SVM Regression

Authors:

Andrew Sen
Neo Zhao

Date:

10/23/2022

Data

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 /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.

The predictors include:

- instant: record index
- dteday: date
- season: season (1:winter, 2:spring, 3:summer, 4:fall)
- yr: year (0:2011, 1:2012)
- mnth: month (1 - 12)
- hr: hour (0-23)
- holiday: (0:not a holiday, 1:holiday)
- weekday: (0:Sunday - 6:Saturday)
- workingday: (0:not a workday, 1:not weekend and not holiday)
- weathersit: hourly weather
    - 1: clear, few clouds, partly cloudy
    - 2: mist+cloudy, mist+broken clouds, mist+few clouds, mist
    - 3: light snow, light rain+thunderstorm+scattered clouds, light rain+scattered clouds
    - 4: heavy rain+ice pallets+thunderstorm+mist, snow+fog
- temp: normalized temperature in Celsius
- atemp: normalized feeling temperature in Celsius
- hum: normalized humidity divided by 100
- windspeed: normalized wind speed divided by 67

The possible target columns include:

- casual: number of rentals from casual users
- registered: number of rentals from registered users
- cnt: total rentals

We will be predicting cnt.

Data Cleaning

First, we will read in the data. Then, we will clean the data by removing rows with NAs and removing redundant columns. We will also remove the casual and registered columns because they are not independent from cnt.

```
# loading packages
library(e1071)

df <- read.csv("data/bike-sharing.csv")

# remove instant, dteday, season, yr, workday, temp, casual, registered
df <- df[,c(5:8, 10, 12:14, 17)]

# remove incomplete rows
df <- df[complete.cases(df),]
```

Data Exploration

First, we will divide the data into train and test data with an 80/20 split.

```
set.seed(1234)
i <- sample(nrow(df), size=.8*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]

# sampling 500 rows from train to tune models
i <- sample(nrow(train), size=500, replace=FALSE)
train_sample <- train[i,]
```

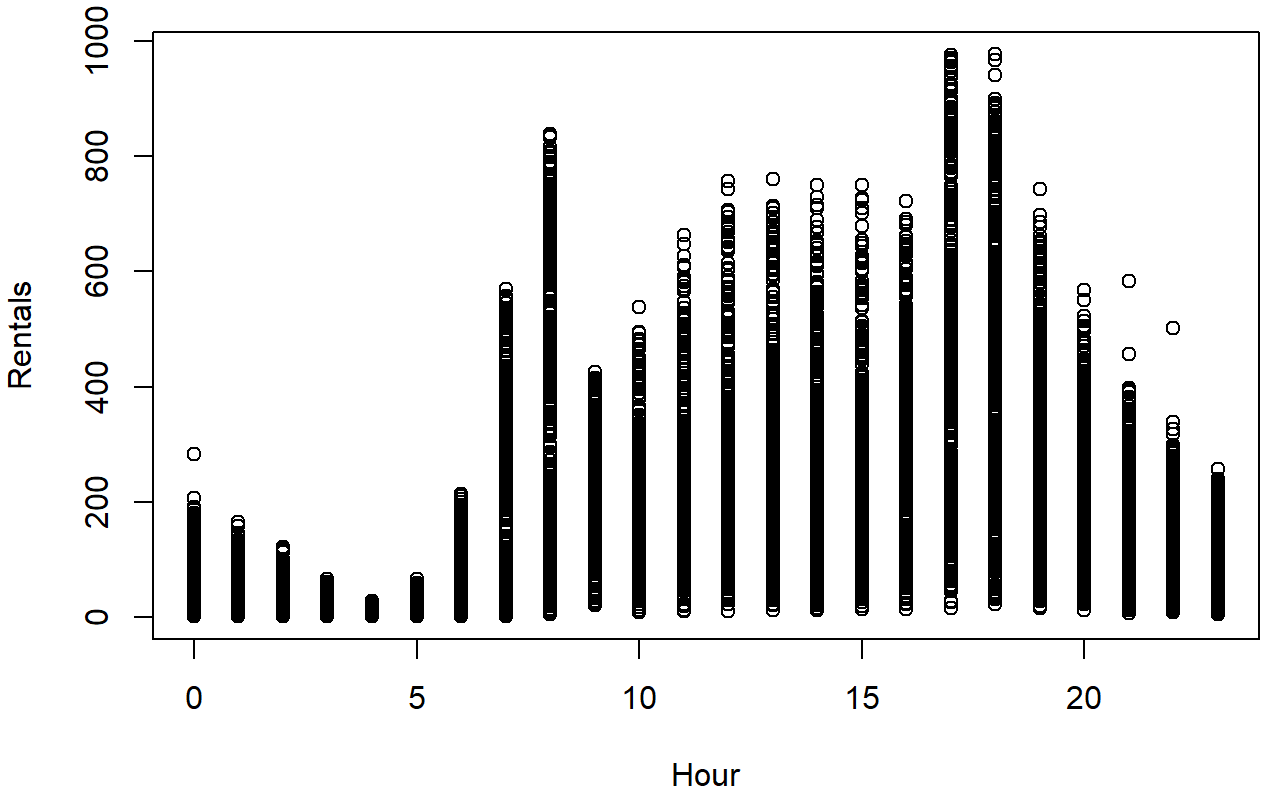
Now, we will explore the training data. First, we will see summaries of all the columns.

```
summary(train)
```

```
##       mnth              hr           holiday           weekday
##  Min.   : 1.000   Min.   : 0.00   Min.   :0.00000   Min.   :0.000
##  1st Qu.: 4.000   1st Qu.: 6.00   1st Qu.:0.00000   1st Qu.:1.000
##  Median : 7.000   Median :12.00   Median :0.00000   Median :3.000
##  Mean   : 6.549   Mean   :11.55   Mean   :0.02877   Mean   :3.011
##  3rd Qu.:10.000   3rd Qu.:18.00   3rd Qu.:0.00000   3rd Qu.:5.000
##  Max.   :12.000   Max.   :23.00   Max.   :1.00000   Max.   :6.000
##    weathersit        atemp             hum           windspeed
##  Min.   :1.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:1.000   1st Qu.:0.3333   1st Qu.:0.4800   1st Qu.:0.1045
##  Median :1.000   Median :0.4848   Median :0.6300   Median :0.1940
##  Mean   :1.421   Mean   :0.4760   Mean   :0.6265   Mean   :0.1896
##  3rd Qu.:2.000   3rd Qu.:0.6212   3rd Qu.:0.7800   3rd Qu.:0.2537
##  Max.   :4.000   Max.   :0.9848   Max.   :1.0000   Max.   :0.8507
##       cnt
##  Min.   :  1.0
##  1st Qu.: 41.0
##  Median :144.0
##  Mean   :190.8
##  3rd Qu.:282.0
##  Max.   :977.0
```

Next, we'll graph the number of bike rentals over the hour of the day.

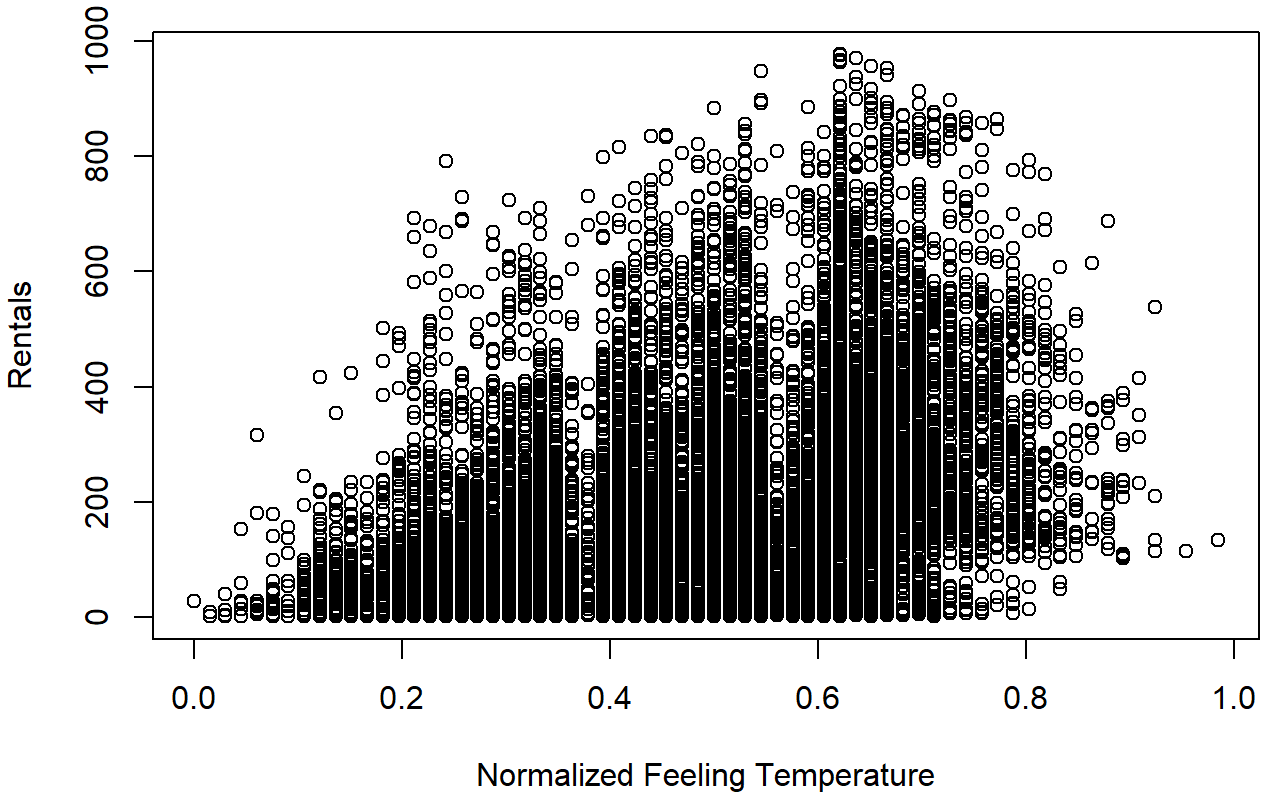
```
plot(train$hr, train$cnt, xlab="Hour", ylab="Rentals")
```



The graph shows that early morning and the afternoon tend to be the most popular times for renting a bike.

We will also plot bike rentals over feeling temperature.

```
plot(train$atemp, train$cnt, xlab="Normalized Feeling Temperature", ylab="Rentals")
```



This shows that bike rentals tend to increase if the temperature is warmer.

Linear Kernel

```
svm1 <- tune.svm(cnt~., data=train_sample, kernel="linear",
  cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)
)$best.model
summary(svm1)
```

```
##
## Call:
## best.svm(x = cnt ~ ., data = train_sample, cost = c(0.001, 0.01,
##     0.1, 1, 5, 10, 100), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.125
##     epsilon:  0.1
##
##
## Number of Support Vectors:  429
```

Polynomial Kernel

```
# tuning takes too long when passing a gamma range, so didn't do that here

svm2 <- tune.svm(cnt~., data=train_sample, kernel="polynomial",
  cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)
)$best.model
summary(svm2)
```

```
##
## Call:
## best.svm(x = cnt ~ ., data = train_sample, cost = c(0.001, 0.01,
##     0.1, 1, 5, 10, 100), kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  0.1
##      degree:  3
##       gamma:  0.125
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  442
```

Radial Kernel

```
svm3 <- tune.svm(cnt~., data=train_sample, kernel="radial",
  cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100),
  gamma=c(0.001, 0.01, 0.1, 1, 5, 10, 100)
)$best.model
summary(svm3)
```

```
##
## Call:
## best.svm(x = cnt ~ ., data = train_sample, gamma = c(0.001, 0.01,
##     0.1, 1, 5, 10, 100), cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100),
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  403
```

Test Models

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

```
## [1] "correlation of svm1: 0.588127428480474"
```

```
print(paste('mse of svm1:', mse_svm1))
```

```
## [1] "mse of svm1: 21988.3996635793"
```

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

```
## [1] "correlation of svm2: 0.514085266581459"
```

```
print(paste('mse of svm2:', mse_svm2))
```

```
## [1] "mse of svm2: 25061.4937866283"
```

```
pred <- predict(svm3, newdata=test)
cor_svm3 <- cor(pred, test$cnt)
mse_svm3 <- mean((pred - test$cnt)^2)
print(paste('correlation of svm3:', cor_svm3))
```

```
## [1] "correlation of svm3: 0.689762402429216"
```

```
print(paste('mse of svm3:', mse_svm3))
```

```
## [1] "mse of svm3: 17075.8894283124"
```

Conclusion

The linear kernel performed somewhat poorly because the data is very non-linear in nature. There is no linear model that would provided a good fit for this dataset, so SVM with a linear kernel is unlikely to significantly outperform simple linear regression, which would get a similar value for correlation.

The polynomial kernel performed poorly likely because the model overfitted the sample given to the tuning function. With only 500 rows to train on, the model is unlikely to sufficiently generalize to the test data. It is worth noting that I also was forced to avoid passing a range of gamma parameters because the tuning function was unable to converge in that scenario. The poor performance of this kernel may also be attributed to the fact that most of the factors of the data are categorical, so a polynomial kernel may not be best suited for this scenario.

The radial kernel performed the best of the three kernels. It makes sense that, assuming the model is not overfitted, this kernel would outperform the linear kernel due to the non-linear nature of the data. It is also possible that the radial kernel is more generalizable to the test data than the polynomial kernel given how few observations were used during tuning.