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), 0.75*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
## -15.784  -2.568  -0.762   1.868  26.256
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  30.558662   5.975130   5.114 5.10e-07 ***
## crim         -0.111162   0.035252  -3.153 0.001748 **
```

```
## zn            0.046119   0.015120   3.050 0.002454 **
## indus        -0.008910   0.068418  -0.130 0.896460
## chas          2.613201   0.970514   2.693 0.007417 **
## nox         -17.335367   4.269429  -4.060 6.00e-05 ***
## rm            4.488140   0.491409   9.133  < 2e-16 ***
## age          -0.013698   0.015168  -0.903 0.367077
## dis          -1.513437   0.224709  -6.735 6.39e-11 ***
## rad           0.287861   0.073895   3.896 0.000117 ***
## tax          -0.012737   0.004172  -3.053 0.002432 **
## ptratio      -0.872409   0.149668  -5.829 1.23e-08 ***
## black         0.009616   0.003129   3.073 0.002279 **
## lstat        -0.395793   0.058779  -6.734 6.45e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.669 on 365 degrees of freedom
## Multiple R-squared:  0.7561, Adjusted R-squared:  0.7474
## F-statistic: 87.05 on 13 and 365 DF,  p-value: < 2.2e-16
```

Evaluate on the test data

We have 82% correlation between the predicted and target home prices. The rmse of 5.09, so we are off by about $5,091 on average for the homes in the neighborhood.

```
pred_lm <- predict(lm1, newdata=test)
cor_lm <- cor(pred_lm, test$medv)
rmse_lm <- sqrt(mean((pred_lm - test$medv)^2))
print(paste("cor = ", cor_lm))
```

```
## [1] "cor =  0.822386579371224"
```

```
print(paste("rmse = ", rmse_lm))
```

```
## [1] "rmse =  5.09168700891912"
```

SVM Linear

Now we try SVM regression with a linear kernel.

```
library(e1071)
train_svm <- train[, c(1, 5, 6, 11, 13, 14)]
test_svm <- test[, c(1, 5, 6, 11, 13, 14)]
svm_fit1 <- svm(medv~., data=train_svm, kernel="linear", scale=TRUE)
summary(svm_fit1)
```

```
##
## Call:
## svm(formula = medv ~ ., data = train_svm, kernel = "linear", scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.2
##     epsilon:  0.1
##
```

```
##
## Number of Support Vectors:  298
```

```
svm_pred1 <- predict(svm_fit1, newdata=test_svm)
cor_svm1 <- cor(svm_pred1, test$medv)
rmse_svm1 <- sqrt(mean((svm_pred1 - test$medv)^2))
print(paste("cor = ", cor_svm1))
```

```
## [1] "cor =  0.757247072134415"
```

```
print(paste("rmse = ", rmse_svm1))
```

```
## [1] "rmse =  5.83980722148527"
```

The tune() function

The linear SVM did not do as well as 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_svm, kernel="linear", scale=TRUE,
                  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: 27.45368
##
## - Detailed performance results:
##    cost    error dispersion
## 1 1e-03 45.97112   16.34500
## 2 1e-02 28.29355   14.16438
## 3 1e-01 27.45368   14.14724
## 4 1e+00 27.63249   14.18857
## 5 5e+00 27.60912   14.19561
## 6 1e+01 27.61104   14.19611
## 7 1e+02 27.58507   14.20083
```

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

```
##
## Call:
## best.tune(method = svm, train.x = medv ~ ., data = train_svm, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100)), kernel = "linear", scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
```

```
##  SVM-Kernel:  linear
##        cost:  0.1
##       gamma:  0.2
##     epsilon:  0.1
##
##
## Number of Support Vectors:  298
```

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 did not.

```
svm_pred2 <- predict(best_mod1, newdata=test_svm)
cor_svm2 <- cor(svm_pred2, test$medv)
rmse_svm2 <- sqrt(mean((svm_pred2 - test$medv)^2))
print(paste("cor = ", cor_svm2))
```

```
## [1] "cor =  0.757990122077014"
```

```
print(paste("rmse = ", rmse_svm2))
```

```
## [1] "rmse =  5.83835049636945"
```

Try the radial kernel

For radial kernel we have an additional hyperparameter, gamma.

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

```
##
## Call:
## svm(formula = medv ~ ., data = train_svm, kernel = "radial", cost = 1,
##     gamma = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  270
```

```
svm_pred3 <- predict(svm_fit2, newdata=test_svm)
cor_svm_radial <- cor(svm_pred3, test$medv)
rmse_svm_radial <- sqrt(mean((svm_pred3 - test$medv)^2))
print(paste("cor = ", cor_svm_radial))
```

```
## [1] "cor =  0.85263118179067"
```

```
print(paste("rmse = ", rmse_svm_radial))
```

```
## [1] "rmse =  4.68818762032307"
```

Tune the hyperparameters

```
set.seed(1234)
tune_svm2 = tune(svm, medv~., data=train_svm, kernel="radial", scale=TRUE,
                 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: 14.00149
##
## - Detailed performance results:
##     cost gamma    error dispersion
## 1  1e-01   0.5 36.53675  21.918765
## 2  1e+00   0.5 16.73496   8.682164
## 3  1e+01   0.5 14.00149   4.707334
## 4  1e+02   0.5 19.61875   5.198223
## 5  1e+03   0.5 30.29342   9.804924
## 6  1e-01   1.0 47.29211  25.134468
## 7  1e+00   1.0 20.94124  10.333790
## 8  1e+01   1.0 18.40845   5.251805
## 9  1e+02   1.0 22.18650   4.486435
## 10 1e+03   1.0 37.28593   7.295399
## 11 1e-01   2.0 59.43867  27.830696
## 12 1e+00   2.0 29.64805  16.655042
## 13 1e+01   2.0 26.17151   7.909560
## 14 1e+02   2.0 32.77222   6.654787
## 15 1e+03   2.0 44.09527  10.536241
## 16 1e-01   3.0 66.13913  28.568993
## 17 1e+00   3.0 35.91291  21.560048
## 18 1e+01   3.0 32.56440  12.760168
## 19 1e+02   3.0 37.64863  11.074919
## 20 1e+03   3.0 50.38632  12.596429
## 21 1e-01   4.0 70.36891  28.839892
## 22 1e+00   4.0 40.58762  24.605328
## 23 1e+01   4.0 38.13857  17.086072
## 24 1e+02   4.0 41.81078  15.518274
## 25 1e+03   4.0 46.62946  13.711731
```

Tuning

Tuning was time consuming but found the best hyperparameters.

```
svm_pred3 <- predict(tune_svm2$best.model, newdata=test_svm)
cor_svm3 <- cor(svm_pred3, test$medv)
rmse_svm3 <- sqrt(mean((svm_pred3 - test$medv)^2))
print(paste("cor = ", cor_svm3))
```

```
## [1] "cor =  0.899398634297311"
```

```
print(paste("rmse = ", rmse_svm3))
```

```
## [1] "rmse =  3.92714589051774"
```