SVM Regression on Boston Housing Data

Karen Mazidi

Load the data

Read more about the data by typing "?Boston" at the console.

```
library(MASS)
df <- Boston[]
str(df)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

Train and test

Divide the data into 80% train and 20% test.

```
set.seed(1234)
i <- sample(nrow(Boston), 0.8*nrow(Boston), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Linear regression

Build a linear regression model on the training data.

```
lm1 <- lm(medv~., data=train)
summary(lm1)
```

```
##
## Call:
## lm(formula = medv ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.537  -2.913  -0.546   1.848  24.915
##
## Coefficients:
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  44.900577   6.016980   7.462 5.59e-13 ***
## crim         -0.085000   0.049892  -1.704  0.08924 .
## zn            0.047219   0.015849   2.979  0.00307 **
## indus         0.038249   0.070942   0.539  0.59008
## chas          2.724575   0.966685   2.818  0.00507 **
## nox         -19.139048   4.382515  -4.367 1.62e-05 ***
## rm            2.949428   0.479982   6.145 1.98e-09 ***
## age          -0.007757   0.015670  -0.495  0.62087
## dis          -1.558391   0.224867  -6.930 1.75e-11 ***
## rad           0.302988   0.076673   3.952 9.21e-05 ***
## tax          -0.012284   0.004206  -2.920  0.00370 **
## ptratio      -1.008491   0.152951  -6.594 1.40e-10 ***
## black         0.008717   0.003345   2.606  0.00951 **
## lstat        -0.555420   0.056482  -9.834  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.884 on 390 degrees of freedom
## Multiple R-squared:  0.7203, Adjusted R-squared:  0.711
## F-statistic: 77.28 on 13 and 390 DF,  p-value: < 2.2e-16
```

Evaluate on the test data

We have 90% correlation between the predicted and target home prices. The mse of 18.9 is a rmse of 4.347, so we are off by about $4,347 on average for the homes in the neighborhood. That's pretty good.

```
pred_lm <- predict(lm1, newdata=test)
cor(pred_lm, test$medv)
```

```
## [1] 0.900081
```

```
mse_lm <- mean((pred_lm - test$medv)^2)
mse_lm
```

```
## [1] 18.93611
```

SVM Linear

Now we try SVM regression with a linear kernel.

```
library(e1071)
svm_fit1 <- svm(medv~., data=train, kernel="linear", cost=10, scale=FALSE)
summary(svm_fit1)
```

```
##
## Call:
## svm(formula = medv ~ ., data = train, kernel = "linear", cost = 10,
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  10
```

```
##       gamma:  0.07692308
##     epsilon:  0.1
##
##
## Number of Support Vectors:  392
```

```
svm_pred1 <- predict(svm_fit1, newdata=test)
cor(svm_pred1, test$medv)
```

```
## [1] 0.912717
```

```
mse_svm1 <- mean((svm_pred1 - test$medv)^2)
mse_svm1
```

```
## [1] 18.4258
```

The tune() function

The linear SVM did slightly better than linear regression. Next we perform some tuning to find the best cost parameter. The tune() function tries to find optimal hyperparameters for the svm using a grid search. This involves trying all the suggested parameters in a cross-validation scheme. Here we asked it to try several different cost parameters. The best model can be extracted from the tune results.

```
tune_svm1 <- tune(svm, medv~., data=train, kernel="linear",
               ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tune_svm1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 26.97133
##
## - Detailed performance results:
##    cost    error dispersion
## 1 1e-03 44.80447   17.51189
## 2 1e-02 28.50440   12.25274
## 3 1e-01 26.97133   11.04768
## 4 1e+00 27.07120   11.18866
## 5 5e+00 27.07697   11.17431
## 6 1e+01 27.08590   11.19209
## 7 1e+02 27.06015   11.16789
```

```
best_mod1 <- tune_svm1$best.model
summary(best_mod1)
```

```
##
## Call:
## best.tune(method = svm, train.x = medv ~ ., data = train, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")
##
```

```
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  0.1
##       gamma:  0.07692308
##     epsilon:  0.1
##
##
## Number of Support Vectors:  318
```

Use the best model

The best model parameters were selected on the train set. It will not necessarily perform better on the test data, and indeed it performed slightly worse.

```
svm_pred2 <- predict(best_mod1, newdata=test)
cor(svm_pred2, test$medv)
```

```
## [1] 0.9074539
```

```
mse_svm2 <- mean((svm_pred2 - test$medv)^2)
mse_svm2
```

```
## [1] 19.36801
```

Try the radial kernel

For radial kernel we have an additional hyperparameter, gamma.

```
svm_fit2 <- svm(medv~., data=train, kernel="radial", cost=1, gamma=1, scale=FALSE)
summary(svm_fit2)
```

```
##
## Call:
## svm(formula = medv ~ ., data = train, kernel = "radial", cost = 1,
##     gamma = 1, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  398
```

```
svm_pred2 <- predict(svm_fit2, newdata=test)
cor(svm_pred2, test$medv)
```

```
## [1] -0.01849452
```

```
mse_svm2 <- mean((svm_pred2 - test$medv)^2)
mse_svm2
```

```
## [1] 95.05349
```

Tune the hyperparameters

```r
set.seed(1234)
tune_svm2 = tune(svm, medv~., data=train, kernel="radial",
                 ranges=list(cost=c(0.1,1,10,100,1000),
                 gamma=c(0.5,1,2,3,4)))
summary(tune_svm2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.5
##
## - best performance: 22.04514
##
## - Detailed performance results:
##     cost gamma    error dispersion
## 1  1e-01   0.5 58.00975  26.042036
## 2  1e+00   0.5 28.57541  13.694933
## 3  1e+01   0.5 22.04514   8.732259
## 4  1e+02   0.5 22.09964   8.574124
## 5  1e+03   0.5 22.09964   8.574124
## 6  1e-01   1.0 68.63374  28.598236
## 7  1e+00   1.0 39.62097  20.094040
## 8  1e+01   1.0 33.87235  15.209260
## 9  1e+02   1.0 33.87235  15.209260
## 10 1e+03   1.0 33.87235  15.209260
## 11 1e-01   2.0 76.36623  29.667097
## 12 1e+00   2.0 53.73030  26.371605
## 13 1e+01   2.0 48.55734  22.109707
## 14 1e+02   2.0 48.55734  22.109707
## 15 1e+03   2.0 48.55734  22.109707
## 16 1e-01   3.0 79.27019  29.696176
## 17 1e+00   3.0 61.93589  28.314759
## 18 1e+01   3.0 57.29582  25.017593
## 19 1e+02   3.0 57.29582  25.017593
## 20 1e+03   3.0 57.29582  25.017593
## 21 1e-01   4.0 80.74464  29.632789
## 22 1e+00   4.0 66.95143  28.820064
## 23 1e+01   4.0 62.79526  26.044450
## 24 1e+02   4.0 62.79526  26.044450
## 25 1e+03   4.0 62.79526  26.044450
```

Again, best model isn't.

```
svm_pred3 <- predict(tune_svm2$best.model, newdata=test)
cor(svm_pred3, test$medv)
```

```
## [1] 0.8926458
```

```
mse_svm3 <- mean((svm_pred3 - test$medv)^2)
mse_svm3
```

```
## [1] 19.41668
```