Multiple Regression Model

Guan-Yuan Wang

2020/9/5

Exploring different multiple regression models for house price prediction

```
dataTesting <- fread("kc_house_test_data.csv")
dataTraining <- fread("kc_house_train_data.csv")
```

```
dataTesting <- dataTesting %>%
  mutate(bedrooms_squared = bedrooms * bedrooms,
         bed_bath_rooms = bedrooms * bathrooms,
         log_sqft_living = log(sqft_living),
         lat_plus_long = lat + long)

dataTraining <- dataTraining  %>%
  mutate(bedrooms_squared = bedrooms * bedrooms,
         bed_bath_rooms = bedrooms * bathrooms,
         log_sqft_living = log(sqft_living),
         lat_plus_long = lat + long)


mean(dataTesting$bedrooms_squared)
```

```
## [1] 12.44668
```

```
mean(dataTesting$bed_bath_rooms)
```

```
## [1] 7.503902
```

```
mean(dataTesting$log_sqft_living)
```

```
## [1] 7.550275
```

```
mean(dataTesting$lat_plus_long)
```

```
## [1] -74.65333
```

```r
m1 <- lm(price ~ sqft_living + bedrooms + bathrooms +
           lat + long, dataTraining)

m2 <- lm(price ~ sqft_living + bedrooms + bathrooms +
           lat + long + bed_bath_rooms, dataTraining)

m3 <- lm(price ~ sqft_living + bedrooms + bathrooms +
           lat + long + bed_bath_rooms + bedrooms_squared +
           log_sqft_living + lat_plus_long, dataTraining)

summary(m1)
```

```
##
## Call:
## lm(formula = price ~ sqft_living + bedrooms + bathrooms + lat +
##     long, data = dataTraining)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1714877  -120222   -16106    84299  4037317
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.908e+07  1.647e+06 -41.940  < 2e-16 ***
## sqft_living  3.123e+02  3.183e+00  98.097  < 2e-16 ***
## bedrooms    -5.959e+04  2.483e+03 -23.999  < 2e-16 ***
## bathrooms    1.571e+04  3.587e+03   4.379  1.2e-05 ***
## lat          6.586e+05  1.310e+04  50.286  < 2e-16 ***
## long        -3.094e+05  1.326e+04 -23.331  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 236000 on 17378 degrees of freedom
## Multiple R-squared:  0.5926, Adjusted R-squared:  0.5925
## F-statistic:  5056 on 5 and 17378 DF,  p-value: < 2.2e-16
```

```r
summary(m2)
```

```
##
## Call:
## lm(formula = price ~ sqft_living + bedrooms + bathrooms + lat +
##     long + bed_bath_rooms, data = dataTraining)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2185337  -117976   -15853    82369  3994569
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -6.687e+07  1.648e+06 -40.584   <2e-16 ***
## sqft_living     3.066e+02  3.197e+00  95.909   <2e-16 ***
## bedrooms       -1.134e+05  4.798e+03 -23.646   <2e-16 ***
```

```
## bathrooms      -7.146e+04  7.553e+03  -9.462   <2e-16 ***
## lat             6.548e+05  1.304e+04  50.230   <2e-16 ***
## long           -2.943e+05  1.325e+04 -22.218   <2e-16 ***
## bed_bath_rooms  2.558e+04  1.953e+03  13.097   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 234800 on 17377 degrees of freedom
## Multiple R-squared:  0.5966, Adjusted R-squared:  0.5964
## F-statistic:  4283 on 6 and 17377 DF,  p-value: < 2.2e-16
```

```r
summary(m3)
```

```
##
## Call:
## lm(formula = price ~ sqft_living + bedrooms + bathrooms + lat +
##     long + bed_bath_rooms + bedrooms_squared + log_sqft_living +
##     lat_plus_long, data = dataTraining)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3085947  -110695   -17188    76060  3183077
##
## Coefficients: (1 not defined because of singularities)
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -6.204e+07  1.613e+06 -38.449  < 2e-16 ***
## sqft_living       5.294e+02  7.691e+00  68.836  < 2e-16 ***
## bedrooms          3.451e+04  9.386e+03   3.677 0.000236 ***
## bathrooms         6.706e+04  1.078e+04   6.219 5.13e-10 ***
## lat               6.614e+05  1.268e+04  52.169  < 2e-16 ***
## long             -2.794e+05  1.289e+04 -21.676  < 2e-16 ***
## bed_bath_rooms   -8.571e+03  2.856e+03  -3.001 0.002696 **
## bedrooms_squared -6.789e+03  1.493e+03  -4.546 5.51e-06 ***
## log_sqft_living  -5.618e+05  1.755e+04 -32.014  < 2e-16 ***
## lat_plus_long            NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 228000 on 17375 degrees of freedom
## Multiple R-squared:  0.6197, Adjusted R-squared:  0.6196
## F-statistic:  3539 on 8 and 17375 DF,  p-value: < 2.2e-16
```

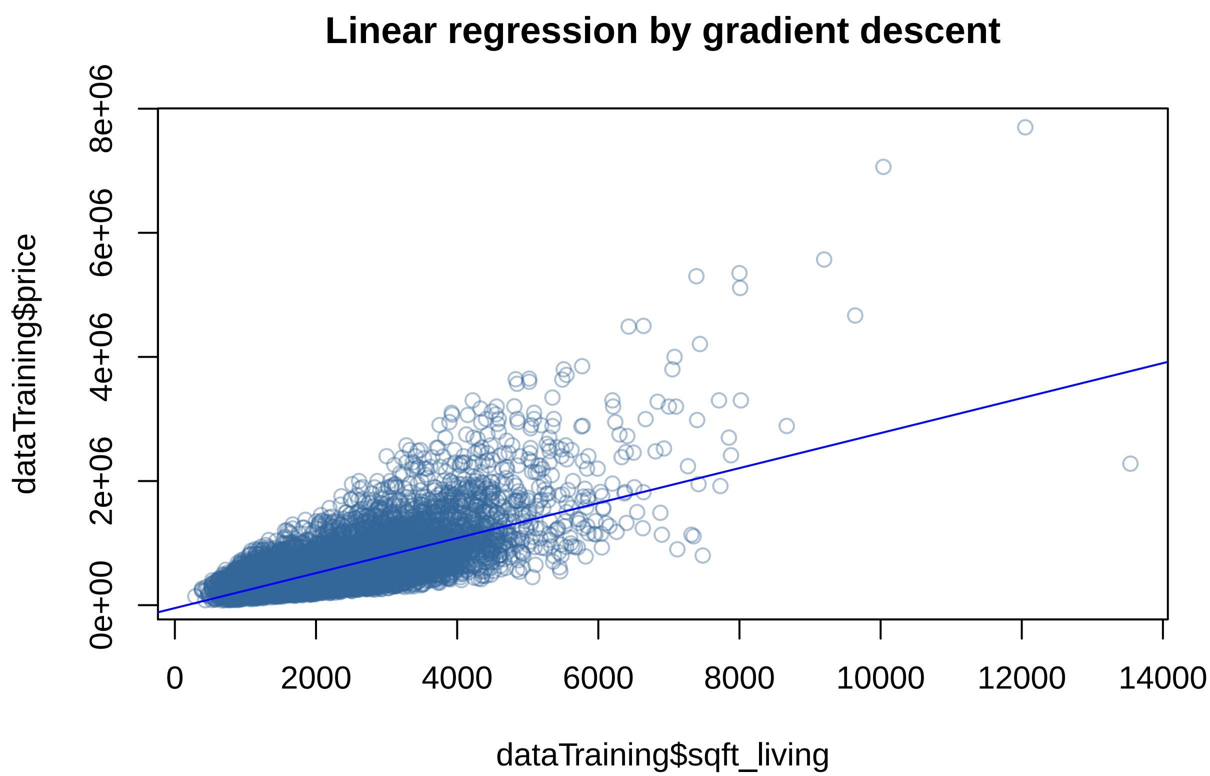
Implementing gradient descent for multiple regression

```r
simpleModel <- lm(price ~ sqft_living, dataTraining)
simpleModel
```

```
##
## Call:
## lm(formula = price ~ sqft_living, data = dataTraining)
```

```
##
## Coefficients:
## (Intercept)  sqft_living
##      -47116          282
```

```r
plot(dataTraining$sqft_living,
     dataTraining$price,
     col = rgb(0.2, 0.4, 0.6, 0.4),
     main = 'Linear regression by gradient descent')
abline(simpleModel, col = 'blue')
```




```r
# learning rate and tolerance
alpha <- 7 * 10 ^(-12)
tolerance <-  2.5 * 10^7

# initialize coefficients
beta <- matrix(c(-47000, 1), nrow = 2)

# add a column of 1's for the intercept coefficient
feature <- cbind(1, matrix(dataTraining$sqft_living))
price <- matrix(dataTraining$price)

# gradient descent
converged <-  F
derivative <- c()
```

```r
while(converged == F){
  # predict outcome
  predictions <- feature %*% beta
  # feature derivative
  errors <- predictions - dataTraining$price
  # initialize the gradient sum of squares
  gradient_sum_squares <- 0

  # while we haven't reached the tolerance yet, update each feature's weight
  for (i in 1:ncol(feature)){
    derivative[i] = 2 * (t(errors) %*% matrix(feature[,i]))
  }

  gradient_sum_squares <- (derivative * derivative) + gradient_sum_squares

  beta <- beta - matrix(alpha * derivative)

  gradient_magnitude = sqrt(gradient_sum_squares)

  if
  ( gradient_magnitude[1] < tolerance & gradient_magnitude[2] < tolerance) {
    converged = T
  }
}

print(beta)
```

```
##              [,1]
## [1,] -46999.8872
## [2,]    281.9121
```

```r
predPrice <- cbind(1, matrix(dataTesting$sqft_living)) %*% beta

predPrice[1]
```

```
## [1] 356134.4
```

```r
dataTesting[1, "price"]
```

```
## [1] 310000
```

```r
# Calculate RSS
residuals <- predPrice - dataTesting[, "price"]
RSS <- sum(residuals * residuals)
```

```r
multipleModel <- lm(price ~ sqft_living + sqft_living15, dataTraining)
multipleModel
```

```
##
## Call:
```

```
## lm(formula = price ~ sqft_living + sqft_living15, data = dataTraining)
##
## Coefficients:
##   (Intercept)    sqft_living  sqft_living15
##    -100262.18         245.19          65.27
```

```
# learning rate and tolerance
alpha <- 4 * 10 ^(-12)
tolerance <-  1 * 10^9

# initialize coefficients
beta <- matrix(c(-100000, 1, 1), nrow = 3)

# add a column of 1's for the intercept coefficient
feature <- cbind(1, matrix(dataTraining$sqft_living),
                 matrix(dataTraining$sqft_living15))
price <- matrix(dataTraining$price)

# gradient descent
converged <-  F
derivative <- c()

while(converged == F){
  # predict outcome
  predictions <- feature %*% beta
  # feature derivative
  errors <- predictions - dataTraining$price
  # initialize the gradient sum of squares
  gradient_sum_squares <- 0

  # while we haven't reached the tolerance yet, update each feature's weight
  for (i in 1:ncol(feature)){
    derivative[i] = 2 * (t(errors) %*% matrix(feature[,i]))
  }

  gradient_sum_squares <- (derivative * derivative) + gradient_sum_squares

  beta <- beta - matrix(alpha * derivative)

  gradient_magnitude = sqrt(gradient_sum_squares)

  if
  ( gradient_magnitude[1] < tolerance & gradient_magnitude[2] < tolerance &
    gradient_magnitude[3] < tolerance) {
    converged = T
  }
}

print(beta)
```

```
##              [,1]
## [1,] -99999.96879
## [2,]    245.03546
## [3,]     65.31986
```

```r
predPrice <- cbind(1, matrix(dataTesting$sqft_living),
                   matrix(dataTesting$sqft_living15)) %*% beta

predPrice[1]
```

```
## [1] 366670.1
```

```r
dataTesting[1, "price"]
```

```
## [1] 310000
```

```r
residuals <- predPrice - dataTesting[, "price"]
RSS.multiple <- sum(residuals * residuals)
RSS.multiple
```

```
## [1] 2.702617e+14
```

```r
RSS.multiple < RSS
```

```
## [1] TRUE
```