STAT 5525 Data Analytics Final Project Report

Joe Conwell, Vinayak Siva Kumar, Zihao Cai

Abstract

In our final project, we analyze the high-dimensional data-set *Scheetz 2006* by three different modern sparse regression methods: Lasso Regression, Horseshoe Regression, and Adaptive Elastic Net. We then compare the models and contrast the performances of those methods for this dataset, as well as explore their variable selection capabilities.

Dataset description

The dataset *Scheetz 2006* is about gene expression and regulation in the mammalian eye of laboratory rats. The data-set contains 18,975 probes out of the total 31,000 different probes that were detected as valid expressions in the table. And there two elements in this dataset are:

- y: Gene expression measurement for Trim32
- X: Gene expression measurements for remaining genes

(Data Reference: https://myweb.uiowa.edu/pbreheny/data/Scheetz2006.html)

Trim32 is known to be linked with a genetic disorder called Bardet-Biedl Syndrome (BBS): the mutation (P130S) in Trim32 gives rise to BBS. For this reason, Trim32 is also sometimes called Bbs11.

The dimensions of *Scheetz 2006*: n = 120, p = 18,975. It is obvious that the number of features is larger than the number of observations in this dataset, therefore traditional linear regression methods are not effective to analyze this dataset because of the curse of dimensionality. Therefore, by using Sparse Regression methods, it is possible to shrink the number of features to get more effective and relevant elements to generate better predictions in the module.

Sparse Regression

First, import the *Scheetz2006.rds* data-set, and split it into two sets: train set and test set.

```
remotes::install_github("pbreheny/hdrm")
library(hdrm)
downloadData(Scheetz2006)
attachData(Scheetz2006)
n = nrow(X)

train_rows <- sample(1:n, n/2)
X.train <- X[train_rows, ]
X.test <- X[-train_rows, ]

y.train <- y[train_rows]
y.test <- y[-train_rows]

dim(X.train)
dim(X.test)
```

Then checking the dimension of the train set and test train:

```
> dim(X.train)
[1]    60 18975
> dim(X.test)
[1]    60 18975
```

1. Lasso

Code:

```
install.packages("glmnet")
library(glmnet)

lasso.mod = glmnet(x.train, y.train, family="gaussian",nlambda = 100, alpha = 1)
plot(lasso.mod)
```

Output:

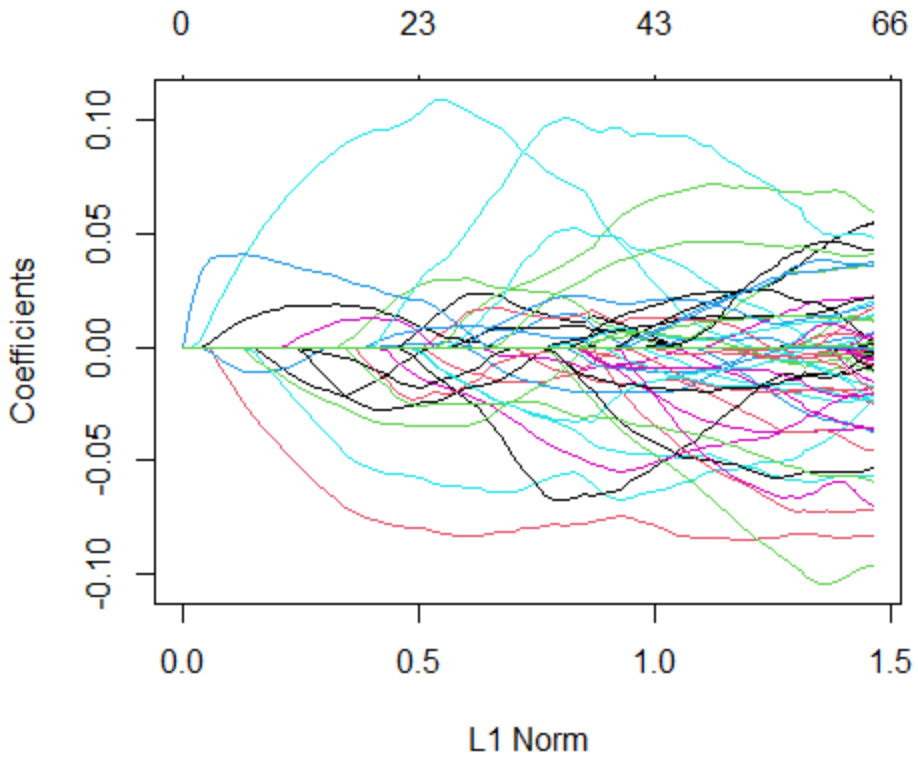

Figure 1: L1 Normalization of Lasso coefficients

Applying the glment to build up a lasso model with the train sets. Set *nlambda* as 100 which means generating 100 lambda values to model fits. The graph shows how those coefficients would affect the predicting and the problem is that the number of coefficients is too large.

Then calculate the best lambda for next step:

```
lasso.pred = predict(lasso.mod, s = bestlam, newx = x.test)
mean((lasso.pred-y.test)^2)

lasso.coef=predict(lasso.mod, type ="coefficients",s=bestlam)
length(lasso.coef)
lasso.coef[lasso.coef !=0]
```

The best lambda for Lasso is 0.05164217. Using the Lasso model with this lambda, we predict the test set to check the performance and accuracy of Lasso in this situation.

Code:

```
lasso.pred = predict(lasso.mod, s = bestlam, newx = x.test)
mean((lasso.pred-y.test)^2)

lasso.coef=predict(lasso.mod, type ="coefficients",s=bestlam)
lasso.coef[lasso.coef !=0]
```

```
> lasso.pred = predict(lasso.mod, s = bestlam, newx = x.test)
> mean((lasso.pred-y.test)^2)
[1] 0.01950226
```

First, the MSE of the Lasso model in Scheetz.rds is 0.01950226. The Lasso model is well behaved in the *Scheetz.rds* .

```
> length(lasso.coef)
[1] 18976
> lasso.coef[lasso.coef !=0]

 [1]  7.9370580933 -0.0143627335  0.0715704462 -0.0252740336  0.0040737411  0.0180275808 -0.0526027732
 [8]  0.0345184571 -0.0072234088 -0.0009163894 -0.0177375057 -0.0003142105
```

Next, check the status of coefficients in the Lasso model. There are a total 18976 coefficients affecting the predicting. With penalties and shrinkage by Lasso, we could know there are 12 of the 18976 coefficient estimates that are non-zero and have valid attributions on predicting in Lasso mode.

2. Horseshoe

Another method that utilizes Bayesian shrinkage is the *horseshoe* method. The horseshoe method is an apt method for handling sparse regression. We will show the horseshoe prior for the *Scheetz dataset*. The horseshoe prior is better than lasso for model selection when the model is sparse, as this case creates a space where model selection is useful (Carvalho 2009).

Code:

```
library(horseshoe)


horseshoe.fit <- horseshoe(y=y.train, X=x.train,
method.tau="truncatedCauchy", method.sigma="Jeffreys")

plot(y, X%*%horseshoe.fit$BetaHat)
```

```
[1] 1000
[1] 2000
[1] 3000
[1] 4000
[1] 5000
[1] 6000
```

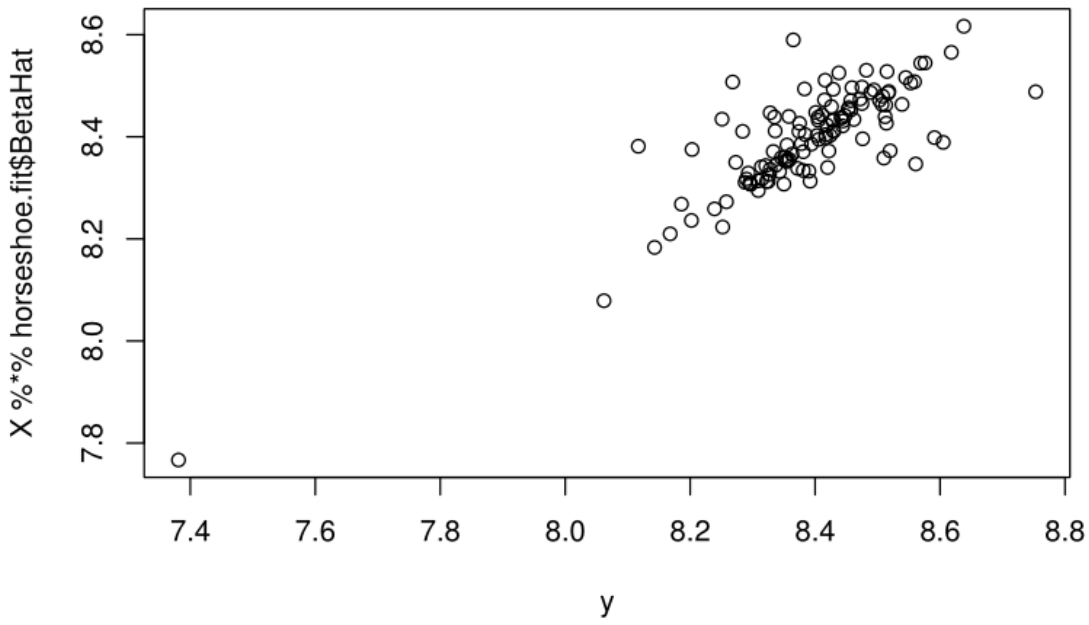

Figure 2: Predictive values against the Scheetz 2006 data

```
horseshoe.fit$TauHat #posterior mean of tau
```

```
## [1] 0.0002896575
```

Now we look for credible variables and we find that we have eight potential ones to work with, with their estimated coefficients equal to -8.481e-5.

```
library(horseshoe)

horseshoe.betas <- HS.var.select(horseshoe.fit, y.train,
method="interval")
sum(horseshoe.betas) #variables
horseshoe.fit$BetaHat[horseshoe.betas] #coeffs
```

```
[1] 8
[1] -8.481615e-05 -8.481615e-05 -8.481615e-05 -8.481615e-05
[5] -8.481615e-05 -8.481615e-05 -8.481615e-05 -8.481615e-05
```

Now we plot the credible intervals:

```
library(Hmisc)
xYplot(Cbind(horseshoe.fit$BetaHat, horseshoe.fit$LeftCI,
horseshoe.fit$RightCI) ~ 1:30)
```

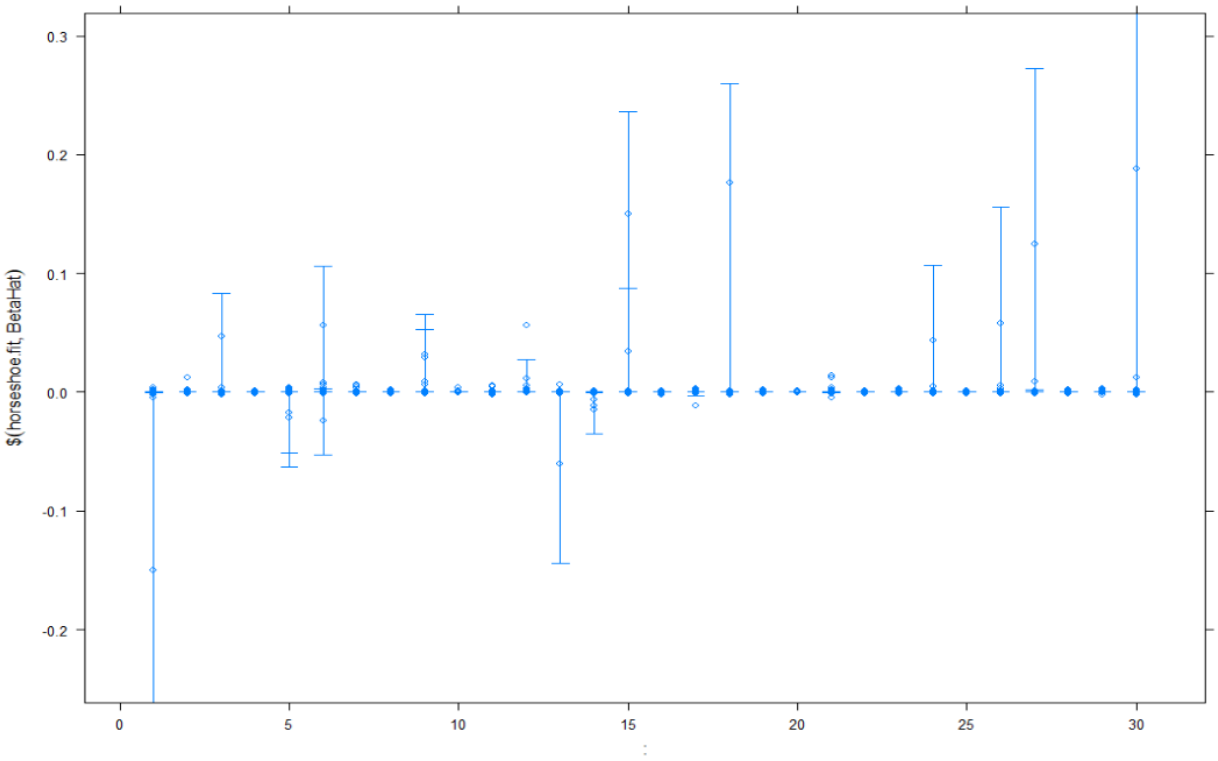

Figure 3: Credible Intervals

```
horseshoe.pred <- x.test %*% horseshoe.fit$BetaHat
(MSE = mean((y.test - horseshoe.pred)^2))
[1] 0.01674154
```

With a mean square error of: 0.01674154 we know that the horseshoe method is well behaved for the dataset.

3. Adaptive Elastic Net

The adaptive elastic net is a combination of the “adaptive lasso” model and the elastic net model. Adaptive lasso has an additional “oracle procedure” to the regular lasso model which estimates the subset of true parameters with zero coefficients (Zou H 2009).

Instead of the regular lasso model which forces all coefficients to be equally penalized, the adaptive lasso model assigns different weights to the coefficients (also called as data-driven weights). The weights enable the adaptive lasso model to apply different quantities of shrinkage to various coefficients. Therefore, it is allowed to severely penalize coefficients more with small values compared to the original elastic net model.

Hence the term “oracle” as it has some information on the true subset model beforehand. The Adaptive Elastic net combines the weighted l1 penalty of the adaptive lasso and the elastic net regularization.

Code:

```
#install.packages("gcdnet")
library(gcdnet)
adapt_en_model = gcdnet(X.train, y.train, method ="ls")
plot(adapt_en_model)
```

Output:

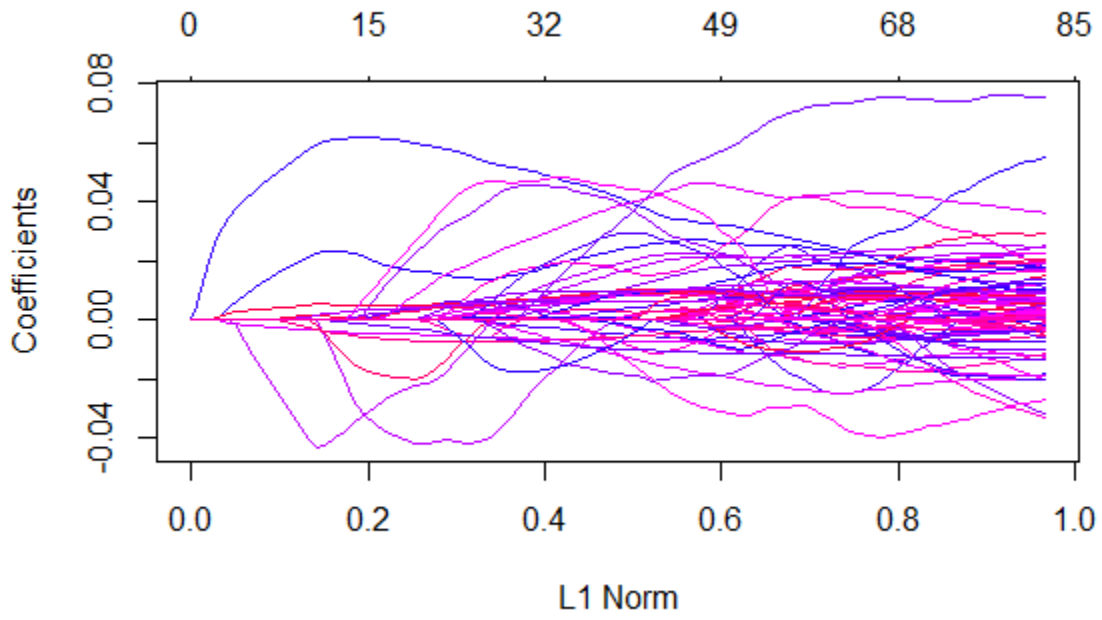

Figure 4: L1 Normalization of Adaptive ElasticNet Coefficients

Similarly to lasso, we can see that the number of coefficients the current model uses is pretty large.

We can use cross-validation to choose the best lambda value.

Code:

```
set.seed(1)
#Cross validation to find the best lambda
cv.out = cv.gcdnet(X.train, y.train, method ="ls",pred.loss="loss")
plot(cv.out)
```

Output:

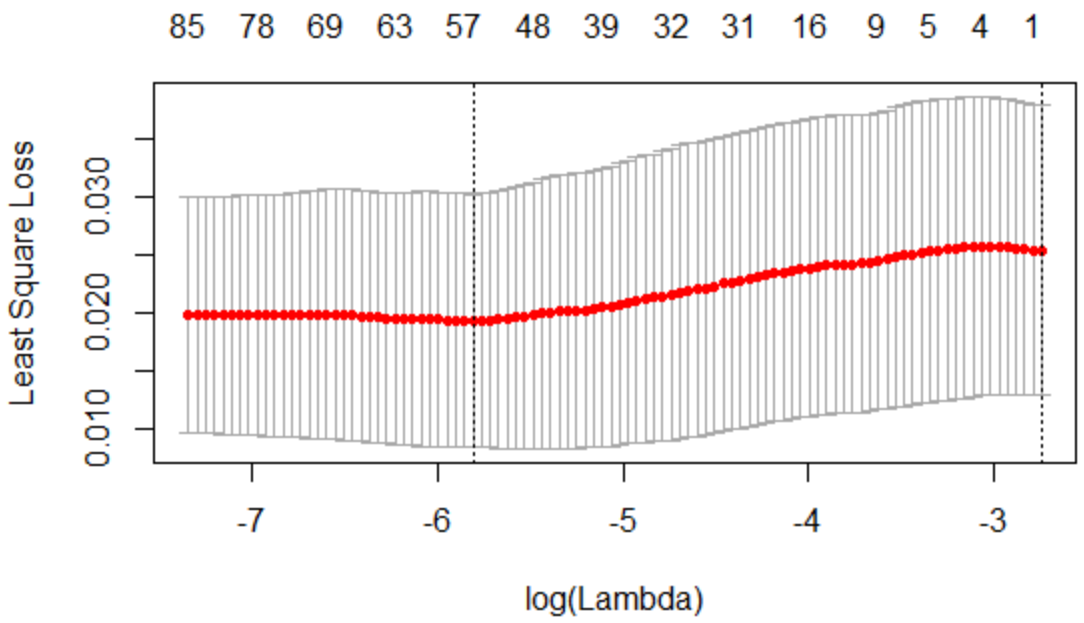

Figure 5: Lambda Plot for Adaptive ElasticNet model

Now with our lambda value, we can calculate the MSE for this particular adaptive elastic net model.

Code:

```
#Get the optimum lambda value
bestlam_adapt_en =cv.out$lambda.min
print(bestlam_adapt_en)

#Get the Mean Squared Error
adapt_en_pred=predict(adapt_en_model, s = bestlam_adapt_en , newx = X.test)
print(mean((adapt_en_pred - y.test)^2))
```

Output:

```
> print(bestlam_adapt_en)
[1] 0.004490548
> print(mean((adapt_en_pred - y.test)^2))
[1] 0.01043307
```

Hence the chosen lambda value is 0.004490548 and the MSE is 0.01043307

Code:

```
#Find number of non-zero coefficients
adapt_en.coef=coef(cv.out , s =bestlam_adapt_en , type="coefficients")
print(nnzero(adapt_en.coef))
```

Output:

```
> print(nnzero(adapt_en.coef))
[1] 44
```

There are 44 non-zero coefficients in the adaptive elastic net model.

Results

The mean squared error was calculated amongst all three models and the number of non-zero coefficients were also found per model.

The MSE of the Lasso model in predicting the *Scheetz.rds* is **0.01950226**, and the number of efficient coefficients affecting the predictions in Lasso is **12 out of 18976**.

The MSE of the Horseshoe model is 0.**01674154** and the number of efficient coefficients is **30 out of 18976**.

The MSE of the Adaptive Elastic Net model is **0.01043307** and the number of efficient coefficients is **44 out of 18976**.

Model	**Mean Squared Error (MSE)**	**Variables selected**
Lasso	0.01950226	12
Horseshoe	0.01674154	8
Adaptive Elastic Net	0.01043307	44

Table 1: Evaluation metrics of the models

Conclusion

From these sparse regression methods, we find that the horseshoe method results in the lowest number of features selected at eight while the adaptive elastic net method uses 44. The adaptive elastic net method had the lowest MSE at 0.01043307 while the lasso method had the highest at 0.01950226. What this likely means is that although the adaptive elastic net method collected the most features, it also likely retained more valuable features than either the lasso or horseshoe method.

Due to the nature of the “Gene expression in the mammalian eye” dataset, the high dimensionality and low sample size led to a sparsity problem when trying to fit a regression model. Each of the models proposed have their own respective strengths and weaknesses to solve this problem and which model is considered the “best” is based on what metric is deemed most important.

If the end goal is to obtain the least number of features with an acceptable error rate, the Horseshoe model would be suitable. If more relevant features can be added and the lowest error rate is to be prioritized, then the Adaptive ElasticNet model should be chosen. The Lasso model generates a decent baseline for this sparse regression problem, but unfortunately undergoes a few shortcomings such as the lesser results compared to the other two models and the saturation problem which the Lasso model can encounter if the number of features increases in further training (the Lasso model cannot have the number of features exceed the sample size of the data, and hence additional relevant features may be lost due to this).

Citations

- Scheetz TE, Kim K-YA, Swiderski RE, Philp AR, Braun TA, Knudtson KL, Dorrance AM, DiBona GF, Huang J, Casavant TL, Sheffield VC and Stone EM (2006). Regulation of gene expression in the mammalian eye and its relevance to eye disease. *Proceedings of the National Academy of Sciences*, **103**: 14429-14434.
- Carlos M. Carvalho, Nicholas G. Polson, & James G. Scott (2009). Handling sparsity via the horseshoe*. Journal of Machine Learning Research, W&CP*, 9.
- Zou H, Zhang HH: On the adaptive elastic-net with a diverging number of parameters. The Annals of Statistics 2009, 37:1733-1751.