使用SVR构建预测模型

Alex / 2023-09-14 / free_learner@163.com / AlexBrain.cn

本文介绍在R中使用支持向量回归方法（Support Vector Regression，SVR）构建预测模型的基本步骤。

一、背景

SVR是脑成像领域最常用的机器学习模型之一，在文献中经常可以看到。使用场景通常是根据脑成像指标去预测行为表现或者临床症状，其中行为表现或临床症状是一个连续的变量。由于SVR背后的数学原理对我来说比较复杂（如果你想进一步了解，我自己看过一个小册子感觉还不错），这里只介绍如何使用R的`e1071`包构建SVR预测模型。为了便于理解，我这里分为三个部分来介绍：模型训练和测试、交叉验证和参数优化。我自己是机器学习小白，建议多参考网上其他教程。

二、样例数据

这里我使用的是`mlbench`包里的`BostonHousing`数据集，总共包括14个变量。我们需要根据前13个变量（房屋的各种细节）来预测最后一个变量（房屋价格）。`chas`变量是一个分类变量（因子），所以进行了手动编码（如果在调用`svm`函数时使用公式形式，那会自动处理分类变量的编码问题）。

```
library(mlbench)
data(BostonHousing)
y <- BostonHousing[,14]
x <- BostonHousing[,c(1:13)]
x$chas <- ifelse(x$chas == 0, 0, 1)
```

三、模型训练和测试

首先将数据集划分出训练集（80%的数据）和测试集（20%的数据）两部分。训练集用来训练SVR模型，测试集用来测试模型预测表现。

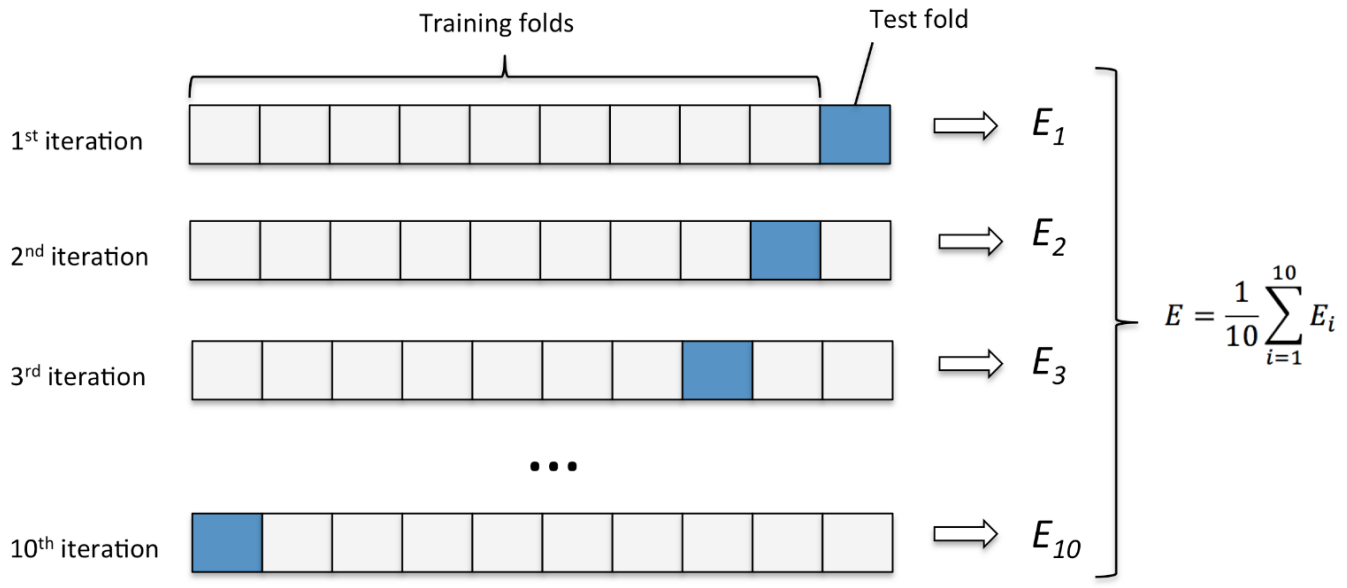
```
## Split the whole dataset into training and test sets
N <- nrow(x)
set.seed(100)
train_idx <- sample(c(1:N), round(0.8*N))
x_train <- x[train_idx,]
y_train <- y[train_idx]
x_test <- x[-train_idx,]
y_test <- y[-train_idx]
```

`svm`函数既可以用于分类任务，也可以用于回归任务（SVR）。`svm`函数会根据反应变量（response variable）的数据类型来判断，如果反应变量是连续变量，那么就是SVR。有很多种评估模型预测表现的指标，我这里采用的是相关系数。我这里测试的结果是，在训练集上相关系数是0.95，在测试集上是0.89。

```
library(e1071)
## Train SVR model
mymod <- svm(x_train, y_train)
myacc_train <- cor(mymod$fitted, y_train)
## Test SVR model
mypred <- predict(mymod, x_test)
myacc_test <- cor(mypred, y_test)
```

四、交叉验证

在上一步，我们将数据划分成了训练集和测试集。在实践中，为了充分利用数据从而更好地评估模型表现，经常采用交叉验证（cross validation）的方法。如下图所示（来源于网络），K折交叉验证的思路就是将数据集分成K份，其中一份作为测试集，其余K-1份作为训练集。重复K次，使得每一份都做一次测试集。



`svm`函数也包含交叉验证的选项，这里我自己实现了一下，因为过程比较简单。在交叉验证的情况下，一般会在每折里评估表现，然后对所有K折的表现进行平均。由于我这里采用的是相关系数来评估模型表现，所以我将K折的结果合在一起计算相关。5折交叉验证的相关系数是0.91。由于在划分训练集和测试集的过程中存在随机性，也可以重复交叉验证多次，取平均表现作为一个稳定的估计。

```
## 5-fold cross validation
set.seed(100)
mypred <- numeric(length = N)
rand_idx <- sample(N)
K <- 5
folds <- cut(c(1:N), breaks=5, labels=FALSE)
## Loop through each fold
for (curr_fold in c(1:K)){
  fold_idx <- which(folds == curr_fold, arr.ind=TRUE)
  x_train <- x[-rand_idx[fold_idx],]
  y_train <- y[-rand_idx[fold_idx]]
  x_test <- x[rand_idx[fold_idx],,drop=FALSE]
  ## Train model
  mymod <- svm(x_train, y_train)
  ## Test model
  mypred[rand_idx[fold_idx]] <- predict(mymod, x_test)
}
myacc <- cor(mypred, y)
```

五、参数优化

一般机器学习模型都包含有超参数（hyper-parameter），超参数是无法从数据中估计的参数，需要人为进行选择。超参数的选择对模型表现往往有很大影响，因此为了取得最佳表现，一般需要进行参数优化。对于SVR，`cost`和`epsilon`是两个需要优化的超参数。`svm`函数默认的核函数是`radial`，因此`gamma`也是一个超参数。我看到一些资料说，`gamma`的影响没有`cost`和`epsilon`大，所以我一般只调试`cost`和`epsilon`这两个超参数，因为太多超参数，模型训练时间会很长。参数优化一般使用网格搜索（grid search）的策略，对所有的参数组合分别训练模型，选择表现最好的参数组合即为最佳参数。

`ranges`选项通过列表的方式输入需要调优的参数，`tunecontrol`选项这里我设置了5折交叉验证以及自定义的误差函数。默认情况下，误差函数是Mean Squared Error (MSE)，即最佳参数组合是MSE最小的参数。

```
set.seed(100)
## Define custom error function
myerr_fun <- function(v1, v2){
  MAE <- mean(abs(v1-v2))
  return(MAE)
}
## Using tune function
mytune <- tune(svm, x, y,
               ranges = list(epsilon = seq(0,1,0.1), cost = 10^(-5:5)),
               tunecontrol = tune.control(nrepeat = 1, sampling = "cross",
               cross = 5, error.fun = myerr_fun))
## Get best parameters & model
mytune$best.parameters
mytune$best.model
```