Linear Regression

Jim Harner

1/6/2020

sparklyr requires a dplyr compatible back-end to Spark.

```
library(dplyr, warn.conflicts = FALSE)

# load the sparklyr package
library(sparklyr)
# start the sparklyr session
master <- "local"
# master <- "spark://master:7077"
sc <- spark_connect(master)
```

6.3 Concrete Slump Test Regression

Load slump.csv into Spark with spark_read_csv from the local filesystem.

```
slump_sdf <- spark_read_csv(sc, "slump_sdf",
                path =  "file:///home/rstudio/rspark-tutorial/data/slump.csv")
head(slump_sdf)
```

```
## # Source: spark<?> [?? x 10]
##   cement  slag fly_ash water    sp coarse_aggr fine_aggr slump  flow
##    <dbl> <dbl>   <dbl> <dbl> <dbl>       <dbl>     <dbl> <dbl> <dbl>
## 1    273    82     105   210     9         904       680    23  62
## 2    163   149     191   180    12         843       746     0  20
## 3    162   148     191   179    16         840       743     1  20
## 4    162   148     190   179    19         838       741     3  21.5
## 5    154   112     144   220    10         923       658    20  64
## 6    147    89     115   202     9         860       829    23  55
## # ... with 1 more variable: compressive_strength <dbl>
```

First we need to split slump_sdf into a training and a test Spark DataFrame.

```
slump_partition <- tbl(sc, "slump_sdf") %>%
  sdf_partition(training = 0.7, test = 0.3, seed = 2)
slump_train_sdf <- slump_partition$training
slump_test_sdf <- slump_partition$test
```

The full model is now run.

```
slump_lr_full_fit <- slump_partition$training %>%
  ml_linear_regression(compressive_strength ~ cement + slag + fly_ash + water
                       + sp + coarse_aggr + fine_aggr)
summary(slump_lr_full_fit)
```

```
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -5.6280 -1.6192 -0.3183  0.9372  7.1920
##
## Coefficients:
```

```
##  (Intercept)       cement         slag      fly_ash        water           sp
## 219.36232986   0.03777496  -0.06065688   0.02819246  -0.31892157  -0.12983604
##  coarse_aggr    fine_aggr
##  -0.08744781  -0.06805072
##
## R-Squared: 0.8987
## Root Mean Squared Error: 2.507
```

Notice that the model summary does not provide much useful information. We can p-values by by getting a `tidy` summary.

```r
tidy(slump_lr_full_fit)
```

```
## # A tibble: 8 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept) 219.       92.4        2.37  0.0207
## 2 cement        0.0378    0.0290     1.30  0.198
## 3 slag         -0.0607    0.0409    -1.48  0.143
## 4 fly_ash       0.0282    0.0301     0.938 0.352
## 5 water        -0.319     0.0927    -3.44  0.00104
## 6 sp           -0.130     0.183     -0.708 0.481
## 7 coarse_aggr  -0.0874    0.0355    -2.46  0.0166
## 8 fine_aggr    -0.0681    0.0379    -1.79  0.0778
```

Performance metrics for regression are generally obtained first be getting predictions and then using an evaluator to get a specific metric.

```r
slump_lr_full_predict <- ml_predict(slump_lr_full_fit)
slump_lr_full_predict
```

```
## # Source: spark<?> [?? x 11]
##    cement  slag fly_ash water    sp coarse_aggr fine_aggr slump  flow
##     <dbl> <dbl>   <dbl> <dbl> <dbl>       <dbl>     <dbl> <dbl> <dbl>
##  1   137  167       214   226    6          708       757   27.5  70
##  2   140    1.4     198.  175.   4.4       1050.      780.  16.2  31
##  3   140  128       164   183   12          871       775   23.8  53
##  4   140  128       164   237    6          869       656   24    65
##  5   140.   4.2     216.  194.   4.7       1050.      710.  24.5  57
##  6   140.  11.8     226.  208.   4.9       1021.      684.  21    64
##  7   140.  44.8     235.  171.   5.5       1048.      704   23.5  52.5
##  8   140.  61.1     239.  182.   5.7       1018.      681.  24.5  60
##  9   142  130       167   174   11          883       785    0    20
## 10   143  131       168   217    6          891       672   25    69
## # ... with more rows, and 2 more variables: compressive_strength <dbl>,
## #   prediction <dbl>
```

```r
ml_regression_evaluator(slump_lr_full_predict, label_col = "compressive_strength",
                        prediction_col = "prediction", metric_name = "rmse")
```

```
## [1] 2.50723
```

This would be awkward if want to evaluate a series of models for several metrics.

The model for the lasso with varying values of the regularization parameter λ.

```r
slump_perf_metrics <- function(l) {
  slump_train_sdf %>%
```

```
    ml_linear_regression(compressive_strength ~ cement + slag + fly_ash +
                         water + sp + coarse_aggr + fine_aggr,
                         elastic_net_param = 1, reg_param = l)
}
```

First, we Initialize the performance data frames for $\lambda = 0$. Notice that we can get the performance metrics as the components of summary list, which in turn if an element of the fitted list.

```
regParm <- c(0.02, 0.04, 0.06, 0.08, 0.1, 0.12, 0.14)
slump_lr_errors <- data.frame(lambda = 0,
                        r2 = slump_lr_full_fit$summary$r2,
                        rmse = slump_lr_full_fit$summary$root_mean_squared_error,
                        mae = slump_lr_full_fit$summary$mean_absolute_error)
slump_lr_coef <- as.data.frame(slump_lr_full_fit$coefficients)
```

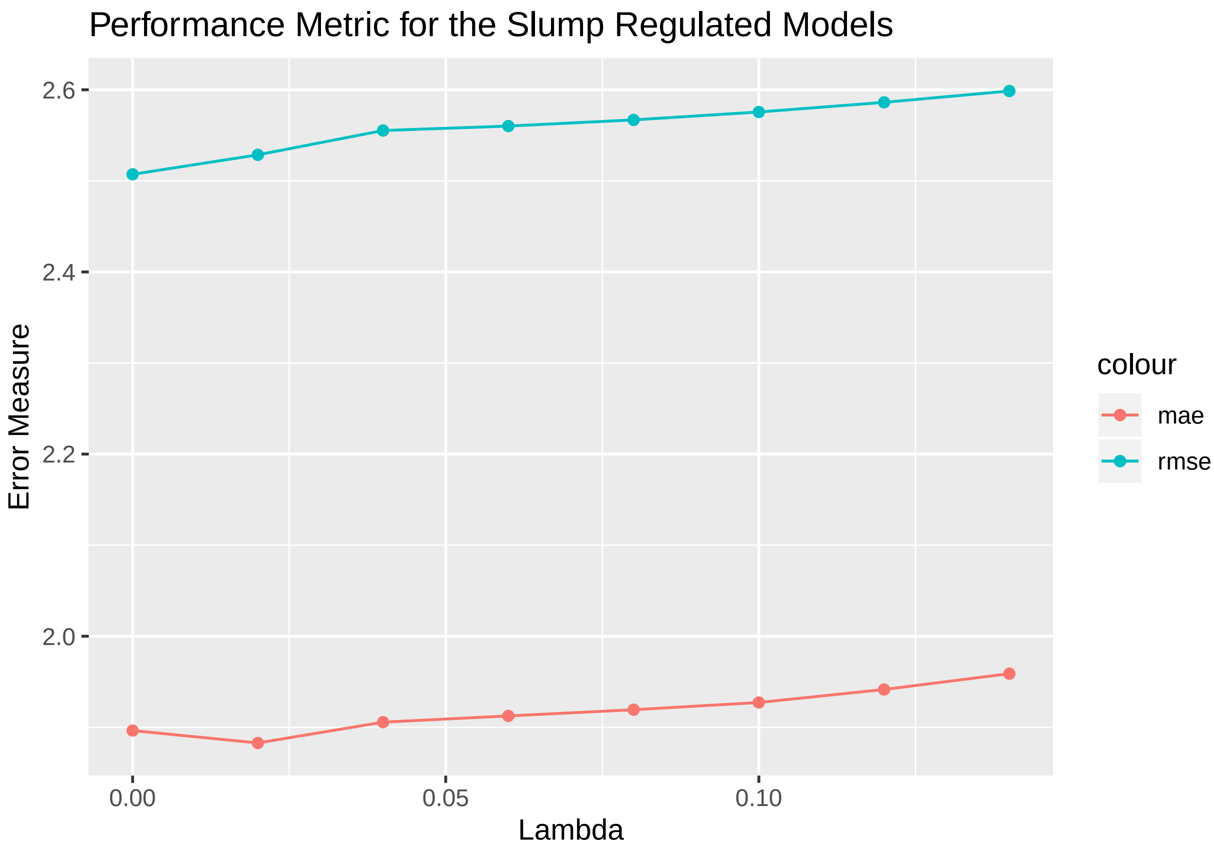
We now calculate `r2`, `rmse`, and `mae` for each of the models.

```
for(l in regParm) {
  slump_lr_fit <- slump_perf_metrics(l)
  slump_lr_errors <-
    data.frame(lambda = l,
               r2 = slump_lr_fit$summary$r2,
               rmse = slump_lr_fit$summary$root_mean_squared_error,
               mae = slump_lr_fit$summary$mean_absolute_error) %>%
    rbind(slump_lr_errors, .)
  slump_lr_coef <-
    as.data.frame(slump_lr_fit$coefficients) %>%
    cbind(slump_lr_coef, .)
}
slump_lr_errors
```

```
##   lambda        r2     rmse      mae
## 1   0.00 0.8987442 2.507230 1.896407
## 2   0.02 0.8970112 2.528594 1.882784
## 3   0.04 0.8948250 2.555291 1.905668
## 4   0.06 0.8944239 2.560160 1.912491
## 5   0.08 0.8938638 2.566941 1.919315
## 6   0.10 0.8931438 2.575633 1.927251
## 7   0.12 0.8922638 2.586218 1.941522
## 8   0.14 0.8912237 2.598671 1.958873
```

Finally, we plot the performance measures.

```
library(ggplot2)
slump_lr_errors %>%
  ggplot(aes(x = lambda)) +
  geom_point(aes(y = rmse, color = 'rmse')) +
  geom_line(aes(y = rmse, color = 'rmse')) +
  geom_point(aes(y = mae, color = 'mae')) +
  geom_line(aes(y = mae, color = 'mae')) +
  ggtitle("Performance Metric for the Slump Regulated Models") +
  xlab("Lambda") + ylab("Error Measure")
```



Based on the performance metrics, it is clear we want `lambda` to be small. However, we also want parsimony.

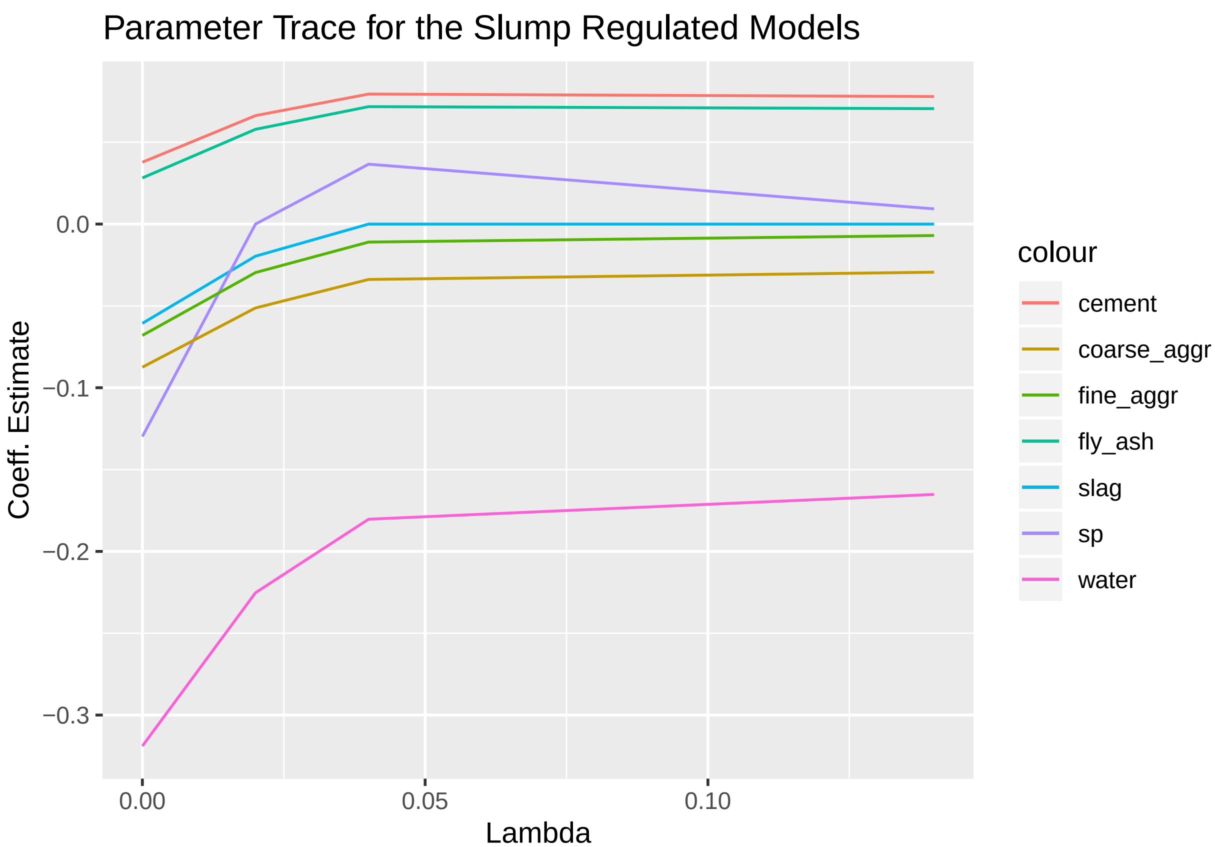
We now get the parameter estimates as `lambda` increases.

```
names(slump_lr_coef) <- as.character(rbind(c(0.0, regParm)))
slump_lr_coef <- t(slump_lr_coef)
slump_lr_coef
```

```
##      (Intercept)     cement        slag    fly_ash      water           sp
## 0      219.36233 0.03777496 -0.06065688 0.02819246 -0.3189216 -0.129836041
## 0.02   125.22235 0.06625311 -0.01963970 0.05789226 -0.2252704  0.000000000
## 0.04    80.23832 0.07943291  0.00000000 0.07174224 -0.1803432  0.036599089
## 0.06    78.41981 0.07912863  0.00000000 0.07149383 -0.1772967  0.031142392
## 0.08    76.60486 0.07882439  0.00000000 0.07125072 -0.1742739  0.025677415
## 0.1     74.78988 0.07852014  0.00000000 0.07100761 -0.1712511  0.020212383
## 0.12    72.97495 0.07821590  0.00000000 0.07076451 -0.1682284  0.014747354
## 0.14    71.15997 0.07791167  0.00000000 0.07052140 -0.1652056  0.009282288
##      coarse_aggr   fine_aggr
## 0    -0.08744781 -0.068050721
## 0.02 -0.05126774 -0.029635158
## 0.04 -0.03384715 -0.010993751
## 0.06 -0.03296010 -0.010196193
## 0.08 -0.03207201 -0.009399424
## 0.1  -0.03118391 -0.008602646
## 0.12 -0.03029584 -0.007805887
## 0.14 -0.02940773 -0.007009110
```

The lasso trace of the coefficient estimates provides a way of picking the strength of regulation.

```r
library(ggplot2)
as.data.frame(cbind(lambda = c(0.0, regParm), slump_lr_coef)) %>%
  ggplot(aes(x = lambda)) +
  geom_line(aes(y = cement, color = 'cement')) +
  geom_line(aes(y = slag, color = 'slag')) +
  geom_line(aes(y = fly_ash, color = 'fly_ash')) +
  geom_line(aes(y = water, color = 'water')) +
  geom_line(aes(y = sp, color = 'sp')) +
  geom_line(aes(y = coarse_aggr, color = 'coarse_aggr')) +
  geom_line(aes(y = fine_aggr, color = 'fine_aggr')) +
  ggtitle("Parameter Trace for the Slump Regulated Models") +
  xlab("Lambda") + ylab("Coeff. Estimate")
```



Over the range of λ, we have 3 features (`cement`, `fly_ash`, and `water`) with consistently non-zero coefficient estimates. Arguably, `coarse_aggr` also deviates from 0. These agree with the model we found by *ad hoc* variable selection in Section 6.1.

At this point we could pick several models to run on the test Spark DataFrame for final selection.

```r
spark_disconnect(sc)
```