SVM Regression

Jeffrey Li, Valari Graham

10/23/2022

The data set being used is called **Bike Sharing** and is retrieved from the University of California, Irvine machine learning repository. Source: https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset

Load packages and clean data

Remove the first 10 columns from the data, only keeping weathersit, temp, atemp, hum, windspeed, cnt. Also, turn weather situation into a factor.

```
library(e1071)
library(MASS)
bike <- read.csv("hour.csv")
bike <- bike[,c(10:14,17)]
bike$weathersit <- factor(bike$weathersit)
```

Divide into train, test, validate

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

train <- bike[sample=="train",]
test <- bike[sample=="test",]
validate <- bike[sample=="validate",]
```

Explore data statistically

```
# The head function returns the column headers and first 6 rows
head(train)
```

```
##   weathersit temp  atemp  hum windspeed cnt
## 1          1 0.24 0.2879 0.81    0.0000  16
## 2          1 0.22 0.2727 0.80    0.0000  40
## 3          1 0.22 0.2727 0.80    0.0000  32
## 4          1 0.24 0.2879 0.75    0.0000  13
## 5          1 0.24 0.2879 0.75    0.0000   1
## 6          2 0.24 0.2576 0.75    0.0896   1
```

```
# The summary function gives more information on the model, residuals, coefficients
summary(train)
```

```
##  weathersit      temp            atemp             hum           windspeed
##  1:6874     Min.   :0.0200   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  2:2706     1st Qu.:0.3400   1st Qu.:0.3333   1st Qu.:0.4800   1st Qu.:0.1045
##  3: 845     Median :0.5000   Median :0.4848   Median :0.6300   Median :0.1940
##  4:   2     Mean   :0.4976   Mean   :0.4764   Mean   :0.6276   Mean   :0.1890
##             3rd Qu.:0.6600   3rd Qu.:0.6212   3rd Qu.:0.7800   3rd Qu.:0.2537
##             Max.   :0.9800   Max.   :0.9848   Max.   :1.0000   Max.   :0.8507
##       cnt
##  Min.   :  1.0
##  1st Qu.: 39.0
##  Median :143.0
##  Mean   :190.9
##  3rd Qu.:282.0
##  Max.   :976.0
```

```
# Find the mean of the total rental bike count column
mean(train$cnt)
```

```
## [1] 190.8513
```

Linear regression on the data

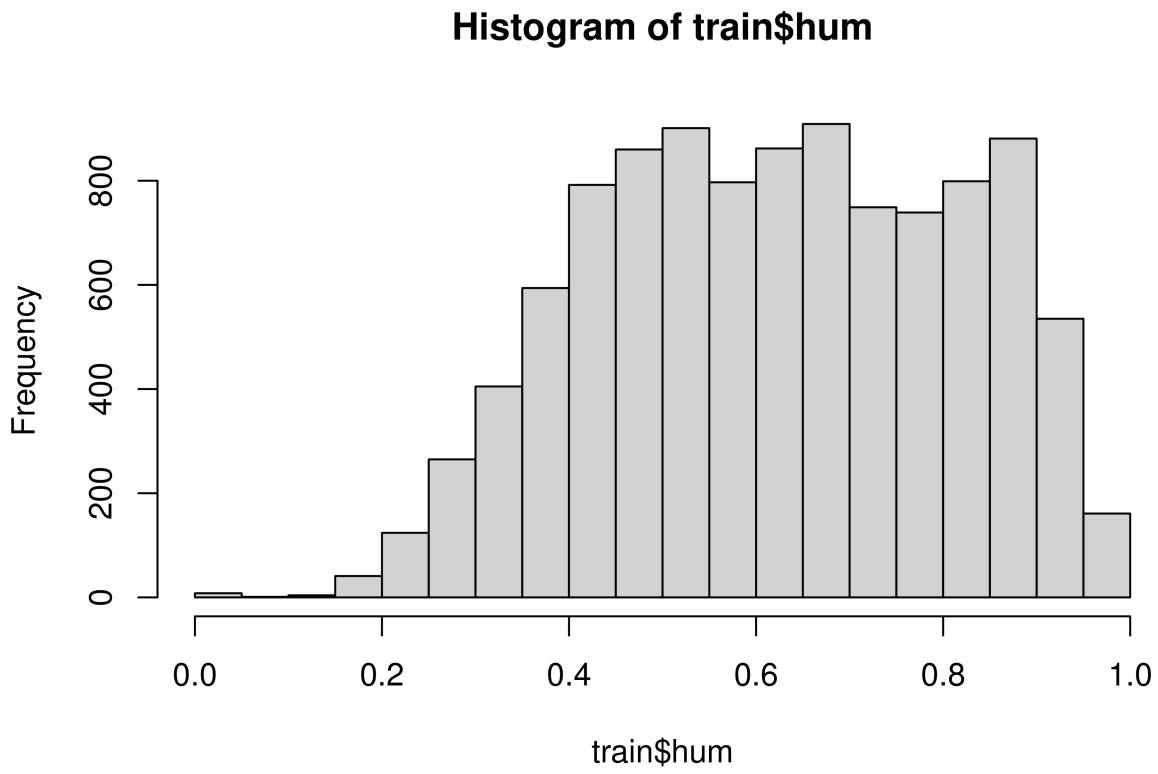
```
lm1 <- lm(cnt~., data=train)
summary(lm1)
```

```
##
## Call:
## lm(formula = cnt ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -343.31 -103.65  -33.65   64.91  711.94
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  166.1406     8.7102  19.074  < 2e-16 ***
## weathersit2   19.8198     3.7957   5.222 1.81e-07 ***
## weathersit3   -0.8795     6.3686  -0.138 0.890165
## weathersit4   99.7353   112.2929   0.888 0.374470
## temp         136.7241    54.8140   2.494 0.012635 *
## atemp        262.2920    61.5245   4.263 2.03e-05 ***
## hum         -291.0132     9.5229 -30.559  < 2e-16 ***
## windspeed     49.0452    14.0151   3.499 0.000468 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 158.7 on 10419 degrees of freedom
## Multiple R-squared:  0.2572, Adjusted R-squared:  0.2567
## F-statistic: 515.4 on 7 and 10419 DF,  p-value: < 2.2e-16
```

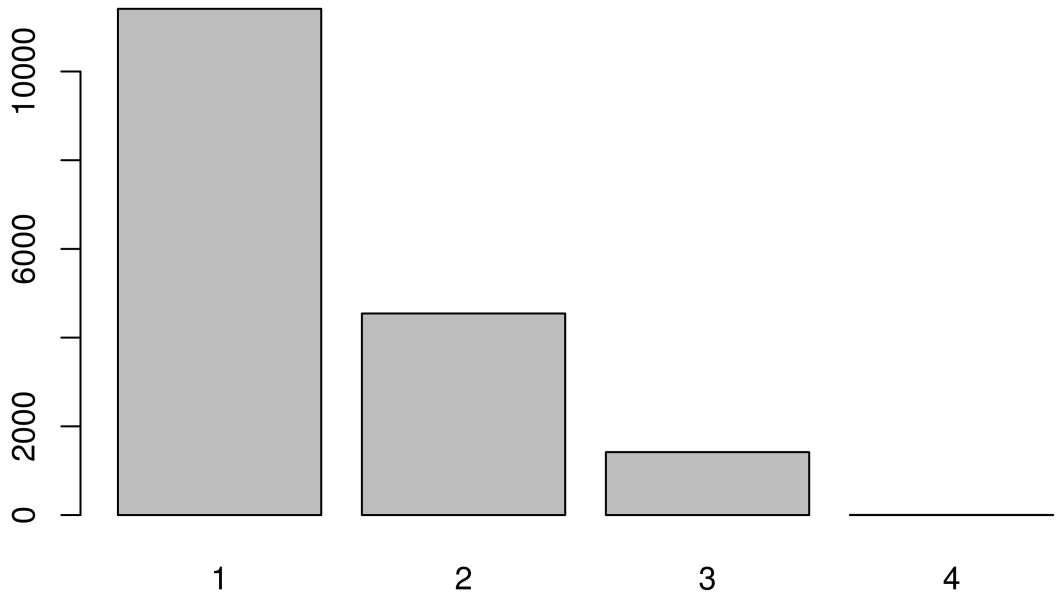
```
pred <- predict(lm1, newdata=test)
cor_lm1 <- cor(pred, test$cnt)
mse_lm1 <- mean((pred-test$cnt)^2)
```

Explore data graphically

```
# Histogram of humidity
hist(train$hum)
```




```
# Barplot of the weather situations
# 1: weather is "Clear, Few clouds, Partly cloudy, Partly cloudy"
# 2: weather is "Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist"
# 3: weather is "Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered cloud
# 4: weather is "Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog"
barplot(table(bike$weathersit))
```



SVM Regression: Linear Kernel

```
# Try linear kernel with a cost of 1
svm1 <- svm(cnt~., data=train, kernel="linear", cost=1, scale=TRUE)
summary(svm1)
```

```
##
## Call:
## svm(formula = cnt ~ ., data = train, kernel = "linear", cost = 1,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.125
##     epsilon:  0.1
##
##
## Number of Support Vectors:  9063
```

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

```
## [1] "Correlation:  0.52738752751985"
```

```
print(paste("MSE: ", mse_svm1))
```

```
## [1] "MSE:  23318.4611709304"
```

Tune to find the best linear C

```
tune_svm1 <- tune(svm, cnt~., data=validate, 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: 25934.8
##
## - Detailed performance results:
##    cost    error dispersion
## 1 1e-03 26349.19   3278.688
## 2 1e-02 25938.92   3152.105
## 3 1e-01 25934.80   3147.629
## 4 1e+00 25948.92   3152.499
## 5 5e+00 25951.73   3150.573
## 6 1e+01 25952.78   3153.394
## 7 1e+02 25951.57   3152.586
```

From the results, we see that the error values are relatively similar for the cost values of 1, 5, 10, 100.

Evaluate with best linear SVM

```
pred <- predict(tune_svm1$best.model, newdata=test)
cor_svm1_tune <- cor(pred, test$cnt)
mse_svm1_tune <- mean((pred - test$cnt)^2)
print(paste("Correlation: ", cor_svm1_tune))
```

```
## [1] "Correlation:  0.527989323902123"
```

```
print(paste("MSE: ", mse_svm1_tune))
```

```
## [1] "MSE:  23546.891385561"
```

When we evaluate with the best linear SVM, we see that the error is still higher than the previous models.

SVM Regression: Polynomial Kernel

```
svm2 <- svm(cnt~., data=train, kernel="polynomial", cost=1, scale=TRUE)
summary(svm2)
```

```
##
## Call:
## svm(formula = cnt ~ ., data = train, kernel = "polynomial", cost = 1,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  3
##       gamma:  0.125
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  8998
```

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

```
## [1] "Correlation:  0.557117842720389"
```

```
print(paste("MSE: ", mse_svm2))
```

```
## [1] "MSE:  22258.0513669117"
```

We see that the resulting error is the lowest so far.

Tune to find the best polynomial C

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

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     5
##
## - best performance: 24962.78
##
## - Detailed performance results:
##    cost    error dispersion
## 1 1e-03 31744.38   2309.091
## 2 1e-02 28549.00   1892.866
## 3 1e-01 26783.96   1723.961
## 4 1e+00 25288.00   1554.467
## 5 5e+00 24962.78   1555.267
## 6 1e+01 24962.80   1561.702
## 7 1e+02 24973.94   1523.342
```

From the results, we see that the error values are relatively similar for the cost values of 5, 10, 100.

Evaluate with best polynomial SVM

```
pred <- predict(tune_svm2$best.model, newdata=test)
cor_svm2_tune <- cor(pred, test$cnt)
mse_svm2_tune <- mean((pred - test$cnt)^2)
print(paste("Correlation: ", cor_svm2_tune))
```

```
## [1] "Correlation:  0.557507075885472"
```

```
print(paste("MSE: ", mse_svm2_tune))
```

```
## [1] "MSE:  22409.4875762153"
```

SVM Regression: Radial Kernel

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

```
##
## Call:
## svm(formula = cnt ~ ., data = train, 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:  8992
```

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

```
## [1] "Correlation:  0.546355901645455"
```

```
print(paste("MSE: ", mse_svm3))
```

```
## [1] "MSE:  22584.4474644196"
```

Tune hyperparameters for radial kernel

```
set.seed(1234)
tune.out <- tune(svm, cnt~., data=validate, kernel="radial",
                 ranges=list(cost=c(0.1,1,10,100),
                             gamma=c(1,2,3,4)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##   0.1     1
##
## - best performance: 25681.36
##
## - Detailed performance results:
##     cost gamma    error dispersion
## 1    0.1     1 25681.36   3382.174
## 2    1.0     1 25876.98   3227.232
## 3   10.0     1 27451.90   3392.582
## 4  100.0     1 32436.78   3838.433
## 5    0.1     2 26525.14   3450.427
## 6    1.0     2 26322.11   3172.261
## 7   10.0     2 29480.14   2618.255
## 8  100.0     2 39494.04   4303.345
## 9    0.1     3 27198.53   3461.740
```

```
## 10   1.0     3 26620.51   3030.747
## 11  10.0     3 31184.62   2718.668
## 12 100.0     3 45334.25   5465.335
## 13   0.1     4 27777.10   3552.060
## 14   1.0     4 26738.41   2954.792
## 15  10.0     4 33118.51   3084.506
## 16 100.0     4 47785.12   6355.815
```

```
tune_svm_3 <- svm(cnt~., data=train, kernel="radial", cost=0.1, gamma=1, scale=TRUE)
summary(tune_svm_3)
```

```
##
## Call:
## svm(formula = cnt ~ ., data = train, kernel = "radial", cost = 0.1,
##     gamma = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  0.1
##       gamma:  1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  9021
```

```
pred <- predict(tune_svm_3, newdata=test)
cor_svm3_tune <- cor(pred, test$cnt)
mse_svm3_tune <- mean((pred - test$cnt)^2)
print(paste("Correlation: ", cor_svm3_tune))
```

```
## [1] "Correlation:  0.55140750035338"
```

```
print(paste("MSE: ", mse_svm3_tune))
```

```
## [1] "MSE:  22478.2034942236"
```

Analysis

The original linear regression model had a mean squared error of 22307. Out of the different kernel methods, the polynomial kernel had the best error of 22195, followed by the radial kernel with an error of 22478, and finally the linear kernel with an error of 23318. Only the polynomial kernel beat the original linear model.

Both the original linear regression model and the linear kernel had similar results, which makes sense. The fact that the polynomial kernel had the best results suggests that the data may be more polynomial in shape, and the most general radial kernel is only slightly worse.