HW0

Amit Arora

September 14, 2018

Problem 1 Regularization

We read the hitters dataset into a dataframe ad then apply Lasso and ridge regression to predict salary based on other parameters in the dataset. Note that all character (in this case same as categorical) fields are removed from the dataset prior to modelling.

The code to read the dataset and a function to run the glm model with shrinkage is presented below. Same function is used for ridge and lasso, with the alpha parameter set to 0 for ridge and 1 for lasso.

```
run_regularization_model <- function(x, y, train, alpha) {
  # check if this is ridge or lasso
  if(alpha == 0) {
    reg_type = "Ridge"
  } else if(alpha == 1) {
    reg_type = "Lasso"
  } else {
    cat(sprintf("unsupported value for alpha %d, exiting\n", alpha))
    return;
  }

  # set the grid for a cross validation grid search for best lambda
  grid <- 10^seq(10,-2, length =100)

  # regression without CV
  mod <- suppressWarnings(glmnet(x[train ,],y[train],alpha =alpha, lambda =grid ,
                          thresh =1e-12))

  # cross validation to find best lambda
  cv.out <- suppressWarnings(cv.glmnet(x[train ,], y[train], alpha = alpha))
  # plot the cv output
  suppressWarnings(plot(cv.out))

  # lambda min is the best lambda
  bestlam <- cv.out$lambda.min
  cat(sprintf("the best lambda for %s regression is %f\n", reg_type, bestlam))

  # how to coefficients change with lambda
  suppressWarnings(plot(cv.out$glmnet.fit, xvar = c("lambda"), label = TRUE))

  # test mse
  pred <- suppressWarnings(predict(mod, s=bestlam, newx=x[-train,]))
  mse <- suppressWarnings(mean((pred -y[-train])^2))
  cat(sprintf("test mse with %s regression is %f\n", reg_type, mse))

  reg_coeff <- suppressWarnings(predict(mod, type ="coefficients", s=bestlam))
  cat(sprintf("coefficients for %s regression are as follows\n", reg_type))
  print(reg_coeff)
}
```

```r
# get the data
hitters <- read_csv("https://gist.githubusercontent.com/keeganhines/59974f1ebef97bbaa44fb19143f90bad/ra
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_integer(),
##   X1 = col_character(),
##   League = col_character(),
##   Division = col_character(),
##   Salary = col_double(),
##   NewLeague = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```r
# here is what it looks like
glimpse(hitters)
```

```
## Observations: 322
## Variables: 21
## $ X1        <chr> "-Andy Allanson", "-Alan Ashby", "-Alvin Davis", "-A...
## $ AtBat     <int> 293, 315, 479, 496, 321, 594, 185, 298, 323, 401, 57...
## $ Hits      <int> 66, 81, 130, 141, 87, 169, 37, 73, 81, 92, 159, 53, ...
## $ HmRun     <int> 1, 7, 18, 20, 10, 4, 1, 0, 6, 17, 21, 4, 13, 0, 7, 3...
## $ Runs      <int> 30, 24, 66, 65, 39, 74, 23, 24, 26, 49, 107, 31, 48,...
## $ RBI       <int> 29, 38, 72, 78, 42, 51, 8, 24, 32, 66, 75, 26, 61, 1...
## $ Walks     <int> 14, 39, 76, 37, 30, 35, 21, 7, 8, 65, 59, 27, 47, 22...
## $ Years     <int> 1, 14, 3, 11, 2, 11, 2, 3, 2, 13, 10, 9, 4, 6, 13, 3...
## $ CAtBat    <int> 293, 3449, 1624, 5628, 396, 4408, 214, 509, 341, 520...
## $ CHits     <int> 66, 835, 457, 1575, 101, 1133, 42, 108, 86, 1332, 13...
## $ CHmRun    <int> 1, 69, 63, 225, 12, 19, 1, 0, 6, 253, 90, 15, 41, 4,...
## $ CRuns     <int> 30, 321, 224, 828, 48, 501, 30, 41, 32, 784, 702, 19...
## $ CRBI      <int> 29, 414, 266, 838, 46, 336, 9, 37, 34, 890, 504, 186...
## $ CWalks    <int> 14, 375, 263, 354, 33, 194, 24, 12, 8, 866, 488, 161...
## $ League    <chr> "A", "N", "A", "N", "N", "A", "N", "A", "N", "A", "A...
## $ Division  <chr> "E", "W", "W", "E", "E", "W", "E", "W", "W", "E", "E...
## $ PutOuts   <int> 446, 632, 880, 200, 805, 282, 76, 121, 143, 0, 238, ...
## $ Assists   <int> 33, 43, 82, 11, 40, 421, 127, 283, 290, 0, 445, 45, ...
## $ Errors    <int> 20, 10, 14, 3, 4, 25, 7, 9, 19, 0, 22, 11, 7, 6, 8, ...
## $ Salary    <dbl> NA, 475.000, 480.000, 500.000, 91.500, 750.000, 70.0...
## $ NewLeague <chr> "A", "N", "A", "N", "N", "A", "A", "A", "N", "A", "A...
```

```r
# remove all char fields
hitters <- hitters[, !sapply(hitters, is.character)]

# remove any NAs
hitters <- hitters %>%
  drop_na()

# train. test split
set.seed(1)
train <- sample(nrow(hitters), 0.5*nrow(hitters))
```

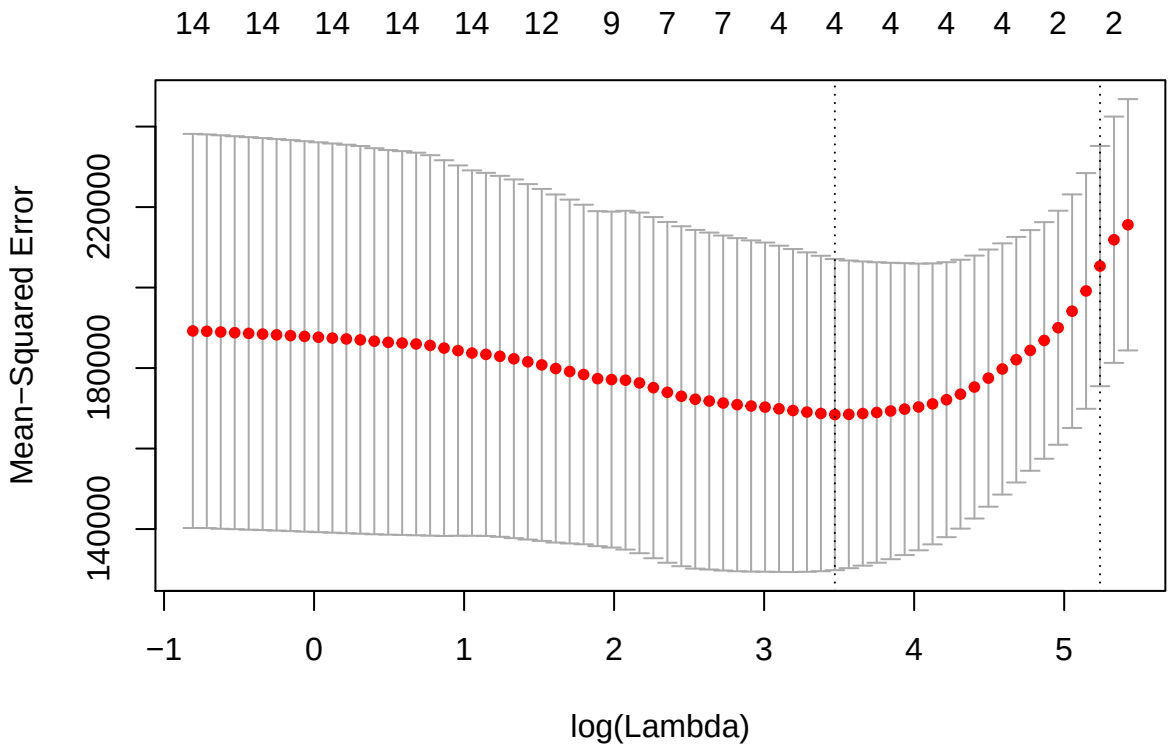
```
# model matrix
x <- model.matrix (Salary ~ ., hitters)[,-1]
y <- hitters$Salary
```

1.1 Lasso

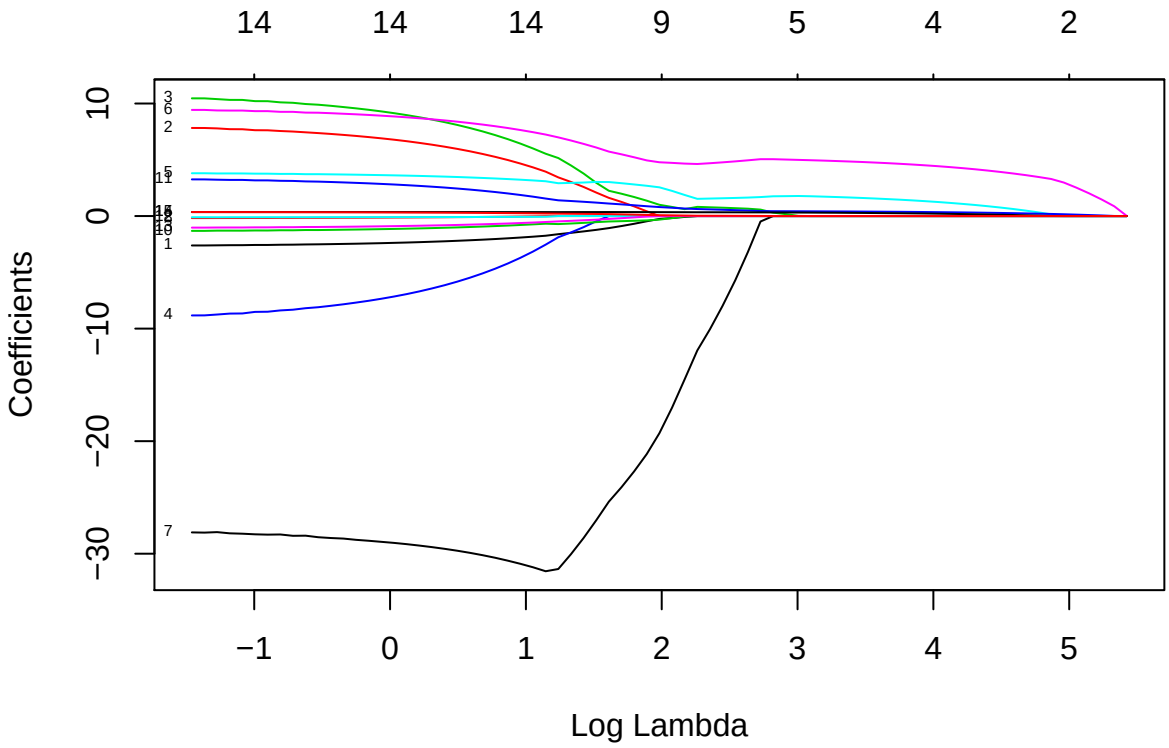
1. Visualization of the coefficient trajectories with lambda is presented below.
2. A 50% train/test split is used. Depending upon the split size and the initial seed the coefficient values may vary slightly.
3. For the best value of lambda there are 4 predictors left. These predictors are RBI, Walks, CRuns and PutOuts. The values of the coefficients for the best value of lambda (which is 32.18) is provided in the output of the code below. The coefficient of PutOuts is smallest amongst the predictors and is quite close to 0 (it is 0.27) so we could also consider that only 3 predictors (RBI, Walks, CRuns) are really left.
4. The best value of lambda is found using cross-validation on a grid search for lambda from 0.01 to 10^{10}

```
# lasso
run_regularization_model(x, y, train, 1)
```

```
## Warning in seq.default(10, -2, length = 100): partial argument match of
## 'length' to 'length.out'
```




```
## the best lambda for Lasso regression is 32.182838
```



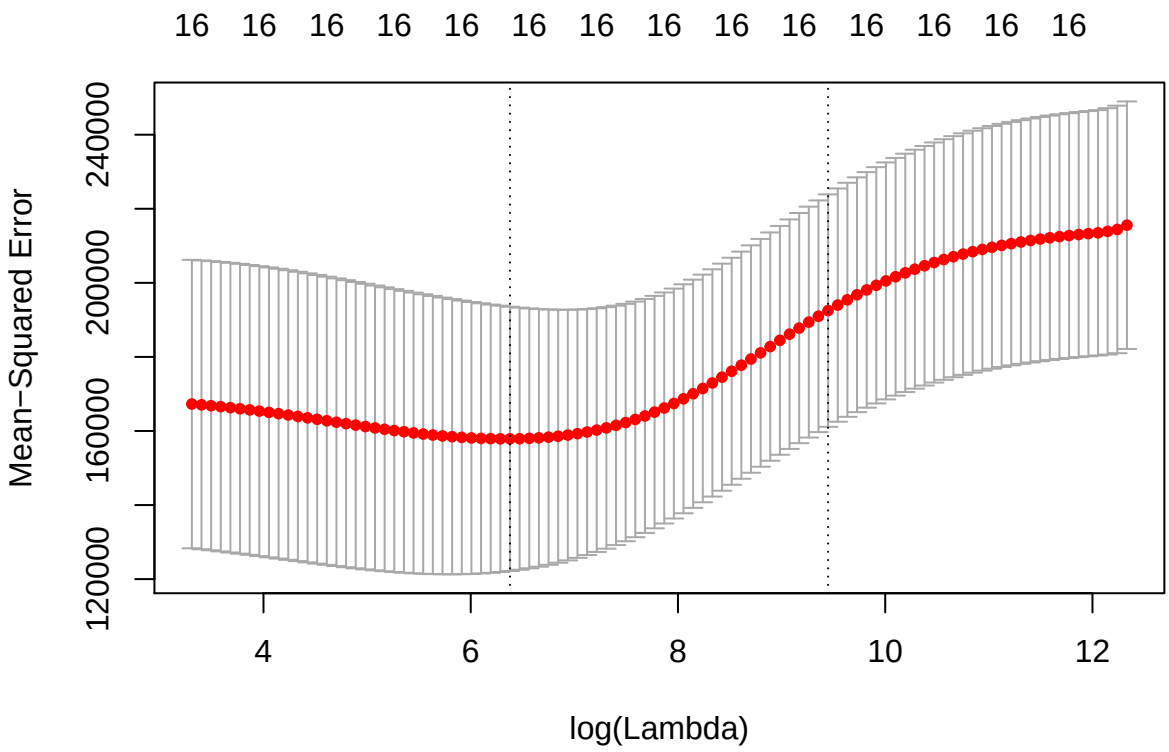

```
## test mse with Lasso regression is 105997.834444
## coefficients for Lasso regression are as follows
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 58.5220797
## AtBat        .
## Hits         .
## HmRun        .
## Runs         .
## RBI          1.6093134
## Walks        4.8055125
## Years        .
## CAtBat       .
## CHits        .
## CHmRun       .
## CRuns        0.3992952
## CRBI         .
## CWalks       .
## PutOuts      0.2784412
## Assists      .
## Errors       .
```

1.2 Ridge

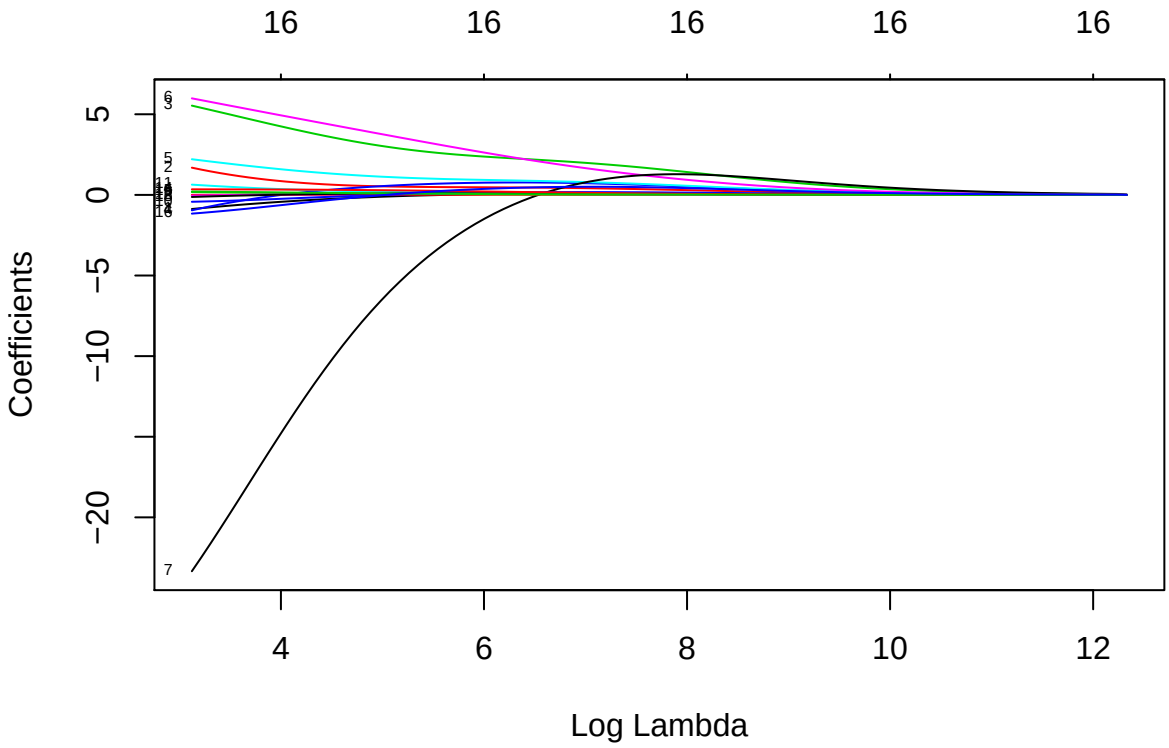
The results for the Ridge regression are presented below. As expected, the coefficients do not go down to 0. The test MSE for best lambda for ridge and lasso is very similar, although the error with ridge being a little bit less than the error with lasso.

```
# ridge
run_regularization_model(x, y, train, 0)
```

```
## Warning in seq.default(10, -2, length = 100): partial argument match of
## 'length' to 'length.out'
```




```
## the best lambda for Ridge regression is 589.183541
```

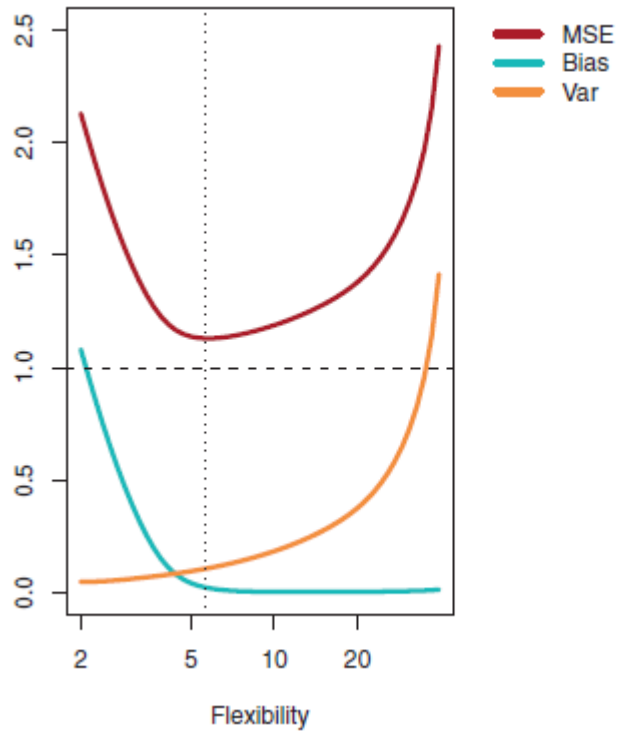



```
## test mse with Ridge regression is 102025.496430
## coefficients for Ridge regression are as follows
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) 74.432544645
## AtBat        0.071311110
## Hits         0.457778387
## HmRun        2.237272706
## Runs         0.769550718
## RBI          0.884901390
## Walks        2.244889534
## Years       -0.402018583
## CAtBat       0.008833718
## CHits        0.041989467
## CHmRun       0.159023878
## CRuns        0.101232703
## CRBI         0.058000556
## CWalks       0.093594118
## PutOuts      0.180564714
## Assists      0.044096688
## Errors       0.445423003
```

2. Bias-Variance tradeoff

Bias variance tradeoff means that as the variance in the parameters of a model increases the prediction error on training data decreases and vice versa. In practical terms it means as the model becomes more and more flexible it gets the ability to fit the training data better but that also results in reducing the models ability to fit the new (unseen) data. Thus while the bias or training error can be reduced by increasing the flexibility (variance) but there is a tradeoff to be made to determine at what point we stop increasing the variance to reduce the error and decide that the corresponding amount of error is acceptable.

We also use the terms underfitting i.e. low bias and overfitting i.e. high variance in this context. The following diagram shows the bias variance tradeoff in terms of bias, variance and test MSE.



The bias-variance tradeoff is illustrated in both ridge and lasso regression as can be seen in the plots between the bias (training MSE) and the shrinkage parameter λ. As lambda increases, the penalty on the coefficients increase i.e. variance decreases and the model becomes less and less flexible and therefore the bias increases. Thus the MSE curve starts to move upwards as we move from left to right on the x-axis (λ space) and the sweet spot is where the bias just begins its increase.