# Linear Regression

Jim Harner

1/12/2021

`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 <- "spark://master:7077"
master <- "local"
sc <- spark_connect(master = 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_random_split(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
## -5.7501 -1.6642 -0.2428  1.2498  6.9504
##
## Coefficients:
##  (Intercept)       cement         slag      fly_ash        water           sp
## 117.20416259   0.07075548  -0.01835116   0.05690188  -0.21973660  -0.04664274
##  coarse_aggr    fine_aggr
##  -0.04619533  -0.02701631
##
## R-Squared: 0.9074
## Root Mean Squared Error: 2.545
```

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) 117.       82.0         1.43  0.158
## 2 cement        0.0708    0.0265      2.67  0.00946
## 3 slag         -0.0184    0.0368     -0.498 0.620
## 4 fly_ash       0.0569    0.0266      2.14  0.0358
## 5 water        -0.220     0.0829     -2.65  0.00998
## 6 sp           -0.0466    0.180      -0.259 0.796
## 7 coarse_aggr  -0.0462    0.0318     -1.45  0.151
## 8 fine_aggr    -0.0270    0.0331     -0.815 0.418
```

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.   11.8    226.  208.   4.9       1021.      684.  21    64
##  6   140.   30.5    239   169.   5.3       1028.      743.  21.2  46
##  7   140.   44.8    235.  171.   5.5       1048.      704   23.5  52.5
##  8   141.    0.6    210.  189.   4.6        996.      789.  23.5  53
##  9   142   130      167   174   11          883       785    0    20
## 10   142   130      167   215    6          735       836   25.5  67
## # ... 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] 2.544609
```

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 $\lambda$.

```
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)
```

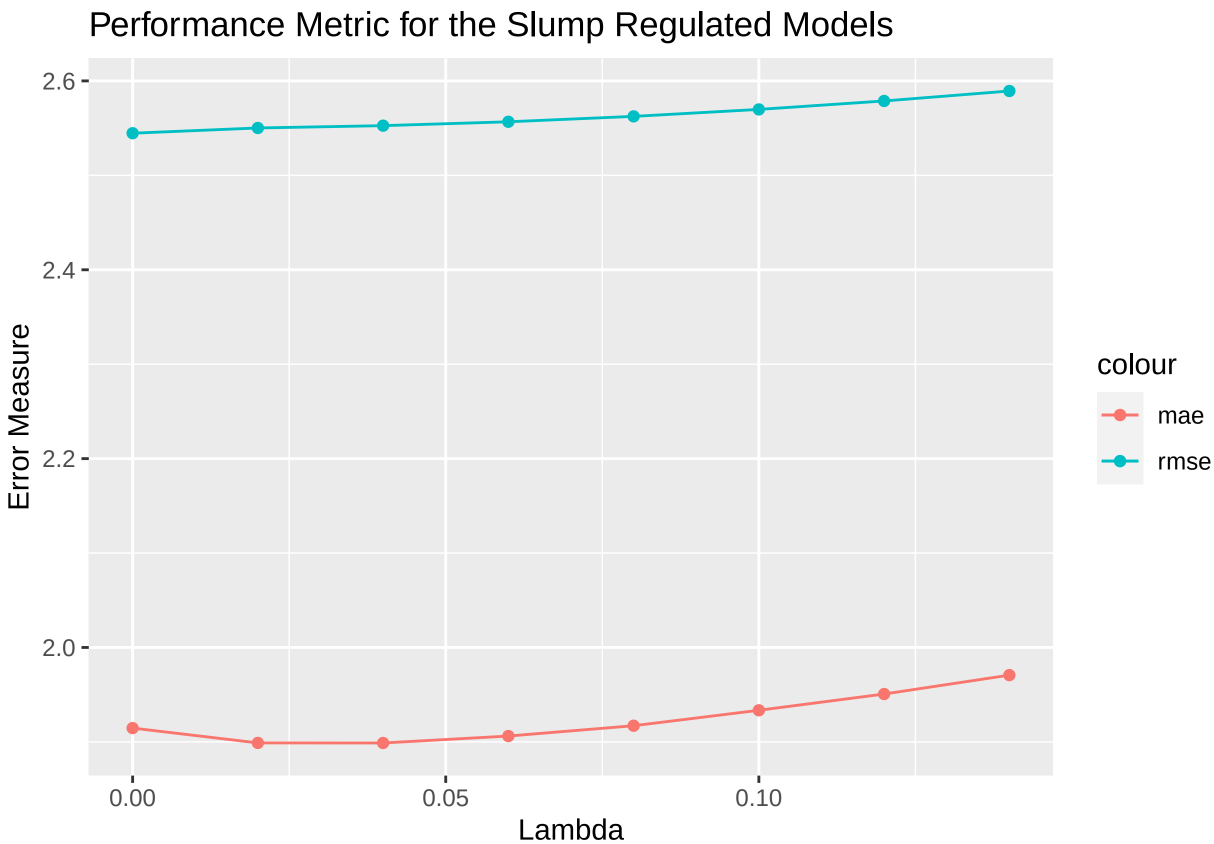
We now calculate `r2`, `rmse`, and `mae` for each of the models.

```
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 0.9073766 2.544609 1.914607
## 2   0.02 0.9069733 2.550142 1.898923
## 3   0.04 0.9067934 2.552607 1.898834
## 4   0.06 0.9064935 2.556710 1.906198
## 5   0.08 0.9060736 2.562444 1.917134
## 6   0.10 0.9055338 2.569797 1.933465
## 7   0.12 0.9048740 2.578755 1.950691
## 8   0.14 0.9040943 2.589302 1.970697
```

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, e.g., about 0.025. However, we also want parsimony.

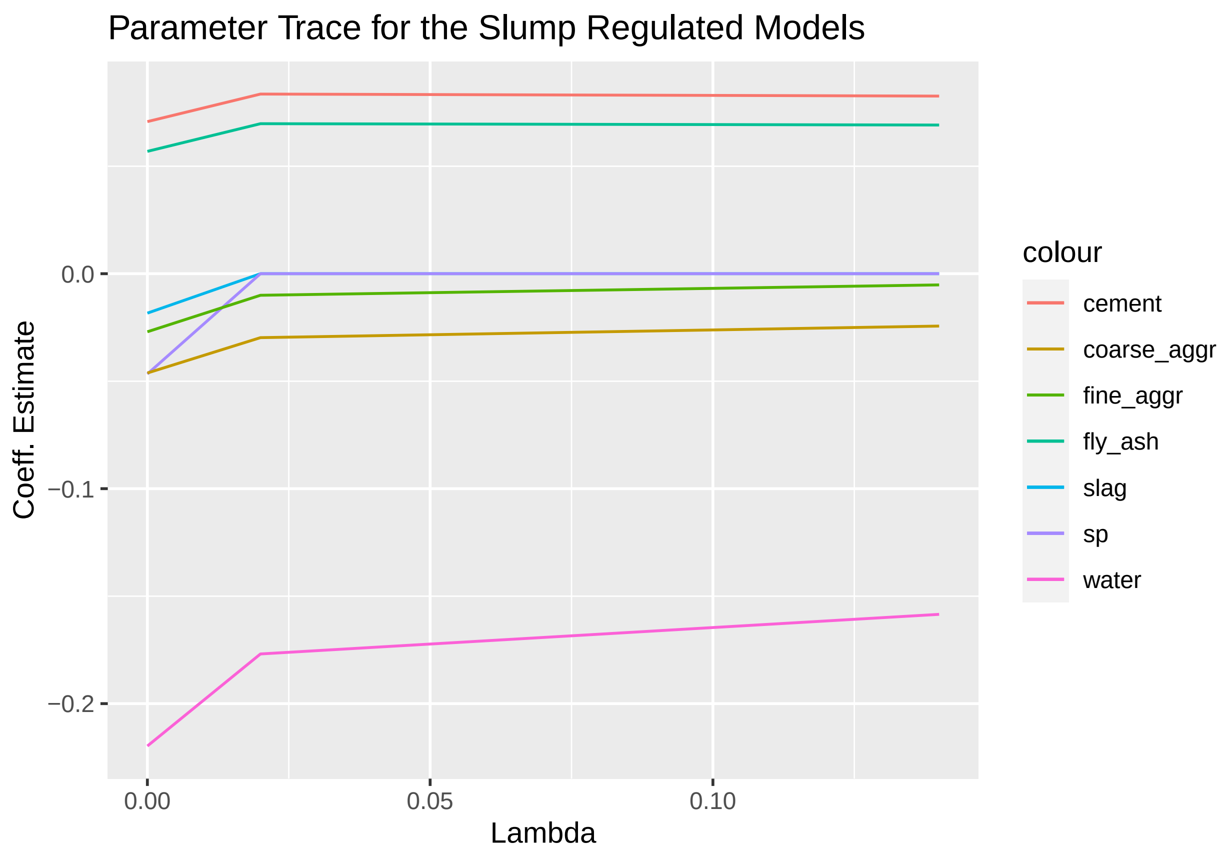
We now get the parameter estimates as `lambda` increases.

```
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      117.20416 0.07075548 -0.01835116 0.05690188 -0.2197366 -0.04664274
## 0.02    74.95717 0.08357135  0.00000000 0.06978957 -0.1768884  0.00000000
## 0.04    73.01203 0.08341408  0.00000000 0.06968541 -0.1738196  0.00000000
## 0.06    71.06689 0.08325680  0.00000000 0.06958124 -0.1707508  0.00000000
## 0.08    69.12175 0.08309952  0.00000000 0.06947707 -0.1676820  0.00000000
## 0.1     67.17661 0.08294224  0.00000000 0.06937291 -0.1646132  0.00000000
## 0.12    65.23147 0.08278496  0.00000000 0.06926875 -0.1615443  0.00000000
## 0.14    63.28631 0.08262769  0.00000000 0.06916458 -0.1584755  0.00000000
##      coarse_aggr    fine_aggr
## 0    -0.04619533 -0.027016311
## 0.02 -0.02975010 -0.010036086
## 0.04 -0.02885223 -0.009231771
## 0.06 -0.02795436 -0.008427455
## 0.08 -0.02705649 -0.007623130
## 0.1  -0.02615862 -0.006818821
## 0.12 -0.02526075 -0.006014506
## 0.14 -0.02436287 -0.005210183
```

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 $\lambda$, 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)
```