1e+02   0.5  0.17     0.0823
5  1e+03   0.5  0.21     0.0994
6  1e-01   1.0  0.25     0.1354
7  1e+00   1.0  0.13     0.0823
. . .
```

因此，最优的选择为 cost =1 和 gamma =2。使用 prediction() 函数可以查看这个模型在测试集上的预测结果。注意到，在这个过程中，以 -train 为索引集把数据框 dat 分成子集。

```
> table(true=dat[-train,"y"], pred=predict(tune.out$best.model,
    newx=dat[-train,]))
```

这个 SVM 误分了 39% 的测试观测。

9.6.3 ROC 曲线

ROCR 包可以用来生成图 9-10 和图 9-11 中的 ROC 曲线。为了画出 ROC 曲线，首先需生成一个函数，这个函数能够在给定的包含每个观测数值得分的 pred 向量和包含每个观测真实类别的 truth 向量前提下，画出 ROC 曲线。

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

SVM 和支持向量分类器会输出每个观测的预测的类别标签。但是，获得每个观测的**拟合值**（fitted value）也是可能的，而拟合值正是用来获得类别标签的数值得分。例如，在支持向量分类器中，观测 $X=(X_1,X_2,\cdots,X_p)^T$ 的拟合值的形式为 $\hat{\beta}_0+\hat{\beta}_1X_1+\hat{\beta}_2X_2+\cdots+\hat{\beta}_pX_p$。对于非线性核函数的 SVM，计算拟合值的式子为式（9.23）。从本质上来说，拟合值的符号决定了观测落在决策边界的哪一侧。因此，对于一个给定的观测来说，拟合值和预测类别的关系是非常简单的：如果拟合值大于 0，那么观测就被分成第一类，如果拟合值小于 0，则观测被分为另一类。为了得到一个给定的 SVM 模型的拟合值，在拟合 svm() 的时候使用 decision.value =TRUE。然后 predict() 函数会输出拟合值。

```
> 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
```

接下来就可以画 ROC 曲线了。

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

SVM 似乎能给出精确的预测。但是增加 γ 可以更加光滑地拟合数据，并且进一步提高预测的准确度。

```
> 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")
```

然而这些都是训练数据的 ROC 曲线。通常我们更关注测试数据的预测的精度。根据测试数据的 ROC 曲线来评价，$\gamma=2$ 的模型似乎能够提供更准确的预测结果。

```
> 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")
```

9.6.4 多分类的 SVM

如果响应因子的水平数超过 2，那么 svm() 函数就会使用“一类对一类”的方法进行分类。首先生成第三类观测来了解这种情况。

```
> 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))
```

现在用 SVM 来拟合这个数据：

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

如果输入函数 svm() 的响应变量是数值变量而不是因子变量的话，e1071 库也可以用来做支持向量回归。

9.6.5 基因表达数据的应用

以下分析都使用 Khan 数据集，这个数据集由许多组织样本构成，这些样本对应四种不同的蓝色小圆细胞肿瘤。对于每个组织样本，基因表达测定都是可用的。数据集由训练数据 xtrain 和 ytrain，以及测试数据 xtest 和 ytest 组成。

首先看数据的维数：

```
> library(ISLR)
> names(Khan)
[1]    "xtrain"  "xtest"  "ytrain"  "ytest"
> dim(Khan$xtrain)
[1]    63 2308
> dim(Khan$xtest)
[1]    20 2308
> length(Khan$ytrain)
[1] 63
> length(Khan$ytest)
[1] 20
```

数据集由 2 308 个基因的表达测定组成。训练集和测试集分别由 63 和 20 个观测组成。

```
> table(Khan$ytrain)
 1  2  3  4
 8 23 12 20
> table(Khan$ytest)
1 2 3 4
3 6 6 5
```

对基因表达测定数据，使用支持向量机方法来预测癌症亚型。在这个数据集中，相对于观测的数目来说，特征的数目非常多。基于这一特点建议使用线性核函数，因为使用多项式核函数和径向核函数得到更高光滑性是没有必要的。

```
> dat=data.frame(x=Khan$xtrain, y=as.factor(Khan$ytrain))
> out=svm(y~., data=dat, kernel="linear",cost=10)
> summary(out)
Call:
svm(formula = y ~ ., data = dat, kernel = "linear",
    cost = 10)
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10
      gamma:  0.000433
Number of Support Vectors:  58
 ( 20 20 11 7 )
Number of Classes:  4
Levels:
 1 2 3 4
> table(out$fitted, dat$y)

    1  2  3  4
  1 8  0  0  0
  2 0 23  0  0
  3 0  0 12  0
  4 0  0  0 20
```

结果显示，训练集的误差为0。事实上，这并不奇怪，因为相对于观测的数目来说，变量的数目较多意味着很容易找到把这些类别完全分开的超平面。我们最关注的并不是支持向量分类器在训练观测上的分类效果，而是在测试观测上的分类效果。

```
> dat.te=data.frame(x=Khan$xtest, y=as.factor(Khan$ytest))
> pred.te=predict(out, newdata=dat.te)
> table(pred.te, dat.te$y)

pred.te 1 2 3 4
       1 3 0 0 0
       2 0 6 2 0
       3 0 0 4 0
       4 0 0 0 5
```