

SVM_Regression

Code ▼

Atmin Sheth

data information

This notebook will use a dataset found on the UCI Machine Learning Repository: Fanaee-T, Hadi, and Gama, Joao, 'Event labeling combining ensemble detectors and background knowledge', Progress in Artificial Intelligence (2013): pp. 1-15, Springer Berlin Heidelberg, <https://archive.ics.uci.edu> (<https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset> (<https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset>)). The data describes hourly bike rental numbers from the Capital Bikeshare system between 2011 and 2012. Target: cnt

library

Hide

```
library(e1071)
```

Data Cleaning

Hide

```
bikeSharing <- read.csv("Data/bike-sharing.csv")
bikeSharing <- bikeSharing[,c(3:14,17)] #remove instant, dteday, casual , registered
bikeSharing <- bikeSharing[complete.cases(bikeSharing),]#remove incomplete
str(bikeSharing)
```

```
'data.frame': 17379 obs. of 13 variables:
 $ season : int 1 1 1 1 1 1 1 1 1 1 ...
 $ yr : int 0 0 0 0 0 0 0 0 0 0 ...
 $ mnth : int 1 1 1 1 1 1 1 1 1 1 ...
 $ hr : int 0 1 2 3 4 5 6 7 8 9 ...
 $ holiday : int 0 0 0 0 0 0 0 0 0 0 ...
 $ weekday : int 6 6 6 6 6 6 6 6 6 6 ...
 $ workingday: int 0 0 0 0 0 0 0 0 0 0 ...
 $ weathersit: int 1 1 1 1 1 2 1 1 1 1 ...
 $ temp : num 0.24 0.22 0.22 0.24 0.24 0.24 0.22 0.2 0.24 0.32 ...
 $ atemp : num 0.288 0.273 0.273 0.288 0.288 ...
 $ hum : num 0.81 0.8 0.8 0.75 0.75 0.75 0.8 0.86 0.75 0.76 ...
 $ windspeed : num 0 0 0 0 0 0.0896 0 0 0 0 ...
 $ cnt : int 16 40 32 13 1 1 2 3 8 14 ...
```

Data Expletation

Hide

```
set.seed(1234)
spec <- c(train=.6,test=.2, validate=.2)
i <- sample(cut(1:nrow(bikeSharing),
               nrow(bikeSharing)*cumsum(c(0,spec))), labels=names(spec))
train <- bikeSharing[i=="train",]
test <- bikeSharing[i=="test",]
valid <- bikeSharing[i=="validate",]
```

#running linear regression

[Hide](#)

```
lm1 <- lm(cnt~hr+atemp+hum+windspeed+weathersit+weekday+holiday+yr+mnth, data=train)
pred <- predict(lm1, newdata=test)
cor_lm1 <- cor(pred,test$cnt)
mse_lm1 <- mean((pred-test$cnt)^2)
print(paste("cor= ",cor_lm1))
```

```
[1] "cor= 0.628313667957987"
```

[Hide](#)

```
print(paste("mse=",mse_lm1))
```

```
[1] "mse= 18736.4441914309"
```

#trying svm to regression

[Hide](#)

```
svm1 <- svm(cnt~. ,data=train, kernel="linear", cost=10, scale=TRUE)
summary(svm1)
```

Call:

```
svm(formula = cnt ~ ., data = train, kernel = "linear", cost = 10, scale = TRUE)
```

Parameters:

```
SVM-Type:  eps-regression
SVM-Kernel: linear
cost:      10
gamma:     0.08333333
epsilon:   0.1
```

Number of Support Vectors: 8839

[Hide](#)

```
pred <- predict(svm1,newdata = test)
cor_svm1 <- cor(pred,test$cnt)
mse_svm1 <- mean((pred-test$cnt)^2)
print(paste("cor= ", cor_svm1))
```

```
[1] "cor= 0.625686249115653"
```

[Hide](#)

```
print(paste("mse= ", mse_svm1))
```

```
[1] "mse= 19787.6405694574"
```

#tune regression let's tune the model to see different cost that can be use

[Hide](#)

```
v <- valid[c(1:500),]
tune_svm1 <- tune(svm,cnt~.,data=v,kernel="linear",
                 ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100)))
```

[illegible]

Hide

```
summary(tune_svm1)
```

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation
- best parameters:

cost
<dbl>

100

1 row

- best performance: 3976.029
- Detailed performance results:

cost <dbl>	error <dbl>	dispersion <dbl>
1e-03	5300.624	1426.765
1e-02	4799.395	1458.886
1e-01	4609.292	1381.434
1e+00	4387.603	1304.959
5e+00	4147.965	1240.034
1e+01	4068.498	1236.034
1e+02	3976.029	1269.133

7 rows

NA

Hide

```
pred <- predict(tune_svm1$best.model,newdata=test)
cor<- cor(pred,test$cnt)
mse <- mean((pred-test$cnt)^2)
print(paste("cor=",cor))
```

```
[1] "cor= 0.570556112769412"
```

Hide

```
print(paste("mase=",mse))
```

```
[1] "mase= 31880.5783080099"
```

Hide

```
svm2 <- svm(cnt~. ,data=train, kernel="polynomial", cost=5, scale=TRUE)
summary(svm2)
```

Call:

```
svm(formula = cnt ~ ., data = train, kernel = "polynomial", cost = 5, scale = TRUE)
```

Parameters:

SVM-Type: eps-regression

SVM-Kernel: polynomial

cost: 5

degree: 3

gamma: 0.08333333

coef.0: 0

epsilon: 0.1

Number of Support Vectors: 8347

Hide

```
pred <- predict(svm2, newdata = test)
cor_svm2 <- cor(pred,test$cnt)
mse_svm2 <- mean((pred-test$cnt)^2)
print(paste("cor= ", cor_svm2))
```

```
[1] "cor= 0.689655797359039"
```

Hide

```
print(paste("mse= ", mse_svm2))
```

```
[1] "mse= 16794.4588227018"
```

tune polinomial

Hide

```
tune_svm2 <- tune(svm,cnt~. ,data=v,kernel="polynomial",
                 ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100)))
```

[illegible]

WARNING: reaching max number of iterations

Warning: Variable(s) 'yr' constant. Cannot scale data.

WARNING: reaching max number of iterations

Warning: Variable(s) 'yr' constant. Cannot scale data.

WARNING: reaching max number of iterations

Warning: Variable(s) 'yr' constant. Cannot scale data.

WARNING: reaching max number of iterations

Warning: Variable(s) 'yr' constant. Cannot scale data.

WARNING: reaching max number of iterations

Warning: Variable(s) 'yr' constant. Cannot scale data.

WARNING: reaching max number of iterations

Warning: Variable(s) 'yr' constant. Cannot scale data.

WARNING: reaching max number of iterations

Warning: Variable(s) 'yr' constant. Cannot scale data.

WARNING: reaching max number of iterations

Warning: Variable(s) 'yr' constant. Cannot scale data.

WARNING: reaching max number of iterations

Warning: Variable(s) 'yr' constant. Cannot scale data.
WARNING: reaching max number of iterations
Warning: Variable(s) 'yr' constant. Cannot scale data.
WARNING: reaching max number of iterations

Hide

summary(tune_svm2)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation
- best parameters:

	cost <dbl>
	100
1 row	

- best performance: 3249.088
- Detailed performance results:

cost <dbl>	error <dbl>	dispersion <dbl>
1e-03	4813.290	1319.6970
1e-02	4171.131	1061.5970
1e-01	3497.608	1053.4462
1e+00	3358.346	1045.3136
5e+00	3290.294	1030.9388
1e+01	3260.885	1029.2913
1e+02	3249.088	976.8794
7 rows		

NA

Hide


```
pred <- predict(tune_svm2$best.model,newdata=test)
cor_svm2<- cor(pred,test$cnt)
mse_svm2<- mean((pred-test$cnt)^2)
print(paste("cor=",cor_svm2))
```

```
[1] "cor= 0.381870161430098"
```

[Hide](#)

```
print(paste("mse=",mse_svm2))
```

```
[1] "mse= 105524.218304026"
```

##Try radial

[Hide](#)

```
svm3 <- svm(cnt~. ,data=train, kernel="radial", cost=10, gamma=1, scale=TRUE)
summary(svm3)
```

Call:

```
svm(formula = cnt ~ ., data = train, kernel = "radial", cost = 10, gamma = 1, scale = TRUE)
```

Parameters:

```
SVM-Type:  eps-regression
SVM-Kernel: radial
cost: 10
gamma: 1
epsilon: 0.1
```

```
Number of Support Vectors: 7900
```

[Hide](#)

```
pred <- predict(svm3,newdata=test)
cor_svm3 <-cor(pred,test$cnt)
mse_svm3 <- mean((pred-test$cnt)^2)
print(paste("cor=",cor_svm2))
```

```
[1] "cor= 0.381870161430098"
```

[Hide](#)

```
print(paste("mse=",mse_svm2))
```

```
[1] "mse= 105524.218304026"
```

#tune radial/ hyperparameter

[Hide](#)

```
tune_svm3 <- tune(svm,cnt~.,data=v,kernel="radial",  
                 ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100),  
                             gamma=c(0.5,1,2,3,4)))
```

[illegible]

[illegible]

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation
- best parameters:

	cost <dbl>	gamma <dbl>
	100	0.5

1 row

- best performance: 1587.069
- Detailed performance results:

cost <dbl>	gamma <dbl>	error <dbl>	dispersion <dbl>
1e-03	0.5	5647.964	1227.7214
1e-02	0.5	5644.991	1227.4138
1e-01	0.5	5615.399	1224.3331
1e+00	0.5	5328.708	1207.8492
5e+00	0.5	4385.150	1123.9152
1e+01	0.5	3643.776	1018.1635
1e+02	0.5	1587.069	563.3740
1e-03	1.0	5648.194	1227.7471
1e-02	1.0	5647.285	1227.6708
1e-01	1.0	5638.218	1226.9077

1-10 of 35 rows

Previous **1** 2 3 4 Next

NA

Hide

```
pred <- predict(tune_svm3$best.model,newdata=test)
cor_svm2<- cor(pred,test$cnt)
mse_svm2<- mean((pred-test$cnt)^2)
print(paste("cor=",cor_svm2))
```

```
[1] "cor= 0.371899507726941"
```

[Hide](#)

```
print(paste("mse=",mse_svm2))
```

```
[1] "mse= 41187.3236721223"
```

analysis

The data seems to be good as we are getting a low cor and high mse, for each kernel we are seeing the same or similar numbers, and with tuning cost 100 having the lowest error in the list for all 3 methods. This best cost of 100 is bring bringing the lowest error and a good dispersion. This shows the way the hyper plans layes out on the set giving the most optimal result