svm_method

Jianfeng Zhang

March 27, 2018

R Markdown

```
data <- read.csv("C:/Users/jianf/Desktop/Project/alldata7.csv")

####
### SVM regression
library(e1071)
colnames(data)
```

```
##  [1] "GDP"           "CPI"           "Interest_Rate" "PMI"
##  [5] "Unemployment"  "Crude_Oil"     "IP"            "Payroll"
##  [9] "Christmas"     "Thx"           "Super"         "Sales"
```

```
set.seed(1)
train_num <- sample(1:nrow(data), 0.8*nrow(data))
train<- data[train_num,]
test <- data[-train_num,]


set.seed(1)
svm_fit <- svm(Sales~. ,data= train)
svm_fit
```

```
##
## Call:
## svm(formula = Sales ~ ., data = train)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.09090909
##     epsilon:  0.1
##
##
## Number of Support Vectors:  61
```

```
predictSale_train <- predict(svm_fit, train[,-12])
mean((predictSale_train - train$Sales)^2)
```

```
## [1] 2981751
```

```
predictSale_test <- predict(svm_fit, test[,-12])
mean((predictSale_test - test$Sales)^2)
```

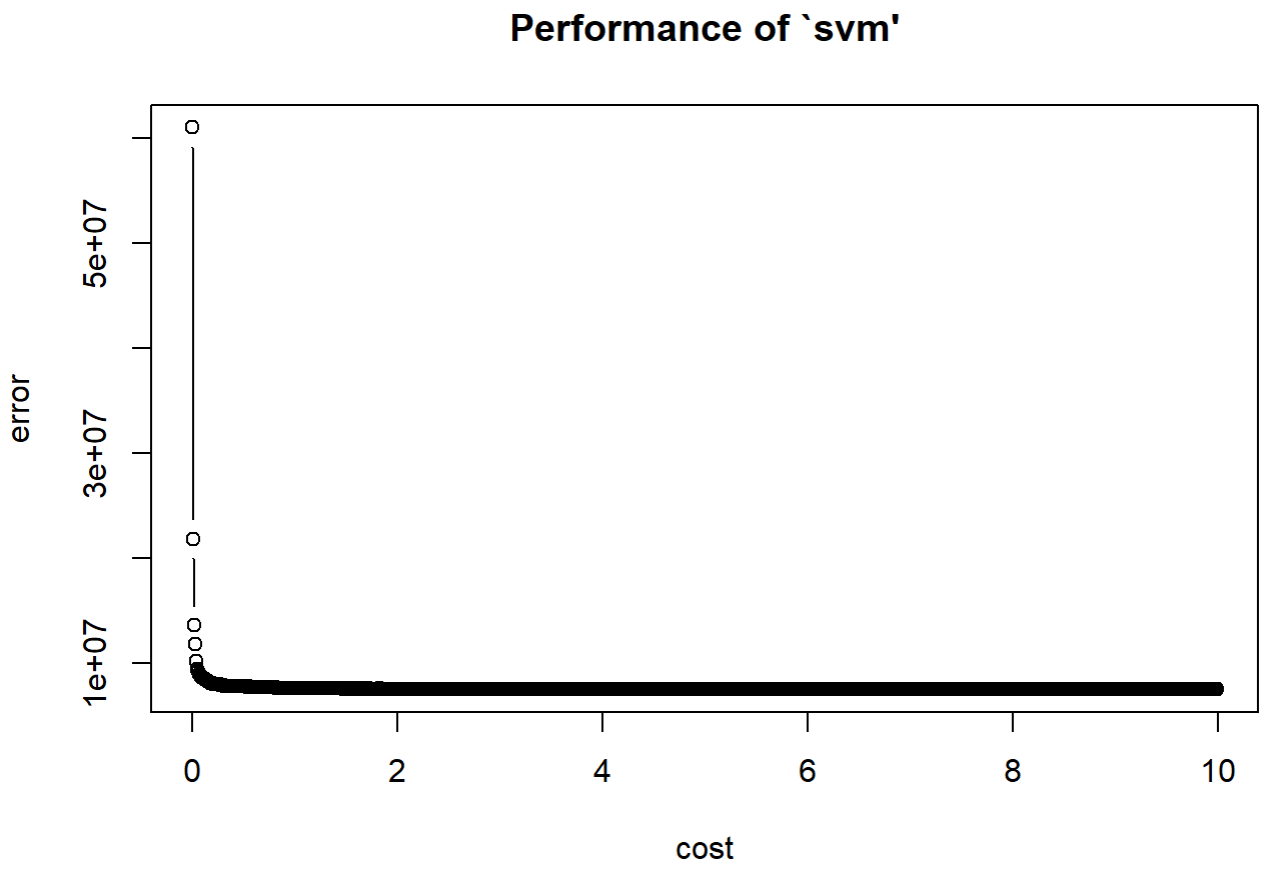
```
## [1] 1607016
```

```
### Tune svm
tc <- tune.control(cross = 5)

set.seed(1)
tuneResult_linear <- tune.svm(Sales ~ ., data = train, cost = seq(0.001,10, 0.01), kernel = "
linear",
                              tunecontrol = tc)
tuneResult_linear
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##   cost
##  9.921
##
## - best performance: 7466976
```

```
plot(tuneResult_linear)
```




```
linear_predict_train <- predict(tuneResult_linear$best.model, train[,-12])
mean( (linear_predict_train - train$Sales)^2 )
```

```
## [1] 6966581
```

```
linear_predict_test <- predict(tuneResult_linear$best.model, test[,-12])
mean( (linear_predict_test - test$Sales)^2)
```

```
## [1] 5303916
```

```
### Train R-square
linear_rss_train <- sum(  (linear_predict_train - train$Sales)^2 )
linear_sst_train <- sum( ( train$Sales - mean(train$Sales))^2)
r_square_linear_train <- 1- (linear_rss_train / linear_sst_train)
r_square_linear_train
```

```
## [1] 0.9557027
```

```
### Test R-square
linear_ssr <- sum(  (linear_predict_test - mean(test$Sales))^2 )
linear_sst <- sum( ( test$Sales - mean(test$Sales))^2)
r_square_linear <- linear_ssr / linear_sst
```

```
r_square_linear
```

```
## [1] 0.9111777
```

```
RMSE_linear_test <- sqrt(mean( (linear_predict_test - test$Sales)^2))
RMSE_linear_test
```

```
## [1] 2303.023
```

```
### Radial
set.seed(1)
tuneResult_radial <- tune.svm(Sales ~ ., data = train, cost = seq(0.001, 10, 0.1),gamma = seq
(0.001, 5, 0.1), kernel = "radial",tunecontrol = tc)
tuneResult_radial
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  gamma  cost
##  0.101 9.801
##
## - best performance: 3870994
```

```
plot(tuneResult_radial)


radial_predict_train <- predict(tuneResult_radial$best.model, train[,-12])
mean( (radial_predict_train - train$Sales)^2 )
```

```
## [1] 686635.4
```

```
radial_predict_test <- predict(tuneResult_radial$best.model, test[,-12])
mean( (radial_predict_test - test$Sales)^2)
```

```
## [1] 1363443
```

```
### R-squared of train data set
radial_rss_train <- sum(  (radial_predict_train - train$Sales)^2 )
radial_sst_train <- sum( ( train$Sales - mean(train$Sales))^2)
r_square_radial_train <- 1- ( radial_rss_train / radial_sst_train)
r_square_radial_train
```

```
## [1] 0.995634
```

```
### R-squared of test data set
radial_ssr <- sum(  (radial_predict_test - mean(test$Sales))^2 )

radial_rss <- sum(  (radial_predict_test - test$Sales)^2 )

radial_sst <- sum( ( test$Sales - mean(test$Sales))^2)
r_square <-  1 - (radial_rss / radial_sst)
r_square
```

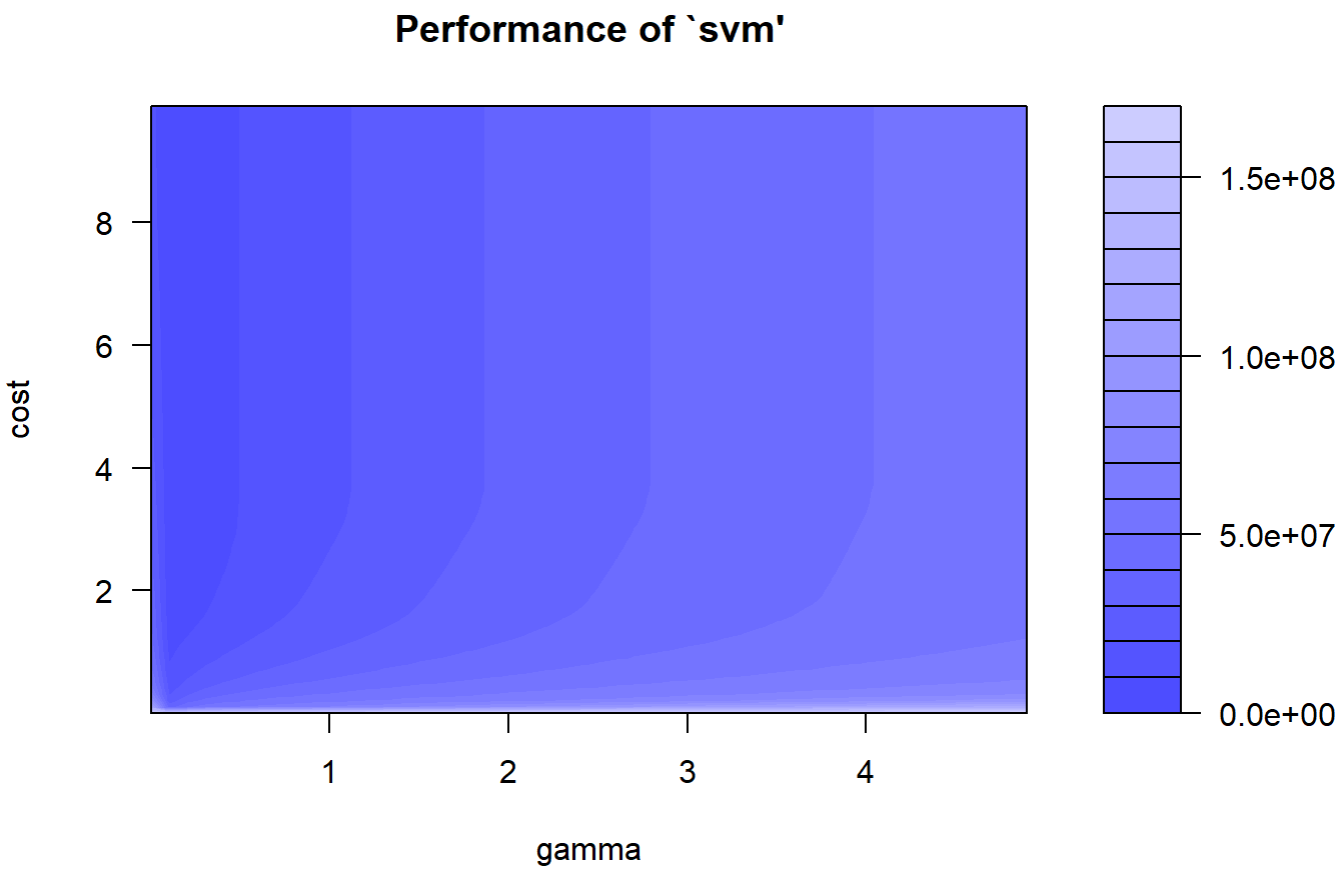
```
## [1] 0.9902732
```

```
RMSE_radial_test <- sqrt( mean( (radial_predict_test - test$Sales)^2) )
RMSE_radial_test
```

```
## [1] 1167.666
```

```
### Importance of variables

library(rminer)
```




```
M <- fit(Sales~., data=train, model="svm", kpar=list(sigma=0.10), C=2)
svm.imp <- Importance(M, data=train)
```

```
svm.imp
```

```
## $value
##  [1] 3795.6698 3029.9417  399.7570 1573.2090 1921.3280  875.4624  937.2465
##  [8] 3165.9889 2372.8124 1108.4167  553.2534    0.0000
##
## $imp
##  [1] 0.19235054 0.15354627 0.02025821 0.07972443 0.09736582 0.04436521
##  [7] 0.04749619 0.16044064 0.12024538 0.05617047 0.02803684 0.00000000
##
## $sresponses
## $sresponses[[1]]
## $sresponses[[1]]$n
## [1] "GDP"
##
## $sresponses[[1]]$l
## [1] 7
##
## $sresponses[[1]]$x
## [1]  9189.454 10527.607 11865.759 13203.912 14542.065 15880.217 17218.370
##
## $sresponses[[1]]$y
## [1]  8425.347  7509.906  7948.359  9854.578 12995.271 16823.959 20634.070
##
## $sresponses[[1]]$yy
## NULL
##
##
## $sresponses[[2]]
## $sresponses[[2]]$n
## [1] "CPI"
##
## $sresponses[[2]]$l
## [1] 7
##
## $sresponses[[2]]$x
## [1] 139.30 157.22 175.14 193.06 210.98 228.90 246.82
##
## $sresponses[[2]]$y
## [1]  9143.238  8476.232  8854.404 10345.365 12796.556 15852.091 19034.820
##
## $sresponses[[2]]$yy
## NULL
##
##
## $sresponses[[3]]
## $sresponses[[3]]$n
## [1] "Interest_Rate"
##
## $sresponses[[3]]$l
## [1] 7
##
## $sresponses[[3]]$x
```

```
## [1] 0.010000 1.026667 2.043333 3.060000 4.076667 5.093333 6.110000
##
## $sresponses[[3]]$y
## [1] 11266.98 10758.45 10518.63 10539.60 10808.42 11312.09 12035.90
##
## $sresponses[[3]]$yy
## NULL
##
##
## $sresponses[[4]]
## $sresponses[[4]]$n
## [1] "PMI"
##
## $sresponses[[4]]$l
## [1] 7
##
## $sresponses[[4]]$x
## [1] 116.4 129.9 143.4 156.9 170.4 183.9 197.4
##
## $sresponses[[4]]$y
## [1] 10513.381  9954.466 10012.826 10720.783 12017.393 13754.184 15721.559
##
## $sresponses[[4]]$yy
## NULL
##
##
## $sresponses[[5]]
## $sresponses[[5]]$n
## [1] "Unemployment"
##
## $sresponses[[5]]$l
## [1] 7
##
## $sresponses[[5]]$x
## [1] 3.100000 4.233333 5.366667 6.500000 7.633333 8.766667 9.900000
##
## $sresponses[[5]]$y
## [1] 12204.46 10812.05 10510.25 11401.27 13211.67 15427.46 17533.75
##
## $sresponses[[5]]$yy
## NULL
##
##
## $sresponses[[6]]
## $sresponses[[6]]$n
## [1] "Crude_Oil"
##
## $sresponses[[6]]$l
## [1] 7
##
## $sresponses[[6]]$x
## [1]  11.35000  31.68667  52.02333  72.36000  92.69667 113.03333 133.37000
##
## $sresponses[[6]]$y
```

```
## [1] 12869.876 11414.276 10358.658  9842.512  9951.464 10696.151 11996.719
##
## $sresponses[[6]]$yy
## NULL
##
##
## $sresponses[[7]]
## $sresponses[[7]]$n
## [1] "IP"
##
## $sresponses[[7]]$l
## [1] 7
##
## $sresponses[[7]]$x
## [1]  60.07640  68.52527  76.97413  85.42300  93.87187 102.32073 110.76960
##
## $sresponses[[7]]$y
## [1] 14186.02 12755.68 11585.43 10794.97 10477.79 10693.64 11452.19
##
## $sresponses[[7]]$yy
## NULL
##
##
## $sresponses[[8]]
## $sresponses[[8]]$n
## [1] "Payroll"
##
## $sresponses[[8]]$l
## [1] 7
##
## $sresponses[[8]]$x
## [1] 108369.0 114801.3 121233.7 127666.0 134098.3 140530.7 146963.0
##
## $sresponses[[8]]$y
## [1] 11082.733  9152.574  8395.912  9347.638 12120.789 16231.760 20705.498
##
## $sresponses[[8]]$yy
## NULL
##
##
## $sresponses[[9]]
## $sresponses[[9]]$n
## [1] "Christmas"
##
## $sresponses[[9]]$l
## [1] 7
##
## $sresponses[[9]]$x
## [1] 0.0000000 0.1666667 0.3333333 0.5000000 0.6666667 0.8333333 1.0000000
##
## $sresponses[[9]]$y
## [1] 10470.53 10743.59 11744.88 13242.82 14946.55 16596.93 18025.21
##
## $sresponses[[9]]$yy
```

```
## NULL
##
##
## $sresponses[[10]]
## $sresponses[[10]]$n
## [1] "Thx"
##
## $sresponses[[10]]$l
## [1] 7
##
## $sresponses[[10]]$x
## [1] 0.0000000 0.1666667 0.3333333 0.5000000 0.6666667 0.8333333 1.0000000
##
## $sresponses[[10]]$y
## [1] 10729.12 10425.81 10611.77 11202.11 12082.18 13143.01 14300.44
##
## $sresponses[[10]]$yy
## NULL
##
##
## $sresponses[[11]]
## $sresponses[[11]]$n
## [1] "Super"
##
## $sresponses[[11]]$l
## [1] 7
##
## $sresponses[[11]]$x
## [1] 0.0000000 0.1666667 0.3333333 0.5000000 0.6666667 0.8333333 1.0000000
##
## $sresponses[[11]]$y
## [1] 11105.975 10119.156  9429.509  9064.310  9050.907  9417.937 10180.796
##
## $sresponses[[11]]$yy
## NULL
##
##
## $sresponses[[12]]
## NULL
##
##
## $data
## NULL
##
## $method
## [1] "1D-SA"
##
## $measure
## [1] "AAD"
##
## $agg
## [1] -1
##
## $nclasses
```

```
## [1] 0
##
## $inputs
##  [1]  1  2  3  4  5  6  7  8  9 10 11
##
## $Llevels
##  [1]  7  7  7  7  7  7  7  7  7  7  7 NA
##
## $interactions
## NULL
```

```
svm.imp$imp
```

```
##  [1] 0.19235054 0.15354627 0.02025821 0.07972443 0.09736582 0.04436521
##  [7] 0.04749619 0.16044064 0.12024538 0.05617047 0.02803684 0.00000000
```

```
rank(-svm.imp$imp)
```

```
##  [1]  1  3 11  6  5  9  8  2  4  7 10 12
```

```
######################

data_plot <- read.csv("C:/Users/jianf/Desktop/Project/dataforplot.csv")


linear_prediction_forplot <- predict(tuneResult_linear$best.model, data_plot[,-12])

radial_prediction_forplot <- predict(tuneResult_radial$best.model, data_plot[,-12])

response_plot <- as.data.frame( cbind(linear_prediction_forplot,radial_prediction_forplot))
head(response_plot)
```

```
##   linear_prediction_forplot radial_prediction_forplot
## 1                  -180.045                  3859.344
## 2                  7495.147                  7664.151
## 3                  1833.472                  2723.397
## 4                  3405.806                  2995.250
## 5                  4536.352                  3161.677
## 6                  4677.997                  4593.546
```

```
write.csv(response_plot, "C:/Users/jianf/Desktop/Project/SVMresponse_dataforplot.csv")
```