Final Project

Evan Ott

December 5, 2016

Stochastic Gradient Descent (Logistic Regression)

Let's load in the data and Dr. Scott's algorithm:

```
library(Matrix)
library(Rcpp)
library(RcppEigen)

Rcpp::sourceCpp("scott_sgdlogit.cpp")

raw_data = read.csv("master.csv", header = TRUE)

# Read specific columns from the data

kegg = as.character(raw_data[ , 1])
gene = as.character(raw_data[ , 2])

data = raw_data[ , 3:ncol(raw_data)]

N = ncol(data)
P = nrow(data)
M = rep(1, N) # Used in logistic regression

# For C++ code, it's easiest to use a sparse matrix
# TODO: might want to consider scaling the counts, either in the way DESeq2 does
# it or something similar.
X = Matrix(as.matrix(unname(data)), sparse = TRUE)

# Convert the column names to 0 = control, 1 = treatment
Ynames = names(data)
Y = stringi::stri_endswith(Ynames, fixed = "T") * 1
```

Now, let's run the code and see what happens.

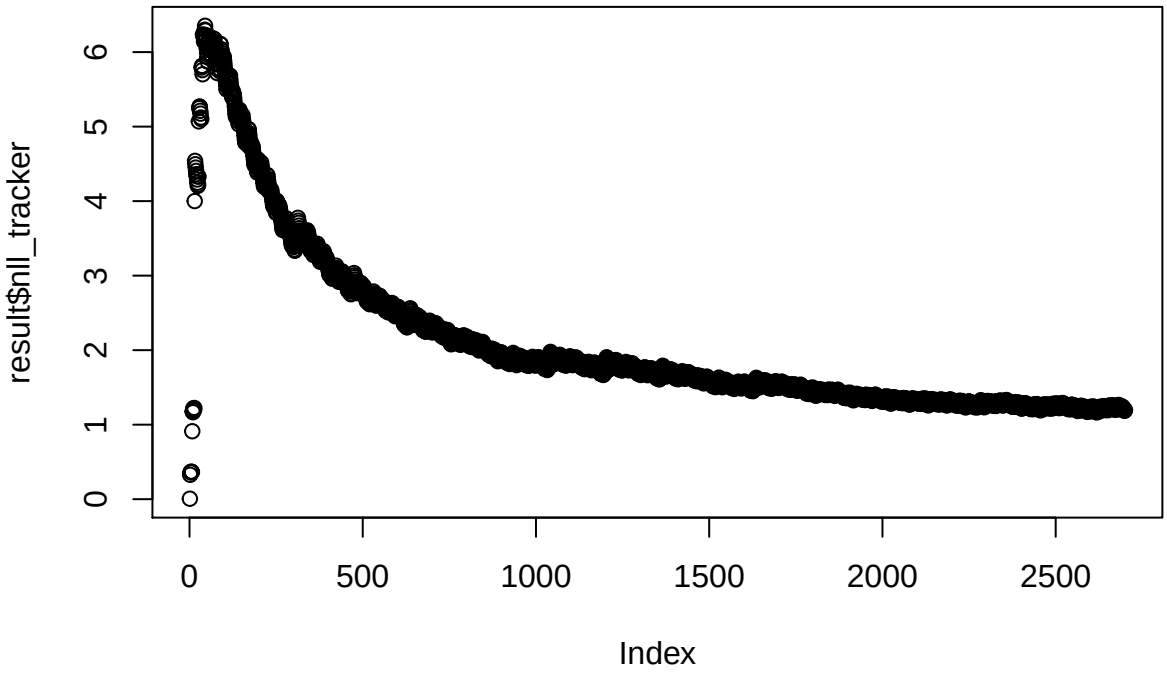
```
# Initial guess for the betas.
beta_init = rep(0, P)
# For the weighting in the skip scenario for a feature
# TODO: algorithm is very sensitive to this value
eta = 0.0002
# passes through the data
nIter = 100
# L1 regularization
# TODO: need cross-validation or other way to choose lambda.
# Right now, using manual value by eye (BAD practice)
lambda = 1200.0
# exponentially-weighted moving average factor
# TODO: could explore other values.
```

```r
discount = 0.01


# X[i, j] == data[i, j] is non-negative, so this trims out the genes with no counts
X_nozero = X[which(rowSums(data) > 0), ]
# Run the algorithm
result = sparsesgd_logit(X_nozero, Y, M, eta, nIter, beta_init, lambda, discount)
```

```
## How many genes are in the final model?  53
## Which genes?
## adef_c_5_1802
## aot172_c_2_112
## aot178_c_15_553
## aot178_c_57_1246
## aot180_c_13_1642
## axyl_c_1_5125
## bdim_c_51_1620
## belk_c_262_3655
## belk_c_623_6138
## blon2249_c_1_1767
## cot380_c_63_1416
## cot876_c_4_661
## ecas2549_c_2_485
## ecas2549_c_2_487
## ecas2549_c_2_492
## ecas2549_c_3_766
## efae1080_c_1_429
## efae1080_c_1_1699
## ehor_c_1_111
## ehor_c_9_1965
## ghae2070_c_18_1519
## ghae2070_c_18_1525
## ghae2070_c_25_1713
## ghae2070_c_26_1723
## ghae2070_c_33_1864
## gmor_c_6_936
## gmor_c_8_1123
## gmor_c_11_1354
## gsan_c_8_657
## kora_c_3_2066
## ksed_c_1_1105
## lfer_c_1_1848
## lfus_c_9_2360
## lfus_c_18_3454
## lvag_c_31_1238
## lwad_c_2_278
## mfer1263_c_1_1129
## mhom2524_c_3_645
## mneo_c_1_1955
## mneo_c_1_4619
## NCBIABWK_c_4_1154
## NCBIACFE_c_1_366
## NCBIACFU_c_47_2923
## not020_c_81_1706
```

```
## nsic_c_7_1124
## pfus_c_18_2013
## pmir_c_10_1585
## pot302_c_4_620
## psac_c_66_1997
## psal_c_109_2279
## psco_c_34_2002
## ptan_c_1_266
## rpic_c_1_2351
```



DESeq2

Now, let's use a pre-fabricated solution instead. I've included the code here to do the parallel execution, but I think RMarkdown will actually only run in one thread because reasons. Maybe.

```
library(DESeq2)
#library(BiocParallel)
# Allow for parallelization of DESeq2 code.
#register(MulticoreParam(4))

# Get the counts
count_data_raw = raw_data[ , 3:ncol(raw_data)]

########################################
#  Format the data in the way DESeq2 expects
########################################
new_colnames = rep(NULL, N)
condition = rep(NULL, N)
control = 0
treatment = 0
for (i in 1:N) {
```

```
  base = c("control", "treatment")[Y[i] + 1]
  val = 0
  if (base == "control") {
    control = control + 1
    val = control
  } else {
    treatment = treatment + 1
    val = treatment
  }
  new_colnames[i] = paste0(base, val)
  condition[i] = base
}
count_data = count_data_raw
colnames(count_data) = new_colnames
rownames(count_data) = gene

col_data = data.frame(condition)
rownames(col_data) = new_colnames

# Run DESeq2
dds = DESeqDataSetFromMatrix(countData = count_data,
                             colData = col_data,
                             design = ~ condition)
dds = DESeq(dds)#, parallel = TRUE)
res = results(dds)#, parallel = TRUE)

# plot(sort(res$pvalue))
# points(sort(res$padj), col = "red")
# P - sum(is.na(res$pvalue))
# P - sum(is.na(res$padj))
```

```
## How many genes in default DESeq2?
## abau_c_1_3356
## aisr_c_20_1998
## amas2385_c_4_872
## aot170_c_17_1854
## aot172_c_99_1663
## aot448_c_6_964
## cdip_c_1_1509
## chom_c_27_1386
## chom_c_55_2086
## cper_c_2_296
## cure_c_1_1578
## dpig_c_1_8
## esak_c_1_3901
## fnucp_c_3_1442
## fper2555_c_2_652
## gmor_c_19_1592
## lcat_c_18_1871
## lgas_c_1_25
## lot107_c_9_1346
## mlot_c_1_5349
## mneo_c_1_1226
## mneo_c_1_1611
```

```
## mot186_c_3_1980
## mtub_c_1_788
## nbac_c_3_267
## NCBIABIX_c_1_1199
## pend_c_1_176
## peno_c_75_2748
## pmel_c_20_1104
## pmuls_c_6_923
## pot786_c_28_1621
## pple_c_7_1120
## pstu_c_1_2331
## pver_c_10_1377
## raer_c_1_313
## raer_c_17_1833
## raer_c_5_1016
## raer_c_9_1389
## rden1994_c_1_356
## sked_c_1_1767
## smal_c_1_1545
## smit_c_1_1759
## smut_c_1_1468
## smut_c_1_1794
## smut_c_1_371
## smut_c_1_836
## sot138_c_21_1472
## sot149_c_17_1639
## ssal_c_18_1887
## tmed_c_12_2247
```

```
# Not exactly sure what this plot does either.
# plotMA(res, main="DESeq2", ylim=c(-3,3))

# Benjamini-Hochberg by hand (DESeq2 does something similar...)
alpha = 0.1
sorted_pval = sort(res$pvalue, na.last = TRUE)
numNA = sum(is.na(res$pvalue))
bh = sapply(1:P, function(i){ sorted_pval[i] <= alpha * (i + 1) / (P - numNA)})
threshold = sorted_pval[max(which(bh))]
```

```
## How many genes in customized Benjamini-Hochberg procedure
## cdip_c_1_1509
## dpig_c_1_8
## raer_c_9_1389
## smut_c_1_1468
```

```
# These two are equal -- throws out genes that were never observed section 1.5.3 of DESeq2 paper
print(paste(sum(rowSums(count_data) == 0),
            sum(is.na(res$pvalue))))
```

```
## [1] "31025 31025"
```

```
# 111675 genes are thrown out by the "independent filtering" for having a low
# mean normalized count
sum(is.na(res$padj)) - sum(is.na(res$pvalue))
```

```
## [1] 111675
```

```r
# Not entirely sure what this plot means.
# plot(metadata(res)$filterNumRej,
#      type = "b", ylab = "number of rejections",
#      xlab = "quantiles of filter")
# lines(metadata(res)$lo.fit, col = "red")
# abline(v = metadata(res)$filterTheta)

#nofilter
# If you turn off the filtering, only two of the adjusted p-values are less than 0.1
# where the adjustment basically just accounts for Benjamini-Hochberg, I think
resNoFilt <- results(dds, independentFiltering = FALSE)
addmargins(table(filtering = (res$padj < .1),
                 noFiltering = (resNoFilt$padj < .1)))
```

```
##          noFiltering
## filtering FALSE  TRUE   Sum
##     FALSE 40723     0 40723
##     TRUE     48     2    50
##     Sum   40771     2 40773
```

```
## How many genes without independent filtering
## dpig_c_1_8
## raer_c_9_1389
```