IIT KANPUR

UNDERGRADUATE PROJECT

DEPARTMENT OF MATHS AND STATISTICS

Feature Selection

Author:
Siddhant GARG
150711
Department of Maths and Statistics
siddhant@iitk.ac.in

Supervisor:
Prof. Debasis KUNDU
Department of Maths and Statistics
kundu@iitk.ac.in

December 14, 2018

Contents

Feature Selection

Siddhant Garg
IIT Kanpur
siddhant@iitk.ac.in

Abstract

The 'lasso' minimizes the residual sum of squares subject to the sum of absolute value of the coefficients being less that a constant. Because of ℓ_1 regularization, it tends to produce some coefficients that are zero and hence give interpretable models. This report discusses how to model data with geometric skew-normal distribution when mean of the distribution is zero with the lasso and the elastic net penalty. We run the model on three different datasets and compare the models with the standard lasso and elastic net models. We will see that the proposed model produced the similar results with that of the standard model.

1 Introduction

In statistics and machine learning, lasso (least absolute shrinkage and selection operator) (also Lasso or LASSO) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical model it produces. It alters the model fitting process to select only a subset of the provided covariates for use in the final model rather than using all of them. Ridge regression was the most popular technique for improving prediction accuracy. Ridge regression improves prediction error by shrinking large regression coefficients in order to reduce overfitting, but it does not perform covariate selection and therefore does not help to make the model more interpretable. Lasso is able to achieve both of these goals by forcing the sum of the absolute value of the regression coefficients to be less than a fixed value, which forces certain coefficients to be set to zero, effectively choosing a simpler model that does not include those coefficients. This idea is similar to ridge regression, in which the sum of the squares of the coefficients is forced to be less than a fixed value, though in the case of ridge regression, this only shrinks the size of the coefficients, it does not set any of them to zero. But lasso has its limitations [9]. In $p > n$ case, the lasso selects atmost n variables because of the nature of the convex optimzation problem. Also, if there are variables among which the pairwise correlations are very high, lasso selects only one of them without caring which is it. In high dimensional dataset, ridge regression dominates lasso regression. Hui and Hastie proposed a new regularization method called *elastic net*. Elastic net penalty consists of both ℓ_1 and ℓ_2 penalty. It can select group of correlated variables and can also do variable selection and shrinkage.

2 The Lasso Problem

This was first given by Tibshirani [7] in 1996 Suppose we have data $(\mathbf{x}^i, y_i)$, $i = 1, \ldots, N$, where $\mathbf{x}^i = (x_{i1}, \ldots, x_{ip})$ are predictor variables and y_i are responses. We assume that x_{ij} are standardized so that $s \sum_{i=1}^N x_{ij}/N = 0$ and $\sum_{i=1}^N x_{ij}^2/N = 1$. Letting $\hat{\boldsymbol{\beta}} = (\hat{\beta}_1, \ldots, \hat{\beta}_p)^T$, the lasso estimate $(\hat{\boldsymbol{\beta}}, \hat{b})$ is defined by

$$(\hat{\boldsymbol{\beta}}, \hat{b}) = \arg\min_{(\boldsymbol{\beta}, b)} \left\{ \sum_{i=1}^N \Big(y_i - b - \sum_j \beta_j x_{ij} \Big)^2 \right\} \quad \text{subject to } \sum_{j=1}^p |\beta_j| \leq t \tag{1}$$

Submitted to 29th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

Here $t \geq 0$ is the tuning hyperparameter. Now, for all t, the solution for b is $\hat{b} = \bar{y}$. We can, without the loss of generality assume that $\bar{y} = 0$, and hence can omit b. So, we can also write eq (1) as

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|^2 \quad \text{subject to } \|\boldsymbol{\beta}\|_1 \leq t \tag{2}$$

2.1 Why does ℓ_1 regularization yield sparse solutions

[6] This section explains why ℓ_1 regularization results in sparse solutions, whereas ℓ_2 regularization does not.

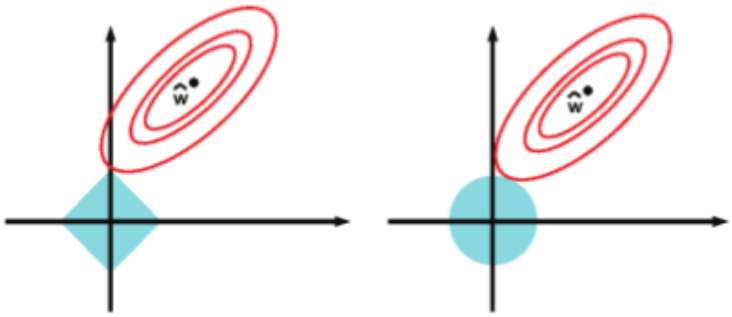


Figure 1: Illustration of ℓ_1 (left) vs ℓ_2 (right) regularization of a least squares problem. Fig. courtesy: [6]

In the Figure, we plot the contours of the RSS objective function, as well as the contours of the ℓ_1 and ℓ_2 constraint surfaces. From the theory of constrained optimization, we know that the optimal solution occurs at the point where the lowest level set of the objective function intersects the constraint surface (assuming the constraint is active). It should be geometrically clear that as we relax the constraint B, we "grow" the ℓ_1 "ball" until it meets the objective; the corners of the ball are more likely to intersect the ellipse than one of the sides, especially in high dimensions, because the corners "stick out" more. The corners correspond to sparse solutions, which lie on the coordinate axes. By contrast, when we grow the ℓ_2 ball, it can intersect the objective at any point; there are no "corners", so there is no preference for sparsity.

2.2 Co-ordinate Descent Algorithm

We will give the overview of how the lasso problem is solved. Details can be referred from here [6]. The lasso objective is given by:

$$f(\boldsymbol{\beta}) = \text{RSS}(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1$$

where $\text{RSS}(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$, and λ is the regularization parameter. The objective function is non-smooth function and hence derivative does not exits at $\beta_j = 0$. We will use the sub-derivative or sub-gradient of the convex function. For the absolute function $g(x) = |x|$, the subderivative is given by:

$$\partial f(x) = \left\{ \begin{array}{cc} \{-1\} & x < 0 \\ [-1, 1] & x = 0 \\ \{1\} & x > 0 \end{array} \right\}$$

Applying the above concepts to the lasso problem, we get

$$\frac{\partial \text{RSS}(\boldsymbol{\beta})}{\partial \beta_j} = a_j \beta_j - c_j$$

$$a_j = 2 \sum_{i=1}^{n} x_{ij}^2$$

$$c_j = 2 \sum_{i=1}^{n} x_{ij}(y_i - \boldsymbol{\beta}_{-j}^T \mathbf{x}_{i,-j})$$

where $\boldsymbol{\beta}_{-j}$ is without the component j, and similarly for $\mathbf{x}_{i,-j}$. We see that c_j is proportional to the correlation between the j^{th} feature $\mathbf{x}_{:,j}$ and the residual due to other features, $\mathbf{r}_{-j} = \mathbf{y} - \mathbf{X}_{:,-j}\boldsymbol{\beta}_{-j}$. Hence the magnitude of c_j is and indication of how relevant feature j is for predicting $\mathbf{y}$ (relative to the other features and the current parameters). The subderivative of the objective function is given by:

$$\begin{aligned} \partial_{\beta_j} f(\boldsymbol{\beta}) &= (a_j \beta_j - c_j) + \lambda \partial_{\beta_j} \|\boldsymbol{\beta}\|_1 \\ &= \left\{ \begin{array}{cc} \{a_j \beta_j - c_j - \lambda\} & if\ \beta_j < 0 \\ [a_j \beta_j - c_j - \lambda, a_j \beta_j - c_j + \lambda] & if\ \beta_j = 0 \\ \{a_j \beta_j - c_j + \lambda] & if\ \beta_j\} > 0 \end{array} \right\} \end{aligned}$$

Depending on the value of c_j, the solution to $\partial_{\beta_j} f(\boldsymbol{\beta}) = 0$ can occur at 3 different values of β_j.

Case 1: If $c_j < \lambda$, then the feature is strongly negatively correlated with the residual, then the subgradient is zero at $w_j = \frac{c_j + \lambda}{a_j} < 0$.

Case 2: If $c_j \in [-\lambda, \lambda]$, so the feature is weakly correlated with the residual, then the subgradient is zero at $\beta_j = 0$.

Case 3: if $c_j > \lambda$, so the feature is strongly positively correlated with the residual, then the subgradient is zero at $beta_j = \frac{c_j - \lambda}{a_j} > 0$.

$$\hat{\beta}_j(c_j) = \left\{ \begin{array}{ccc} (c_j + \lambda)/a_j & if & c_j < -\lambda \\ 0 & if & c_j \in [\lambda, \lambda] \\ (c_j - \lambda)/a_j & if & c_j > \lambda \end{array} \right\}$$

We can write this as follows:

$$\hat{\beta}_j = \text{soft}\Big(\frac{c_j}{a_j}; \frac{\lambda}{a_j}\Big)$$

where $\text{soft}(a; \delta) = \text{sign}(a)(|a| - \delta)_+$, where $x_+ = \max(x, 0)$. This is called soft-thresholding.

Algorithm 1 Co-ordinate Descent for Lasso

1: Initialize: $\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{x}^T\mathbf{y}$;
2: **repeat**
3: **for** $i = 1, \ldots, D$ **do**
4: $a_j = 2\sum_{i=1}^{n} x_{ij}^2$
5: $c_j = 2\sum_{i=1}^{n} x_{ij}(y_i - \beta^T\mathbf{x}_i + \beta_j x_i j)$
6: $\beta_j = \text{soft}\Big(\frac{c_j}{a_j}, \frac{\lambda}{a_j}\Big)$;
7: **until** Not converged

When $\lambda = 0$, we get the OLS solution. If $\lambda > \lambda_{max}$, we get $\hat{\boldsymbol{\beta}} = 0$, where $\lambda_{max} = max_j \left|\mathbf{y}^T\mathbf{x}_{:,j}\right|$

3 Elastic Net

It was first introduced by Hui Zou and Trevor Hastie [9]. Definition: Suppose we have data $(\mathbf{x}^i, y_i)$, $i = 1, \ldots, N$, where $\mathbf{x}^i = (x_{i1}, \ldots, x_{ip})$ are predictor variables and y_i are responses. We assume that x_{ij} are standardized so that $s\sum_{i=1}^{N} x_{ij}/N = 0$ and $\sum_{i=1}^{N} x_{ij}^2/N = 1$. Letting $\hat{\boldsymbol{\beta}} = (\hat{\beta}_1, \ldots, \hat{\beta}_p)^T$, the lasso estimate $(\hat{\boldsymbol{\beta}}, \hat{b})$ is defined by

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\{\alpha \|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2^2\}$$

Here, $\lambda \geq 0$ is regularization hyperparameter and $0 < \alpha < 1$ is the mixing ratio. The above optimization problem can be solved with co-ordinate descent algorithm, whose details we are going to omit.

4 Geometric Skew-Normal Distribution

A normal random variable with mean μ and variance σ^2 will be denoted by $\mathcal{N}(\mu, \sigma^2)$. A geometric random variable with parameter p will be denoted by GE(p), and it has the probability mass function (PMF); $p(1-p)^{n-1}$, for $n = 1, 2, \ldots$. Now we define GSN distribution with parameters μ, σ and p as follows.

Definition : Suppose $N \sim GE(p)$, $\{X_i : i = 1, \ldots\}$ are i.i.d. $N(\mu, \sigma^2)$ random variables, and N and X_i 's are independently distributed. Define

$$\boldsymbol{X} \stackrel{\mathrm{d}}{=} \sum_{i=1}^{N} X_i$$

. Then X is said to be GSN random variable with parameters μ, σ and p. It will be denoted as $GSN(\mu, \sigma, p)$. The joint PDF, $f_{X,N}(x, n)$ of (X, N) is given by

$$f_{X,N}(x,n) = \begin{cases} \frac{1}{\sigma\sqrt{2\pi n}} e^{-\frac{1}{2n\sigma^2}(x-n\mu)^2} p(1-p)^{n-1} & 0 < p < 1 \\ \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} & p = 1 \end{cases}$$

for $-\infty < x < \infty$, $\sigma > 0$ and for any positive integer n. If $p \neq 1$ the PDF of X becomes,

$$\begin{aligned} f_X(x) &= \sum_{n=1}^{\infty} f_{X,N}(x,n) \\ &= \sum_{n=1}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2n\sigma^2}(x-n\mu)^2} p(1-p)^{n-1} \end{aligned}$$

Generation from GSN

- Step 1 : Generate from GE(p)
- Step 2 : Generate x from $N(m\mu, m\sigma^2)$,and x is the required sample.

Conditional Distributions and Expectations: The following conditional distributions will be required for further development.

$$\begin{aligned} P(N|X=x) &= \frac{P(X=x, N=n)}{P(X=x)} \\ &= \frac{(1-p)^{n-1} e^{\frac{-1}{2\sigma^2 n}(x-n\mu)^2}/\sqrt{n}}{\sum_{k=1}^{\infty}(1-p)^{k-1} e^{\frac{-1}{2\sigma^2 k}(x-k\mu)^2}/\sqrt{k}} \\ \mathbf{E}[N|X=x] &= \frac{\sum_{n=1}^{\infty}(1-p)^{n-1} e^{\frac{-1}{2\sigma^2 n}(x-n\mu)^2}\sqrt{n}}{\sum_{k=1}^{\infty}(1-p)^{k-1} e^{\frac{-1}{2\sigma^2 k}(x-k\mu)^2}/\sqrt{k}} \end{aligned}$$

$$\mathbf{E}[N^{-1}|X=x] = \frac{\sum_{n=1}^{\infty}(1-p)^{n-1} e^{\frac{-1}{2\sigma^2 n}(x-n\mu)^2}/n^{3/2}}{\sum_{k=1}^{\infty}(1-p)^{k-1} e^{\frac{-1}{2\sigma^2 k}(x-k\mu)^2}/\sqrt{k}}$$

4.1 EM-Algorithm for estimation of parameters

Suppose we have data $\{x_1, \ldots, x_n\}$ which is a random sample from GSN(μ, σ, p). We want to estimate the parameters μ, σ, p. Following the ref, based on the compelete sample $\{(x_1, m_1), \ldots, (x_n, m_n)\}$ the psuedo-log likelihood is given by:

$$l(\mu, \sigma, p) = -n \ln \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \frac{(x_i - m_i\mu)^2}{m_i} + n \ln p + \ln(1-p) \sum_{i=1}^{n} (m_i - 1)$$

Based on the complete sample, the MLEs of the unknown parameters can be obtained in explicit forms.

$$\hat{\mu} = \frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} m_i}, \quad \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} \frac{(x_i - \hat{\mu})^2}{m_i}, \quad \hat{p} = \frac{n}{\sum_i^n m_i}$$

4.2 Data Analysis and Feature Selection

Suppose we have a data sample $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i$ are the predictors and y_i are the responses. We want to find $\boldsymbol{\beta}$ such that $y_i = \boldsymbol{\beta}^T \mathbf{x}_i + \epsilon$, where $\epsilon \sim$ GSN(μ, σ, p). In this report we will assume $\mu = 0$ for simplicity. Therefore, we want to estimate $\boldsymbol{\beta}$, σ and p. Therefore,

$$\begin{aligned} \epsilon &\sim \text{GSN}(0, \sigma, p) \\ y_i - \boldsymbol{\beta}^T \mathbf{x}_i &\sim \text{GSN}(0, \sigma, p) \\ z_i &\sim \text{GSN}(0, \sigma, p) \end{aligned}$$

where $z_i = y_i - \boldsymbol{\beta}^T \mathbf{x}_i$. Using the EM-Algorithm, we will get the estimates of the parameters σ and p. Based on the complete sample, the pseudo log-likelihood function is given by:

$$\begin{aligned} l(\boldsymbol{\beta}, \sigma, p) &= -n \ln \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \frac{z_i^2}{m_i} + n \ln p + \ln(1-p) \sum_{i=1}^{n} (m_i - 1) \\ &= -n \ln \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \frac{(y_i - \boldsymbol{\beta}^T \mathbf{x}_i)^2}{m_i} + n \ln p + \ln(1-p) \sum_{i=1}^{n} (m_i - 1) \end{aligned}$$

Suppose, we want to select a subset of features that are most relevant to the response variable. For doing the feature selection problem we will minimize the following objective function:

$$\mathcal{L}(\boldsymbol{\beta}, \sigma, p) = -l(\boldsymbol{\beta}, \sigma, p) + \lambda \left\| \boldsymbol{\beta} \right\|_1$$

Note that the ℓ_1 penalty on beta will encourage sparse solutions and only those features will be non-zero that are most relevant in predicting the response variable. For solving for σ and p, we will differentiate the objective function with respect to σ^2 and p and setting it to zero. Therefore, we get

$$\hat{\sigma^2} = \frac{1}{n} \sum_{i=1}^{n} \frac{(y_i - \boldsymbol{\beta}^T \mathbf{x}_i)^2}{m_i} \quad \hat{p} = \frac{n}{K}, \quad K = \sum_{i=1}^{n} m_i$$

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= \arg\min_{\boldsymbol{\beta}} \; -\ell(\boldsymbol{\beta}) + \left\| \boldsymbol{\beta} \right\|_1 \\ &= \arg\min_{\boldsymbol{\beta}} \; \frac{1}{2\sigma^2} \sum_{i=1}^{n} \frac{(y_i - \boldsymbol{\beta}^T \mathbf{x}_i)^2}{m_i} + \lambda \left\| \boldsymbol{\beta} \right\|_1 \end{aligned}$$

which can be solved by using co-ordinate descent algorithm. Notea that we have to replace the values of m_i and m_i^{-1} using $\mathbf{E}[N|X = x]$ and $\mathbf{E}[N^{-1}|X = x]$. The overall sketch of the EM-Algorithm is as follows:

Algorithm 2 EM-Algorithm for parameter estimation and feature selection

1: **Initialize:** $(\boldsymbol{\beta}^{(0)}, \sigma^{(0)}, p^{(0)})$.
2: **repeat** for $k = 1, 2, \ldots,$
3: **for** $i = 1, \ldots, n$ **do**
4: $a_i^{(k)} = \mathbf{E}[N|X = (y_i - \boldsymbol{\beta}^{(k-1)^T}\mathbf{x}_i), \sigma^{(k-1)}, p^{(k-1)}]$
5: $b_i^{(k)} = \mathbf{E}[N^{-1}|X = (y_i - \boldsymbol{\beta}^{(k-1)^T}\mathbf{x}_i), \sigma^{(k-1)}, p^{(k-1)}]$
6: Update the parameters.

$$\boldsymbol{\beta}^{(k)} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2\sigma^{2^{(k-1)}}} \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^T \mathbf{x}_i)^2 b_i^{(k)} + \lambda \|\boldsymbol{\beta}\|_1$$

$$\sigma^{2^{(k)}} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^{(k)^T} \mathbf{x}_i)^2 b_i^{(k)}$$

$$p^{(k)} = \frac{n}{\sum_{i=1}^{n} a_i^{(k)}}$$

7: **until** Convergence

The data can also be modelled using the elastic net penalty. The modified loss function is given by:

$$\mathcal{L}(\boldsymbol{\beta}, \sigma, p) = -l(\boldsymbol{\beta}, \sigma, p) + \lambda\{\alpha \|\boldsymbol{\beta}\|_1 + (1-\alpha) \|\boldsymbol{\beta}\|_2^2\}$$

The EM Algorithm will follow on the same lines except on the step where we update $\boldsymbol{\beta}$. The modified update will be

$$\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^T \mathbf{x}_i)^2 b_i + \lambda\{\alpha \|\boldsymbol{\beta}\|_1 + (1-\alpha) \|\boldsymbol{\beta}\|_2^2\}$$

We used *Lasso*, *ElasticNet* python packages of *sklearn.linear_model* [5] to solve both lasso and elastic net optimization problems.

5 Experiments and Results

We run and tested our model on three different datasets. We present the training error and test error on these datasets and compare the results of our model with the existing baselines.

5.1 Dataset : mtcars

The dataset contains data extracted from the *Motor Trend US magazine*, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 models of car $(N = 32, p = 10)$. There are 32 observation and 10 features, the *response variable* we want to study is **mpg**, that is the miles per gallon (or fuel efficiency) [2]. The explanatory variables are:

- cyl : Number of cylinders
- disp : Displacement (volume of the engine)
- hp : Gross horsepower
- drat : Rear axle ratio
- wt : Weight (1000 lbs)
- qsec : 1/4 mile time
- vs : V/S engine
- am : Transmission (0 = automatic, 1 = manual)11

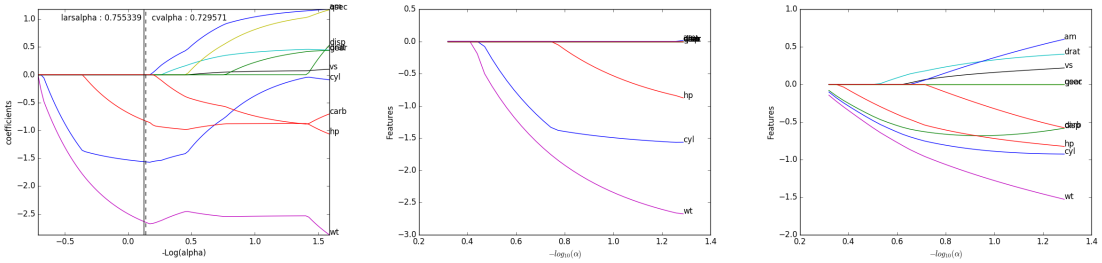
- gear : Number of forward gears
- carb : Number of carburetors

The goal of our analysis is to underline which explanatory variables are most relevant to predict the response variable, mpg, and in order to do so we will use the LASSO method and the proposed Lasso-GSN and ENET-GSN methods.

For implementation of the Lasso objective *sklearn.linear_model* [5] will be used. The lasso objective that is used is

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \alpha \|\mathbf{w}\|_1$$

where α is a tuning hyperparameter. In the figure below, two values of α are indicated with vertical lines. One is selected by Cross-Validation. Both these values are similar and selects 3 most important features for the model, namely, *wt*, *cyl* and *hp*.



(a) Lasso (b) Lasso GSN (c) Enet GSN

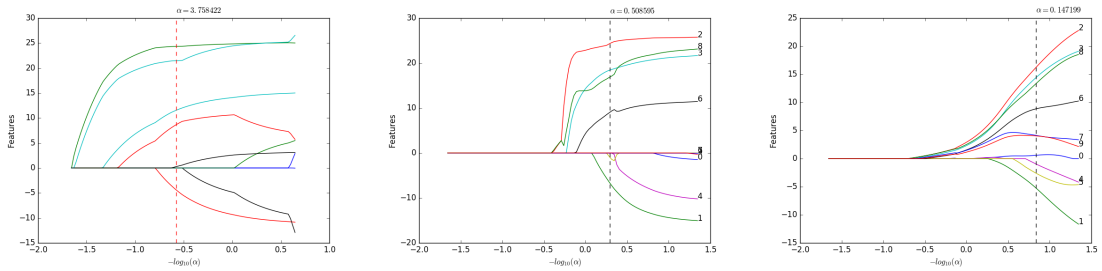
Figure 2: Plots of α vs features

-	Lasso	E-Net	Lasso-GSN	Enet-GSN
MSEs	409.50	410.155	409.668	409.418
#Non-Zero Coef	5	5	4	8

Table 1: Mean-Square-Errors after estimating β by through different models

5.2 Diabetes Dataset

This was first used in LARS paper [1]. This dataset consist of observations on 442 patients, with the response of interest being a quantitative measure of disease progression one year after baseline. There are ten baseline variables - age, sex, body-mass index, average blood pressure, and six blood serum measurements-plus quadratic terms, giving a total of 10 features.



(a) Lasso (b) Lasso GSN (c) Enet GSN

Figure 3: Plots of α vs features

-	Lasso	E-Net	Lasso-GSN	Enet-GSN
MSEs	3019.47	2992.083	2988.837	3005.352
Test-error	2807.944	2808.003	2781.205	2758.712
#Non-Zero Coef	6	10	6	10

Table 2: Mean-Square-Errors after estimating $\boldsymbol{\beta}$ by through different models

5.3 Arcene Dataset

This dataset was used for feature selection challenge [3] ARCENE's task is to distinguish cancer versus normal patterns from mass-spectrometric data. This is a two-class classification problem with continuous input variables. This dataset is one of 5 datasets of the NIPS 2003 feature selection challenge. ARCENE was obtained by merging three mass-spectrometry datasets to obtain enough training and test data for a benchmark. The original features indicate the abundance of proteins in human sera having a given mass value. Based on those features one must separate cancer patients from healthy patients. We added a number of distractor feature called 'probes' having no predictive power. The order of the features and patterns were randomized. The dataset consists of $n = 100$ samples and $p = 10,000$ features for each sample.

-	Lasso	E-Net	Lasso-GSN	Enet-GSN
MSEs	0.215629	0.237298	0.014400	0.014400
Test-error	0.829951	0.803237	0.670685	0.692613
#Non-Zero Coef	45	74	437	572

Table 3: Mean-Square-Errors after estimating $\boldsymbol{\beta}$ by through different models

6 Future Work and Conclusions

We implemented and compared the proposed models with the existing baselines and we see that for some dataset, the model performs similar to the baselines in terms of selecting the features and training and testing errors like in mtcars dataset. In Arcene dataset, however, the proposed model selects far more number of parameters than that of existing baselines and giving significantly less training and training and testing error respectively. So our proposed model outperforms the existing baselines on this dataset.

In this project we only considered the modelling of the data with the **GSN** distribution with zero mean. We can also try to model the data for the more general case when the mean μ is not zero. Note that the resulting problem will change significantly for the step where we have to estimate the regression parameters $\boldsymbol{\beta}$. The resulting problem will be:

$$\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^T \mathbf{x}_i - a_i\mu)^2 b_i + \lambda \|\boldsymbol{\beta}\|_1 + (1-\lambda) \|\boldsymbol{\beta}\|_2^2$$

Note that the above problem is not the ordinary least squares problem because we will have both square and linear terms for the residual and cannot be solved the way that was done here.

Appendix

A Lawson and Hanson Algorithm for finding Lasso solution

We fix $t \geq 0$. Problem (2) can be expressed as a least squares problem with 2^p inequality constraints. Lawson and Hanson (1974) provided a procedure for solving the general linear inequality constraints $G\boldsymbol{\beta} \leq h$. G is $m \times p$ matrix of m linear inequalities. Our problem can be solved by introducing the inequalities sequentially and obtaining the feasible solutions, satisfying the KKT conditions.

Let $g(\boldsymbol{\beta}) = \sum_{i=1}^{N}(y_i - \sum_j \beta_j x_{ij}^2)$ and let $\boldsymbol{\delta}_i, i = 1, \ldots, 2^p$, be the $p-tuples$ of form $(\pm 1, \ldots, \pm 1)_p$. Then the condition $\sum_j |\beta_j| \leq t$ is equivalent to $\boldsymbol{\delta}_i^T \boldsymbol{\beta} \leq t$ for all i. For a given $\boldsymbol{\beta}$, let $E = \{i : \boldsymbol{\delta}^T \boldsymbol{\beta} = t\}$ and $S = \{i : \boldsymbol{\delta}^T \boldsymbol{\beta} < t\}$. Denote G_E matrix whose rows are $\boldsymbol{\delta}_i$ for $i \in E$. Let $\mathbf{1}$ be the vector of ones whose length is number of rows of G_E.

Algorithm 3 Lawson and Hanson Method

1: $g(\boldsymbol{\beta}) = \sum_{i=1}^{N}(y_i - \sum_j \beta_j x_{ij}^2)$
2: **Initialize** : $E = \{i_0\}$, where $\boldsymbol{\delta}_{i_0} = sign(\hat{\boldsymbol{\beta}}_0)$, $\hat{\boldsymbol{\beta}}_{\mathbf{0}}$: Least squares estimate
3: Find $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \ g(\boldsymbol{\beta})$ subject to $G_E \boldsymbol{\beta} \leq t\mathbf{1}$.
4: **while** $\sum_j |\beta_j| > t$ **do**
5: add i to the set E, where $\boldsymbol{\delta}_i = sign(\hat{\boldsymbol{\beta}})$
6: Find $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \ g(\boldsymbol{\beta})$ subject to $G_E \boldsymbol{\beta} \leq t\mathbf{1}$

The step 3 and 6 of the above algorithm are quadratic programming problems. These type of problems can be solved using Active Set Method and Interior Point Methods. Following section will describe the quadratic programming and interior point method more formally.

B Quadratic Programming

Quadratic programming (QP) is the process of solving a special type of mathematical optimization problem, specifically, a quadratic optimization problem, that is, the problem of optimizing (minimizing or maximizing) a quadratic function of several variables subject to linear constraints on these variables. General form of QP is given by

$$\min_{x \in \mathbb{R}^n} \ \frac{1}{2} x^T Q x + q^T x \quad \text{subject to}$$

$$Ax = a$$
$$Bx \leq b$$

where $Q \in \mathbb{R}^{n \times n}$ symmetric matrix (not necessarily positive definite matrix) and $A \in \mathbb{R}^{m_1 \times n}$ and $B \in \mathbb{R}^{m_2 \times n}$ and $m_1 \leq n$.

C Interior-Point Methods

[8] [4] The primal–dual interior-point approach can be applied to convex quadratic programs through a simple extension of the linear-programming algorithms. One characteristic of these methods was that they required all iterates to satisfy the inequality constraints in the problem strictly, so they soon became known as interior-point methods. We will discuss the primal-dual interior-point method for solving the inequality constrained convex quadratic program

$$\min_{x \in \mathbb{R}^n} \ f(x) := \frac{1}{2} x^T G x + g^T x \quad \text{subject to} \tag{3}$$

$$A^T x \geq b$$

x is an $n \times 1$ vector where n is the number of variables, G is a $n \times n$ matrix, g is a $n \times 1$ vector, A is a $n \times m$ matrix, where m is the number of inequality constraints, and b is a $m \times 1$ vector.

C.1 Optimality Conditions

In this section we state the optimality (KKT) conditions for the inequality constrained QP and show how to use Newton's method to progress iteratively towards the solution. The corresponding Lagrangian for the eq (3) is :

$$\mathcal{L}(x, \lambda) = \frac{1}{2} x^T G x + g^T x - \sum_{i=1}^{m} \lambda_i (a_i^T x - b)$$

where $\lambda = (\lambda_1, \ldots, \lambda_m)$ are the langrange multipliers for each of the i^{th} inequality constraint and a_i^T is the i^{th} row in A^T and b_i is the i^{th} element in b. The optimality conditions are:

$$\begin{aligned} \nabla_x \mathcal{L}(x, \lambda) = 0 \implies Gx + g - \lambda A &= 0 \\ A^T x - b &\geq 0 \\ \lambda_i (A^T x - b)_i &= 0 \\ \lambda_i &\geq 0 \end{aligned}$$

The above conditions are rewritten by introducing a slack vector $s = A^T x - b, s \geq 0$ to simplify the notation.

$$Gx + g - \lambda A = 0 \tag{4a}$$

$$s - A^T x + b = 0 \tag{4b}$$

$$s_i \lambda_i = 0 \tag{4c}$$

$$(\lambda_i, s) \geq 0 \tag{4d}$$

We now define the function $F(x, \lambda, s)$ such that the roots of this function are solutions to the first three optimality conditions above.

$$F(x, \lambda, s) = \begin{bmatrix} Gx - A\lambda - g \\ s - A^T x + b \\ S\Lambda \mathbf{e} \end{bmatrix}$$

$S = diag(s_1, \ldots, s_m), \Lambda = diag(\lambda_1, \ldots, \lambda_m)$ and $\mathbf{e} = (1, \ldots, 1)_m$. Because we consider convex problems we know that G is positive semi-definite and furthermore that the constraints are affine which means that the optimality conditions stated above are both necessary and sufficient. Additionally the solution vector will be a global minimizer.

Primal–dual methods find solutions (x^*, λ^*, s^*) of this system by applying variants of Newton's method to the three equalities above and modifying the search directions and step lengths so that the inequalities $(\lambda, s) \geq 0$ are satisfied strictly at every iteration. By applying Newton's method to $F(x, \lambda, s) = 0$ we obtain a search direction from the current iterate (x^k, λ^k, s^k), as shown below where $J(x, \lambda, s)$ is the Jacobian of $F(x, \lambda, s)$.

$$J(x, \lambda, s) \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = -F(x, \lambda, s)$$

Newton's method forms a linear model for F around the current point and obtains the search direction $(x, \Delta\lambda, \Delta s)$ by solving the above system. Writing out the Jacobian we obtain the following system of equations shown below:

$$\begin{bmatrix} G & -A & 0 \\ -A^T & 0 & \mathbf{I} \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -S\Lambda \mathbf{e} \end{bmatrix} \tag{5}$$

If we solve this system iteratively we should reach the optimal point. In practice however this raw approach is not used because a full Newton step will generally be infeasible, because it will violate the bound $(\lambda, s) \geq 0$. Instead a line search is performed along the Newton direction and we find the new iterate as:

$$(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha(\Delta x, \Delta \lambda, \Delta s)$$

for some line search parameter $\alpha \in [0, 1)$. Unfortunately, we often can take only a small step along the direction ($\alpha << 1$) before violating the condition $(\lambda, s) > 0$, hence, the pure Newton direction, which is known as the *affine scaling direction*, often does not allow us to make much progress toward a solution.

The primal-dual methods modify the basic Newton procedure in two important ways:

1. They bias the search direction toward the interior of the non-negative orthant $(\lambda, s) \geq 0$, so that we can move further along the direction before one of the components of (λ, s) becomes negative.
2. They keep the components of(λ, s) from moving "too close" to the boundary of the non-negative orthant.

C.2 Central Path

The central path is defined as

$$F(x, \lambda, s) = \begin{bmatrix} 0 \\ 0 \\ \tau \mathbf{e} \end{bmatrix} \tag{6}$$

It is an arc (curve) of strictly feasible points that is parametrized by a scalar $\tau > 0$. We note that points on the central path are strictly feasible and that $r_d = 0$ and $r_p = 0$ for these points. The idea is to have the iterates (x^k, λ^k, s^k) progress along this central path and to have τ decrease with each step. As τ approaches zero, the equations (6) will approximate the previously defined optimality conditions better and better. This means that the central path follows a path to the solution such that $(\lambda, s) > 0$ and such that the pairwise products $s_i \lambda_i$ are decreased to zero at the same rate. The purpose of using this concept is that we expect to obtain the fastest rate of convergence by having the iterates follow (approximate) the central path. From (6) we also see that the points on the central path satisfy a slightly perturbed version of the optimality conditions, with the only difference, compared to the optimality conditions, being the term $\tau \mathbf{e}$ on the right hand side. We can define the central path as

$$\mathcal{C} = \{(x_\tau, \lambda_\tau, s_\tau) | \tau > 0\}$$

Primal–dual algorithms take Newton steps toward points on $\mathcal{C}$ for which $\tau > 0$, rather than pure Newton steps for F. Since these steps are biased toward the interior of the non-negative orthant defined by $(\lambda, s) \geq 0$, it usually is possible to take longer steps along them than along the pure Newton steps for F, before violating the positivity condition.

To describe the biased search direction, we introduce a centering parameter $\sigma \in [0, 1]$ and a duality measure μ defined by

$$\mu = \frac{1}{n} \sum_{i=1}^{m} \lambda_i s_i = \frac{x^T s}{m}$$

which measures the average value of the pairwise products $x_i s_i$. By writing $\tau = \sigma \mu$ and applying Newton's method to the system (6), we obtain

$$\begin{bmatrix} G & -A & 0 \\ -A^T & 0 & \mathbf{I} \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -S\Lambda \mathbf{e} + \sigma \mu \mathbf{e} \end{bmatrix} \tag{7}$$

The step $(\Delta x, \Delta \lambda, \Delta s)$ is a Newton step towards the point $(x_{\sigma\mu}, \lambda_{\sigma\mu}, s_{\sigma\mu}) \in \mathcal{C}$, at which the pairwise products $x_i s_i$ are all equal to $\sigma\mu$. In contrast, the step (5) aims directly for the point at which the KKT conditions (6) are satisfied. If $\sigma = 1$, the equations (7) define a *centering direction* ,a Newton step towards the point $(x_{mu}, \lambda_{mu}, s_{mu}) \in \mathcal{C}$, at which all the pairwise products $x_i s_i$ are identical to μ. Centering directions are usually biased strongly toward the interior of the non-negative orthant and make little, if any, progress in reducing the duality measure μ. However, by moving closer to $\mathcal{C}$, they set the scene for substantial progress on the next iteration. (Since the next iteration starts near $\mathcal{C}$, it will be able to take a relatively long step without leaving the non-negative orthant.) At the other extreme, the value $\sigma = 0$ gives the standard Newton step (5), sometimes known as the affine-scaling direction.

C.3 Predictor Corrector Method

In practice the predictor-corrector method proposed by Mehrotra is used. As indicated by the name there is both a predictor and a corrector step involved in the algorithm. In order to explain the purpose of each of these steps it is necessary to explain a few concepts first, including that of the central path.The basic idea is to solve the system eq(5) and then set up a second system of equations to correct this step by modifying the right hand side of eq(5).

So far we have assumed that the initial iterates (x^0, λ^0, s^0) are strictly feasible and satisfies all the equalities of eq(4). All subsequent iterates also respect these constraints, because of the zero right-hand-side terms in system (5). For most problems, however, a strictly feasible starting point is difficult to find. Infeasible-interior-point methods require only that the components of x^0 and s^0 be strictly positive. The search direction needs to be modified so that it improves feasibility as well as centrality at each iteration, but this requirement entails only a slight change to the step equation (5). If we define the residuals for the linear equations as

$$\begin{aligned} r_d &= Gx - A\lambda + g \\ r_p &= s - A^T x + b \\ r_{s\lambda} &= S\Lambda\mathbf{e} \end{aligned}$$

the modified step equations is :

$$\begin{bmatrix} G & -A & 0 \\ -A^T & 0 & \mathbf{I} \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_s \\ -r_{s\lambda} + \sigma\mu\mathbf{e} \end{bmatrix} \tag{8}$$

The search direction is still a Newton step toward the point $(x_{\sigma\mu}, \lambda_{\sigma\mu}, s_{\sigma\mu}) \in \mathcal{C}$.It tries to correct all the infeasibility in the equality constraints in a single step. If a full step is taken at any iteration (that is, $\alpha_k = 1$ for some k), the residuals become zero, and all subsequent iterates remain strictly feasible.

Predictor Step : In practice we start by solving the system (11), obtaining the so-called affine scaling direction $(\Delta x^{aff}, \Delta\lambda^{aff}, \Delta s^{aff})$ and then determine a step length α^{aff} for this step. Note that we will use aff in superscript to mark various vectors and scalars associated with the computation of the affine scaling direction.

$$\begin{bmatrix} G & -A & 0 \\ -A^T & 0 & \mathbf{I} \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x^{aff} \\ \Delta\lambda^{aff} \\ \Delta s^{aff} \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_s \\ -r_{s\lambda} \end{bmatrix} \tag{9}$$

Next the complementarity measure for the current iterate is computed along with a predicted complementarity measure μ^{aff} for the computed Newton step is:

$$\mu^{aff} = \frac{(s + \alpha^{aff}\Delta s^{aff})^T(\lambda + \alpha^{aff}\Delta\lambda^{aff})}{m}$$

By comparing the values of μ and μ^{aff} we determine if the computed affine scaling direction is a good search direction. If for example $\mu^{aff} << \mu$ we have a significant reduction in the complementarity measure and the search direction is good so little centering is needed. In practice the centering parameter σ is computed as:

$$\sigma = \left(\frac{\mu^{aff}}{\mu}\right)^3$$

C.4 Step Length Computation

After computing a search direction for the predictor step, and later on the corrector step, we need to decide how long a step we can take in the computed search direction so that we do not violate $(\lambda, s) > 0$. We will take the same approach of choosing the step length for each predictor and corrector step. In every iteration $(\lambda, s) > 0$, so,

$$\begin{aligned} \lambda + \alpha^{aff}\Delta\lambda^{aff} &\geq 0 \\ s + \alpha^{aff}\Delta s^{aff} &\geq 0 \end{aligned}$$

We will look at these equations seperately and for convenience use two seperate scaling parameters, α_λ^{aff} and and α_s^{aff}, such that $\alpha^{aff} = min(\alpha_\lambda^{aff}, \alpha_s^{aff})$.

For each of the equations there are three cases based on the sign of $\Delta\lambda^{aff}$ and Δs^{aff} respectively.

$$\begin{aligned} \Delta\lambda^{aff} > 0 &\implies \alpha_\lambda^{aff} = 1 \\ \Delta\lambda^{aff} = 0 &\implies \alpha_\lambda^{aff} = 1 \end{aligned}$$

$$\Delta\lambda_i^{aff} < 0 \implies \lambda_i + \alpha_\lambda^{aff}\Delta\lambda_i = 0 \implies \alpha_\lambda^{aff} = -\frac{\lambda_i}{\Delta\lambda_i}$$

From the above cases we see that α_λ^{aff} can be chosen as:

$$\alpha_\lambda^{aff} = \min_{i:\Delta\lambda<0}\left(1, -\frac{\lambda_i}{\Delta\lambda_i^{aff}}\right) \tag{10}$$

α_s^{aff} can be equivalently chosen as

$$\alpha_s^{aff} = \min_{i:\Delta s_i<0}\left(1, -\frac{s_i}{\Delta s_i^{aff}}\right) \tag{11}$$

Then α^{aff} can be chosen as:

$$\alpha^{aff} = min(\alpha_s^{aff}, \alpha_\lambda^{aff}) \tag{12}$$

The described approach for determining a step length for the predictor step is also employed to determine a step length α for the corrector step.

C.5 Corrector Step

First step of the algorithm solves the following system:

$$\begin{bmatrix} G & -A & 0 \\ -A^T & 0 & \mathbf{I} \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} {}^{aff} \\ \Delta\lambda^{aff} \\ \Delta s^{aff} \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_s \\ -r_{s\lambda} \end{bmatrix} \tag{13}$$

Above system yields:

$$(s_i + \Delta s_i^{aff})(\lambda_i + \Delta\lambda_i^{aff}) = \Delta s_i^{aff}\Delta\lambda_i^{aff} \neq 0$$

Therefore, we introduced a linearization error in affine scaling direction. To compensate for this error we use a corrector step and solve the system

$$\begin{bmatrix} G & -A & 0 \\ -A^T & 0 & \mathbf{I} \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x^{cor} \\ \Delta\lambda^{cor} \\ \Delta s^{cor} \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_s \\ -r_{s\lambda} + \Delta S^{aff}\Delta\Lambda^{aff}\mathbf{e} \end{bmatrix} \tag{14}$$

where $\Delta S^{aff} = diag(\Delta s_1^{aff}, \ldots, \Delta s_m^{aff})$ and $\Delta\Lambda^{aff} = diag(\Delta\lambda_1^{aff}, \ldots, \Delta\lambda_m^{aff})$ So to summarize we obtain an affine scaling direction by solving the system (**??**). This predictor step is used to compute the centering parameter and to define the right hand side for the corrector and centering step. The system we solve in practice to obtain the search direction contains both the centering and corrector contributions as :

$$\begin{bmatrix} G & -A & 0 \\ -A^T & 0 & \mathbf{I} \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x^{cor} \\ \Delta\lambda^{cor} \\ \Delta s^{cor} \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_s \\ -r_{s\lambda} + \Delta S^{aff}\Delta\Lambda^{aff}\mathbf{e} - \sigma\mu\mathbf{e} \end{bmatrix} \tag{15}$$

Algorithm 4 Predictor Corrector Method

Input : (x_0, λ_0, s_0) and G, A, g, b

Compute the residuals and the complemetarity measure

$$r_d = Gx_0 - g - A\lambda_0$$
$$r_p = s_0 - A^T x_0 + b$$
$$r_{s\lambda} = S_0\Lambda_0\mathbf{e}$$
$$\mu = \frac{s_0^T\lambda_0}{m}$$

while Stopping criteria are not satisfied **do Predictor Step** :

Obtain affine scaling direction $(\Delta x^{aff}, \Delta\lambda^{aff}, \Delta s^{aff})$ by solving

$$\begin{bmatrix} G & -A & 0 \\ -A^T & 0 & \mathbf{I} \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x^{aff} \\ \Delta\lambda^{aff} \\ \Delta s^{aff} \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_s \\ -r_{s\lambda} \end{bmatrix}$$

Compute α^{aff} using (12), (13), (14)

Compute μ^{aff}

$$\mu^{aff} = \frac{(s + \alpha^{aff}\Delta s)^T(\lambda + \alpha^{aff}\Delta\lambda)}{m}$$

Compute centering parameter σ

$$\sigma = \left(\frac{\mu^{aff}}{\mu}\right)^3$$

Corrector and centering step :

Obtain the search direction by solving

$$\begin{bmatrix} G & -A & 0 \\ -A^T & 0 & \mathbf{I} \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x^{cor} \\ \Delta\lambda^{cor} \\ \Delta s^{cor} \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_s \\ -r_{s\lambda} + \Delta S^{aff}\Delta\Lambda^{aff}\mathbf{e} - \sigma\mu\mathbf{e} \end{bmatrix}$$

Compute α using (12), (13), (14)

$$\lambda + \alpha\Delta\lambda^{cor} \geq 0$$
$$s + \alpha\Delta s^{cor} \geq 0$$

Update (x, λ, s)

Update residuals and complementarity measure

$$r_d = Gx - g - A\lambda$$
$$r_p = s - A^T x + b$$
$$r_{s\lambda} = S\Lambda\mathbf{e}$$
$$\mu = \frac{s^T\lambda}{m}$$

C.6 Implementation

In this section a practical way of solving eq() is discussed.

$$\begin{bmatrix} G & -A & 0 \\ -A^T & 0 & \mathbf{I} \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_s \\ -r_{s\lambda} \end{bmatrix} \tag{16}$$

We will separately solve the three equations a use back substitution to solve the other two. From the second block we have

$$\Delta s = -r_p + A^T x \tag{17}$$

From the third block coupled with the (18), we have

$$\begin{aligned} \Delta\lambda &= -S^{-1}(r_{s\lambda} + \Lambda\Delta s) \\ &= S^{-1}(-r_{s\lambda} + \Lambda r_p) - S^{-1}\Lambda A^T \Delta x \end{aligned} \tag{18}$$

Combining first block with (20) gives

$$\begin{aligned} -r_d &= G\Delta x - A\Delta\lambda \\ &= (G + AS^{-1}\Lambda A^T)\Delta x - AS^{-1}(-r_{s\lambda} + \Lambda r_p) \\ -r_d &= \bar{G}\Delta x + \bar{r} \end{aligned} \tag{19}$$

where

$$\begin{aligned} \bar{G} &= G + (AS^{-1}\Lambda)A^T \\ &= G + ADA^T \\ \bar{r} &= A(S^{-1}(r_{s\Lambda} - \Lambda r_p)) \end{aligned}$$

We can therefore first solve for Δx from

$$\bar{G}\Delta x = -\bar{g} = -(r_d + r) \tag{20}$$

and then use back substitution to solve $\Delta\lambda$ and Δs. In practice Cholesky Factorization is used for $\bar{G}$ to solve (20)

$$G = LL^T$$

L is lower traingular matrix. We solve two below equations.

$$\begin{aligned} L^T\Delta x &= Y \\ LY &= \bar{g} \end{aligned}$$

It is important to note that we only have to perform the computationally expensive factorization of the matrix once per iteration because the matrix $\bar{G}$ does not change during a single iteration.

In practice we do not take a full step but use a scaling parameter η to dampen the step to ensure convergence

$$(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \eta\alpha(\Delta x, \Delta\lambda, \Delta s)$$

Here $\eta = 0.95$ is a good choice.

Algorithm 1 requires us to solve the least squares constrained problem.

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= \arg\min_{\boldsymbol{\beta}} \ \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|^2 \\ &= \arg\min_{\boldsymbol{\beta}} \ (\mathbf{X}\boldsymbol{\beta} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\beta} - \mathbf{y}) \\ &= \arg\min_{\boldsymbol{\beta}} \left\{ \boldsymbol{\beta}^T(\mathbf{X}^T\mathbf{X})\boldsymbol{\beta} - 2(\mathbf{X}^T\mathbf{y})^T \right\} \quad \text{subject to } G_E\boldsymbol{\beta} \leq t\mathbf{1} \end{aligned}$$

Here G_E is same as in step 3 and 6 of **algorithm 1** Neccessary changes can be made in **Algorithm 2** to solve the above problem and obtain the desired solution.

References

[1] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[2] Valeria Fonti. Feature selection using lasso. 2007.

[3] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In *Advances in neural information processing systems*, pages 545–552, 2005.

[4] Thomas Reslow Kruth. Interior-point algorithms for quadratic programming, 2008.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] Christian Robert. Machine learning, a probabilistic perspective, 2014.

[7] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[8] Stephen Wright and Jorge Nocedal. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

[9] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.