ECE 273 Project: Coordinate Descent, Dykstra's and LASSO

Sanjeev Anthia Ganesh
A53308351
santhiag@eng.ucsd.edu

Abstract

This project aims to study Dykstra's projection algorithm and block co-ordinate descent to solve certain types of optimization problems. It also studies the application of these algorithms to solve a problem called LASSO.

1 Co-ordinate Descent Algorithm

Input: We are given a convex differentiable function

$$f(x) : R^n -> R$$

whose minimizer needs to be found. Let's assume we have found the global minimizer of this convex function. This statement translates to the fact that the minimizer is achieved along all the co-ordinate axes. In the Co-ordinate Algorithm, we shall explore the fact about the types of functions that Co-ordinate Descent Algorithm works on as well as convergence criteria need for these functions.

1.1 Algorithm

In Block Coordinate Descent algorithm, we first choose a point

$$x^0 = (x_1^0, x_2^0, ...x_n^0) \, \epsilon \, dom(f)$$

. In each iteration, we choose a single coordinate or a block of coordinates and minimize the function along each of these coordinates. At iteration $r+1, r >= 0$, given, $x^r = (x_1^r, x_2^r, ..., x_n^r), x^r \, \epsilon \, dom(f)$ choose an index $s = 1, 2, 3...n$ and compute a new iterate

$$x^{r+1} = (x_1^{r+1}, x_2^{r+1}, ..., x_n^{r+1}) \, \epsilon \, dom(f)$$

that satisfies the following equation,

$$x_s^{r+1} \, \epsilon \arg\max_{x_s} \; f(x_1^r, x_2^r, .., x_s, x_{s+1}^r, ..x_N^r)$$

$$x_j^{r+1} = x_j^r, \forall j \neq s$$

This algorithm is repeated across all the indexes and multiple iterations till convergence to the global minimizer is achieved.

The regularized regression problem can be stated as follows.

$$\begin{aligned} min_{\beta \, \epsilon \, R^p} (1/2) * ||y - X\beta||_2^2 + \sum_{k=1}^{N} h_i(w_i) \\ where \, \beta \, \epsilon \, R^{p*1} \, is \, a \, feature \, vector \\ X \, \epsilon \, R^{N*p} \, is \, a \, transformation \, matrix \\ y \, \epsilon \, R^{N*1} \, is \, a \, observation \, vector. \end{aligned} \tag{1}$$

On an overall level, the co-ordinate descent algorithm for a regularized regression problem given above can be represented as

$$\beta_i^{(k)} = \arg \min_{\beta \, \epsilon \, R^p} (1/2) * ||y - \sum_{j<i} X_j \beta_j - \sum_{j>i} X_j \beta_j||_2^2 + h_i(w_i) \qquad i = 1, 2...d \tag{2}$$

1.2 Compatible function types and Convergence of the Algorithm

The characterization of the objective function for convergence has been explored by Tseng and we shall use Lemma 3.1 and Theorem 4.1a of his paper to prove convergence of the Co-ordinate Descent Algorithm.

For Block Coordinate Descent algorithm to be compatible with the function whose minimizer needs to be found, the function needs to have certain properties. We see that if the function is convex differentiable, then co-ordinate descent would tend to achieve a minimum. If the function is convex but not differentiable, there is a question whether block-coordinate descent could achieve the minimizer since the function is not smooth. Tseng, 2001 has provided several conditions needed by a function that is convex and non-differentiable to compatible with the Co-ordinate Descent Algorithm. Consider a function

$$f(x_1, x_2, ..x_n) = f_0(x_1, x_2, ..x_n) + \sum_{k=1}^{N} f_k(x_k)$$

Here, in this function, we seperated the differentiable part f_0 from the non-differentiable part f_k. Assume that the level set $X^0 = x : f(x) <= f(x^0)$ is compact and f is continuous on X^0 where x^0 is the initialization point taken. If this is the case, then the function f is continuous and bounded. Moreover, if the function $f(x_1, x_2, ..x_n)$ is pseudoconvex in (x_k, x_i) for every $i, k \, \epsilon \, [1, 2, ..N]$, and if the function is regular at every $x \, \epsilon \, X^0$ then every cluster point x^r is a stationary point of f. A function is regular when the lower directional derivative $f'(x, d) >= 0 \forall d \, \epsilon \, d_1, ...d_k, ..d_N$. For a function f to be regular, the $dom(f_0)$ should be open and must be gateaux differentiable. Here, we imply the condition for regularity by taking into account of the smoothness of f_0. Even if the other seperable non-differentiable functions $f_k, k = 1, 2, ..N$ are not smooth, the claim for regularity holds valid.

1.3 Examples on convergence

For example, consider the function on Fig. 1. Here, the function is convex and differentiable, therefore at the global minimizer, I know that the gradient of the function with respect to all the coordinates would be zero. So, Coordinate descent computes a point that minimizes the function along each coordinate axis. Therefore, the coordinate descent Algorithm shall converges to the global minima.

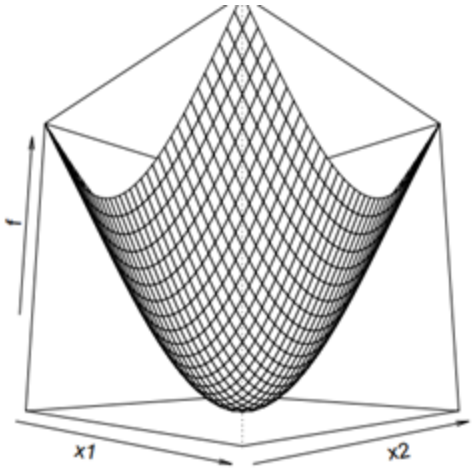

Figure 1: A convex differentiable function

Consider the function represented on the Fig. 2. Here, the function is convex , non-smooth and my minimizer is denoted by the blue dot. Let's say that my algorithm is at the point of intersection of red dotted lines. According to Coordinate Descent Algorithm, in an iteration, I would try to minimize along one of the coordinate axes. But here, when I am stuck at a non-stationary point minimizing the function along either of the axis returns the same stationary point which is obviously not a minimizer. So, the Coordinate Descent Algorithm gets stuck at this non-stationary point. One solution would be to minimize along both the axis simultaneously to find out the minimizer. This means that the non-differentiable part of the convex function is not separable. However, the function represented in 3 has a separable non-differentiable part. Coordinate descent algorithm works in the function to find the minimization point

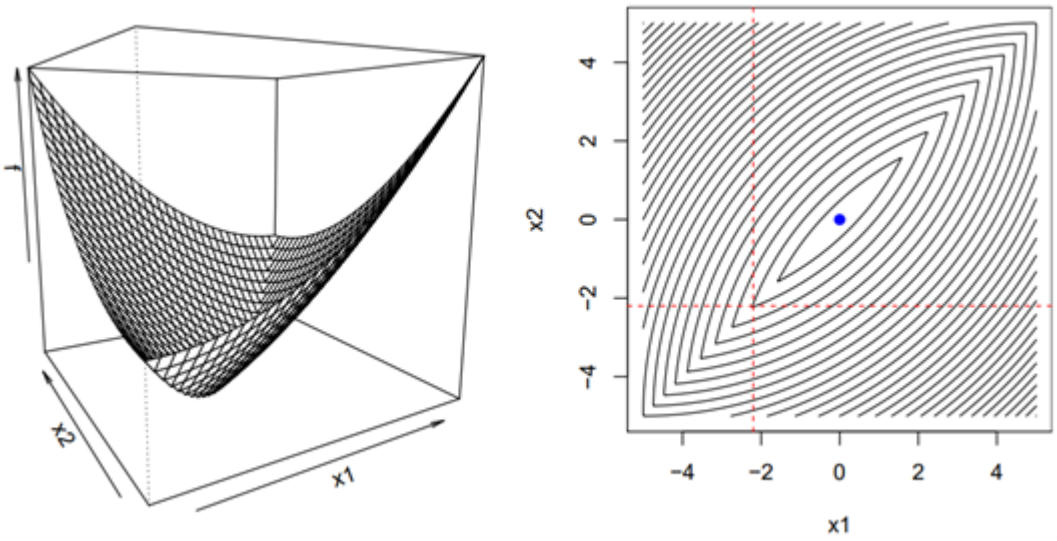

Figure 2: A convex non-differentiable function

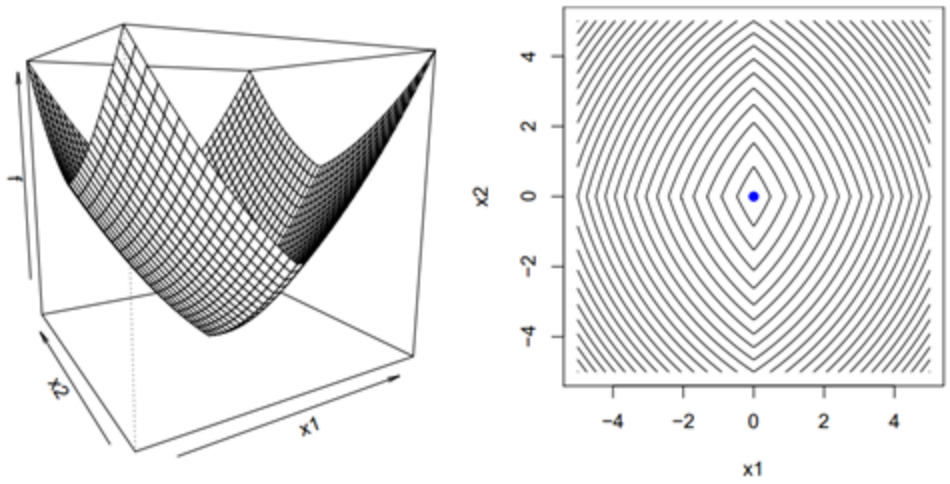

Figure 3: A convex separable non-differentiable function

2 Dykstra's Projection Algorithm

Dykstra's Projection Algorithm is a method for solving the best approximation problem given in Eqn. 3 where $C_1 \cap C_2 ... \cap C_n$ denote the intersection of convex sets. Initially one is unaware of the sets of points contained in the intersection of these convex sets. So, there is an iterative approach followed where the point x is projected from one convex set to an other convex set till the minimum is achieved. This algorithm differs from the Alternating Projections in a way that this algorithm tracks the residual after each projection onto $C_1 \cap C_2 .... \cap C_n$. Meanwhile, the Alternating projection's provides a point from the feasible set of points lying in the intersection, the Dykstra's Algorithm not only provides a point from the feasible set but it also provides the solution in the intersection of convex sets which is near to the point in question.

$$min_x ||y - x||_2^2 \; given \; x \; \epsilon \; C_1 \cap C_2 ... \cap C_n \tag{3}$$

2.1 Algorithm

First, we initialize $u(0)^d = y, z(0)^1 = \; = z(0)^d = 0$, and then repeat, for $k = 1, 2, 3, ..$

$$\begin{aligned} u_0^{(k)} &= u_d^{(k-1)} \\ u_i^{(k)} &= P_{C_i}(u_{i-1}^{(k)} + z_i^{(k-1)}) \\ z_i^{(k)} &= u_{i-1}^{(k)} + z_i^{(k-1)} - u_i^{(k)} \\ \forall i &= 1, 2, ...d \end{aligned} \tag{4}$$

where $P_{(Ci)}(x)$ denotes the convex projection of x onto the closed convex set C_i. Now, the residuals along with the projections are tracked in a cyclic order until the best approximation problem reaches its minimum.

3 LASSO

3.1 Primal Solution of LASSO

Consider the problem of Regularized regression problem in Eqn. 5 where the penalty is the L1 norm of the feature vector beta. Then the problem becomes a LASSO problem. LASSO is widely used because of the sparse solutions they return. The goal of the penalty function here, with respect to the

hyperparameter λ, is to minimize the L1 norm of the solution. So, trying to minimize the L1 norm, one encounters a sparse vectors. L1 norm has this capability because of its diamond shape.

$$\begin{aligned} min_{\beta \, \epsilon \, R^p}(1/2) * ||y - X\beta||_2^2 + \lambda \sum_{k=1}^{N} ||\beta||_1 \\ where \, \beta \, \epsilon \, R^{p*1} \, is \, a \, feature \, vector \\ X \, \epsilon \, R^{N*p} \, is \, a \, transformation \, matrix \\ y \, \epsilon \, R^{N*1} \, is \, a \, observation \, vector. \end{aligned} \tag{5}$$

3.2 Dual of LASSO

The dual of the LASSO can be represented as the best approximation problem where the original objective function is represented as follows:

$$\begin{aligned} max_\theta \, (1/2) * (||y||_2^2 - ||y - \theta||_2^2) \, subject \, to \, ||Xi^T\theta|| <= lambda \\ where \, i \, varies \, from \, 1 \, to \, p. \end{aligned} \tag{6}$$

This problem can be converted since I know that $||y||_2^2$ is a constant term and the negative sign can be treated as minimize the objective function.

$$\begin{aligned} min_\theta \, (1/2) * (||y - \theta||_2^2) \, subject \, to \, ||Xi^T\theta|| <= lambda \\ where \, i \, varies \, from \, 1 \, to \, p. \end{aligned} \tag{7}$$

3.3 Relationship between primal and dual solutions

Suppose that hi, i = 1, . . . , n are seminorms, which we can express in the general form $hi(v) = max \, \epsilon \, Di < hd, vi >$, where $Di \subseteq R^{p_i}$ is a closed, convex set containing 0, for i = 1, . . . , d. Suppose also that $Ci = (X_i^T)^1(D_i) = v \, \epsilon \, R^n : X_i^T v \, \epsilon \, D_i$, the inverse image of D_i under the linear mapping X_i^T, for i = 1, . . . , d...

Equation wise, the abover statement could be written in the following form where we assume that $X_i \, \epsilon \, R^{n*p_i}$ is a full column rank matrix and $h_i(v) = max_{d \, \epsilon \, D_i} < d, v >$ for a closed convex set containing 0. Then for $Ci = (X_i^T)^1(D_i) \subseteq R^n$ and any $b \, \epsilon \, R^n$,

$$\begin{aligned} \hat{w_i} = argmin_{w_i \, \epsilon \, R^{p_i}} = (1/2) * ||b - X_i w_i||_2^2 + h_i(w_i) \\ <=> \, \hat{w_i} = (Id - P_{C_i})(b) \end{aligned} \tag{8}$$

The best approximation problem and regularized regression problem are dual to each other which satisfy $\hat{u} = y - X\hat{w}$. The Dykstra's algorithm and Coordinate Descent Algorithm are equivalent and satisfy each other at all iterations k = 1,2,3.. via duality. From Eqn. 2 and 4, these iterations can be equated as

$$\begin{aligned} z_i^{(k)} = X_i\beta_i^{(k)} \, and \, u_i^{(k)} = y - \sum_{j<i} X_j\beta_j^{(k-1)} - \sum_{j>i} X_j\beta_j^{(k-1)} \\ where \, i = 1, 2, 3...d \end{aligned} \tag{9}$$

3.4 Convergence on LASSO

From Eqn. 5, we know that the sequences defined by the Coordinate Descent Algorithm are defined and bounded. If the function f is regular, then every cluster point from each of the iteration is a stationary point. Now, for f to be a regular function, the $dom(f_0)$ should be open and f_0 should be gateaux differentiable. The f_0 in this problem is differentiable and the $dom(f_0) \, \epsilon \, R^n$ is open. Therefore, the Theorem 4.1a proposed by Tseng is satified and the LASSO converges to a co-ordinate wise minimum point.

3.5 Solution to primal problem using Coordinate Descent

For the Co-ordinate descent algorithm to provide stationary points at each iteration, we note that the non-differentiable L1 norm is separable from the differentiable convex function. Therefore, we can perform minimization with respect to each of the coordinate axis.

Minimizing over β_i, with β_j, $j \neq i$:

$$0 = X_i^T(X_i\beta_i) + X_i^T(X_{-i}\beta_{-i} - y) + \lambda s_i$$

where $s_i \, \epsilon \, \partial|\beta_i|$. The solution is given by soft-thresholding

where

$$beta_i = S_{\lambda/||X_i||_2^2}(X_i^T(y - X_i\beta_i)/(X_i^T * X_i)) \tag{10}$$

The algorithm here is repeated for i = 1,2...p, 1,2...

The soft-thresholding operator is denoted by

$$\begin{aligned} S_\lambda(x) = x - \lambda \, if \; x > \lambda \\ x + \lambda \, if \; x < \lambda \\ 0 \, if \; x = \lambda \end{aligned} \tag{11}$$

The soft-thresholding operator is invoked here because of the non-differentiability of the L!-norm at β equals to zero. The sub-differential concept where the derivative of L1 norm can take any value between -1 to 1 at the non-stationary point of L1 norm is assumed and respectively used as the soft-thresholding operator.

3.6 Solution to dual problem using Dykstra's Projection Algorithm

The dual problem problem is solved by Dykstra's projection algorithm from Eqn. 4 where each of the closed convex set is denoted by $||X_i^T||_1 < lambda$ where i = 1,2, ... p (number of features). The projection of a point towards a convex set is solved by a convex solver. The relationship obtained between the Coordinate Descent Algorithm and the Dykstra's Algorithm is that

$$\theta \; = y - X\beta$$

where θ is obtained from the Dykstra's Algorithm and β is obtained from the Coordinate Descent.

4 Numerical Results

To evaluate the performance of the Algorithm, three separate cases are tested - Variation with respect to the number of observations (N), variation with respect to the sparsity (s), Variation with respect to the number of features (p). The test parameters here are - mean squared error and the support recovery performance. Mean square error denotes the square of the difference between the original feature vector beta and the predicted beta. Support recovery performance denotes the accuracy with which the predicted beta vector has predicted the features of least importance (the features with zero weights).

4.1 Variation of N

In this experiment, the Value of p is fixed at 100 and the number of non-zero elements in the sparse vector s is fixed at 4. The number of observations is varied from 20 to 200 in steps of 10. The error is relatively small considering which suggests that for a number of observations as small as 20, the sparse vector is recovered sufficiently as seen in Fig. 4. Therefore, in the next experiment, N was varied from 2 to 24 in steps of 2 as seen in the section 4.1.2. Here, we see that for a number of observations as small as the number of sparse elements, the error is high. Relative to the error is the support recovery performance. Here, the predicted feature vector should predict the features of least importance. As we see from Fig. 5 and Fig. 7, the support recovery performance is greater than 98 percent with 100 percent recovery for N values larger than 20.

4.1.1 Larger Values of N

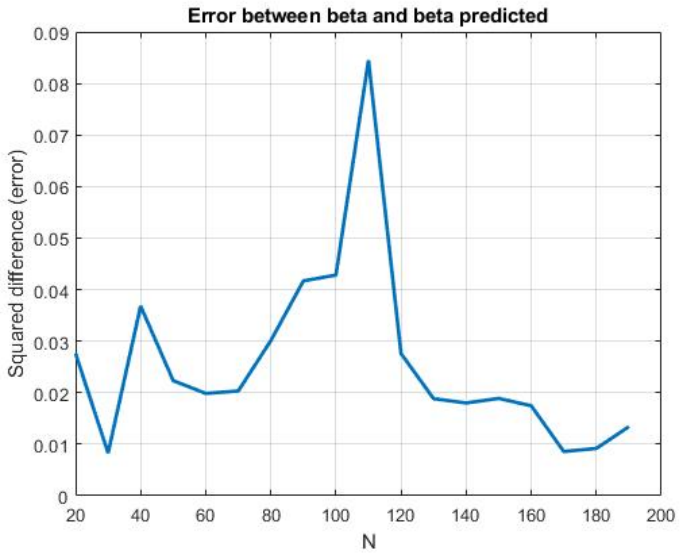

Figure 4: L2 error between the predicted feature vector and original feature vector

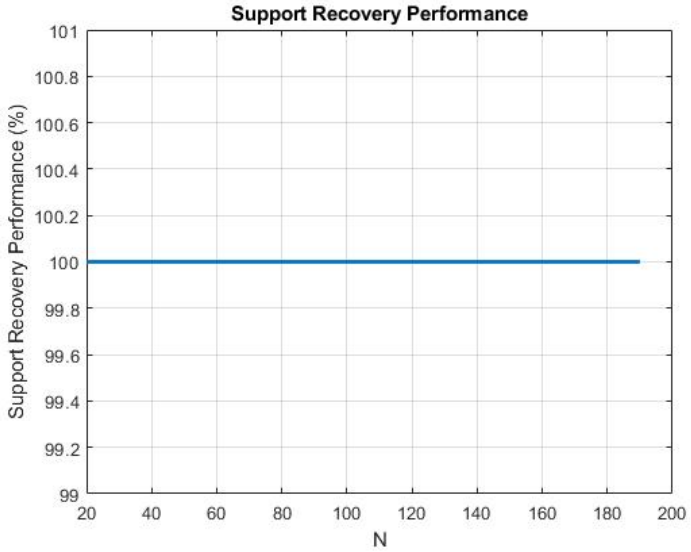

Figure 5: Support Recovery Performance for larger N

4.1.2 Smaller values of N

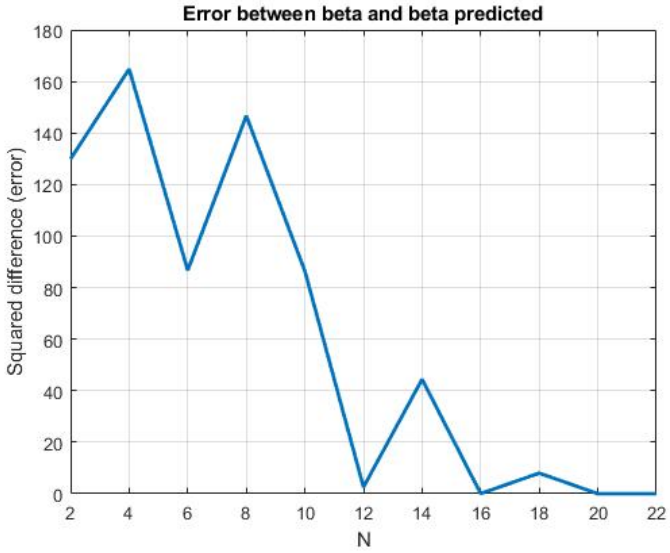

Figure 6: Mean squared error between the predicted feature vector and original feature vector

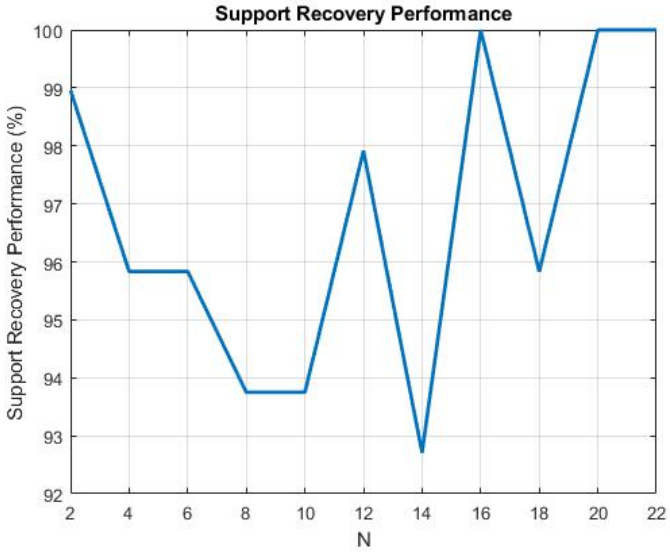

Figure 7: Support Recovery Performance for smaller N

4.1.3 Runtime Performance

One of the information that I can get is how much Runtime is required by the algorithm for processing a large number of observations. The CVXPY with highly optimized python module performs 1000 times better than the current implementation of Co-ordinate Descent Algorithm. The solver used by the CVXPY module is QSQP solver. 8 shows the runtime of Co-ordinate Descent Algorithm across various values of N and 9 shows the runtime of equivalent problem executed by CVXPY module.

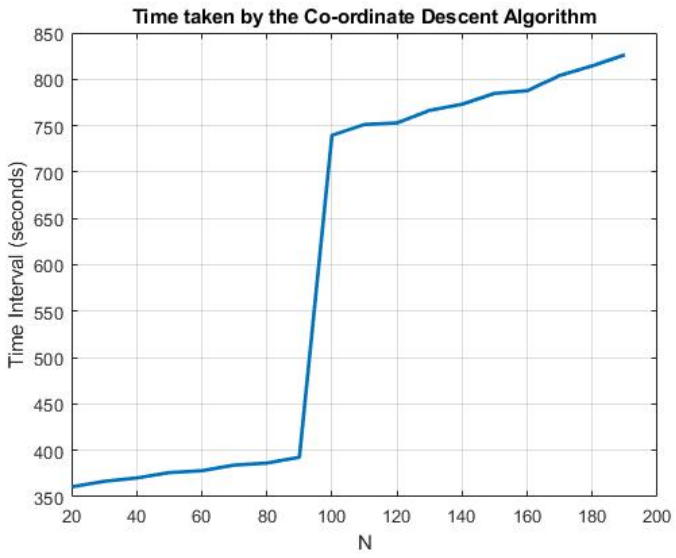

Figure 8: Runtime evaluation of Co-ordinate Descent Algorithm

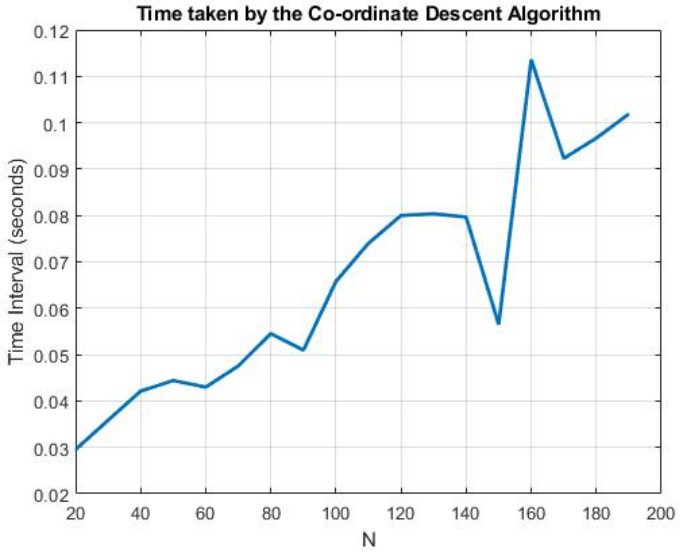

Figure 9: Runtime evaluation of CVXPY module

4.2 Variation of sparsity N

In this experiment, N is fixed at 50 and p is fixed at 100. The number of non-zero elements is varied from 5 to 90 in the steps of 5. The whole point in this experiment is to figure out, given the setting, the features and the observations, we are trying to get an idea on how the support recovery performance is based on the sparsity.

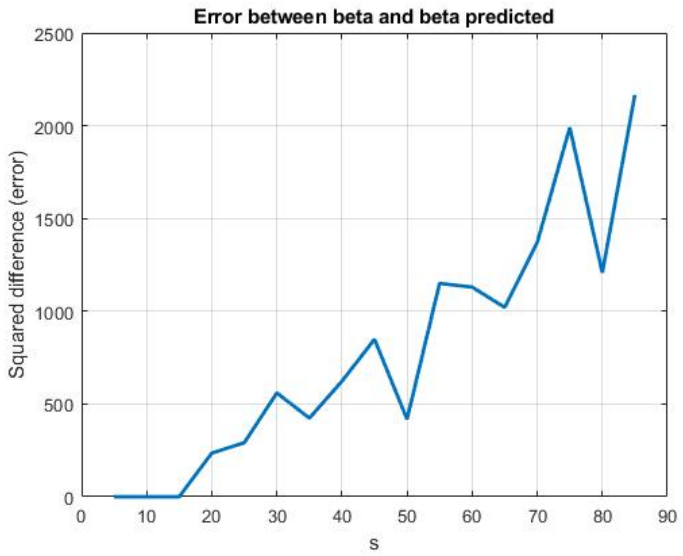

Figure 10: L2 Error between beta and beta predicted

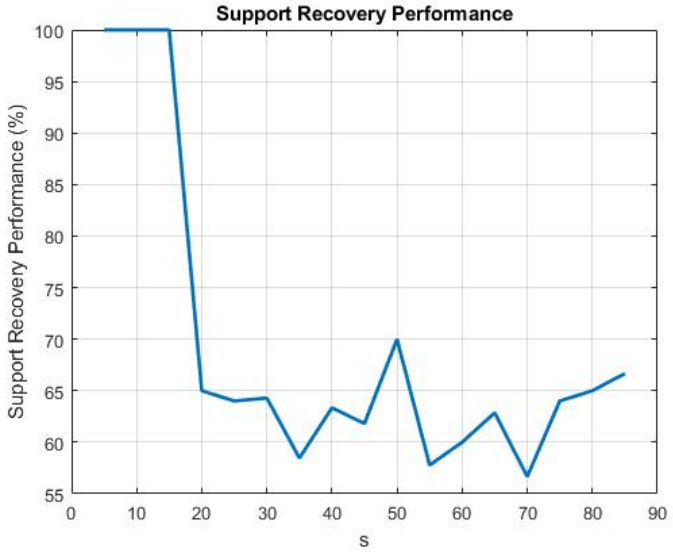

Figure 11: Support Recovery Performance in terms of sparsity

4.3 Variation of number of features p

In this experiment, p is varied from 100 to 300 whereas N is fixed at 50 and the number of non-elements in the sparse feature vector is fixed at 8. The support recovery performance as p increases is around 90 % whereas the error performance is fixed across all the elements over p and no trend is seemed to be observed. Fig. 12 shows the L2 error wrt the number of features and Fig. 13 shows the support recovery performance.

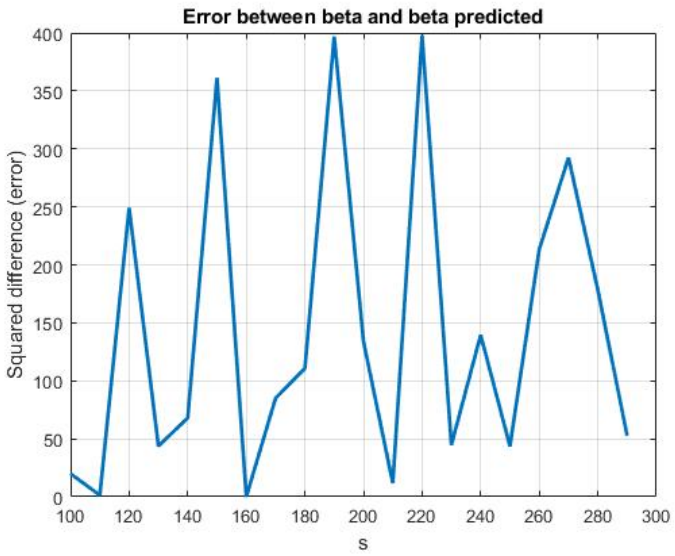

Figure 12: L2 Error between beta and beta predicted in terms of number of features

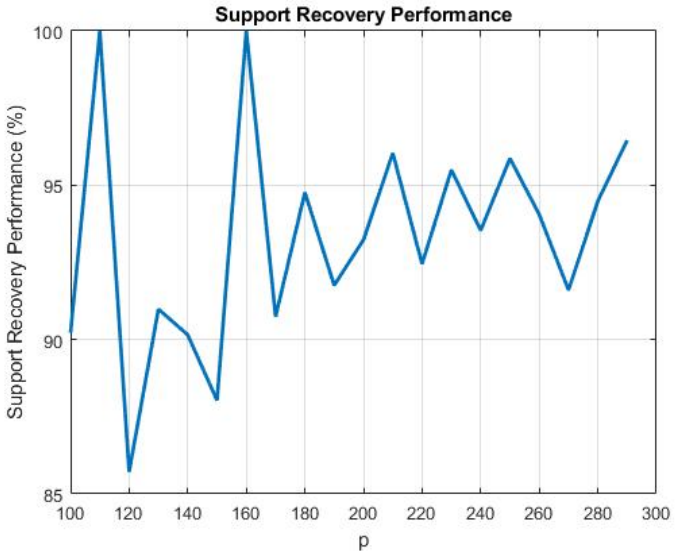

Figure 13: Support Recovery Performance in terms of number of features

4.4 Variation of N and s where s is varied linearly with N

In this experiment, the value of N is varied from 20 to 200 as well as the sparsity simultaneously from 4 to 40. The trend we see in this experiment is that with increasing number of observations, we find the the number of non-zero elements as well increases. This can be seen in the support recovery performance where we have mapped N to a sparsity such that for 20 number of observations, my sparsity feature vector would have 4 sparse elements and at 200 number of observations, my sparse vector would have 40 non-zero elements. It is seen that even with the increase of number of non-zero elements, my support recovery performance doesn't reduce with increase in number of observations as seen in Fig. 15.

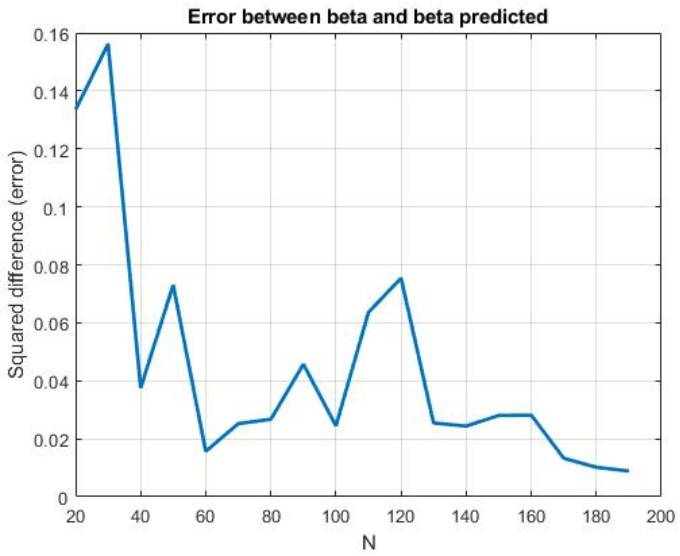

Figure 14: L2 Error between beta and beta predicted in terms of number of features

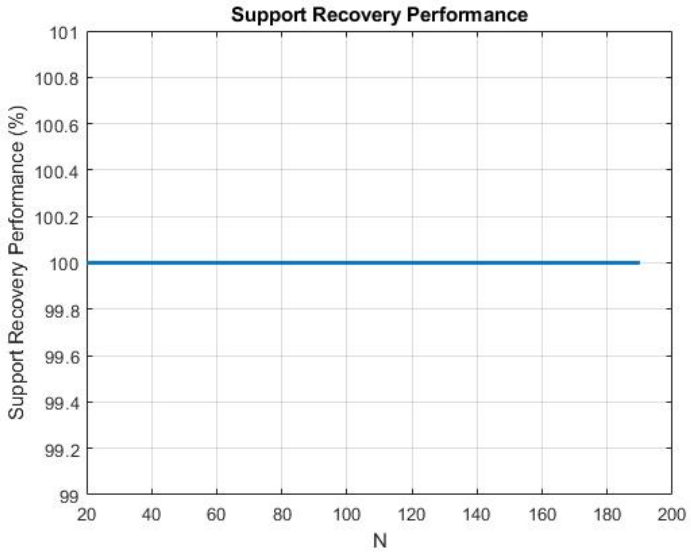

Figure 15: Support Recovery Performance in terms of number of observations

5 Applications to Co-ordinate Descent Algorithm

One of the areas where Co-ordinate Descent could be applied in wireless communications are computing channel capacity and Antenna Array synthesis. In this project, a basic Antenna array synthesis using Co-ordinate Descent is observed. So, in the Antenna Array Synthesis problem, we are given some constraints on the radiation pattern and number of antennas. We need to figure not the complex weights associated with each of the Antenna element to achieve the specification put forth by the radiation pattern. Now, the specification can be location of nulls, the level of side lobes and the direction of main lobes. So, in this project, a simple linear array synthesis based on Dolph-Chebyshev array weights is tried. The constraint here is that the number of Antenna Elements is 8 and the minimum side lobe level achieved is 30dB lesser than the main lobe. Fig. .16 shows such radiation pattern.

Now, I have radiation pattern with respect to θ from -90 to 90 degrees. Given the condition of the number of Antenna Elements and spacing between the elements, I would be able to find out the transformation matrix H which translates the weight vector to the radiation pattern. Each element in the matrix H is given by

$$H(theta, n) = exp(1i * (2 * pi * spacing/lambda) * sin(theta) * n)$$

where lambda is the resonant wavelength of the Antenna, n is the antenna element and theta is the angle in radians.

We can see that H is a full row rank matrix. Therefore, the problem can be treated as a least squares problem where the Coordinate Descent iterations are same as LASSO iterations except that lambda is zero here and all the elements are in complex space. For as few iterations as lambda, the weight vector predicted by the Coordinate Descent Iterations matched the radiation pattern in the constraints.

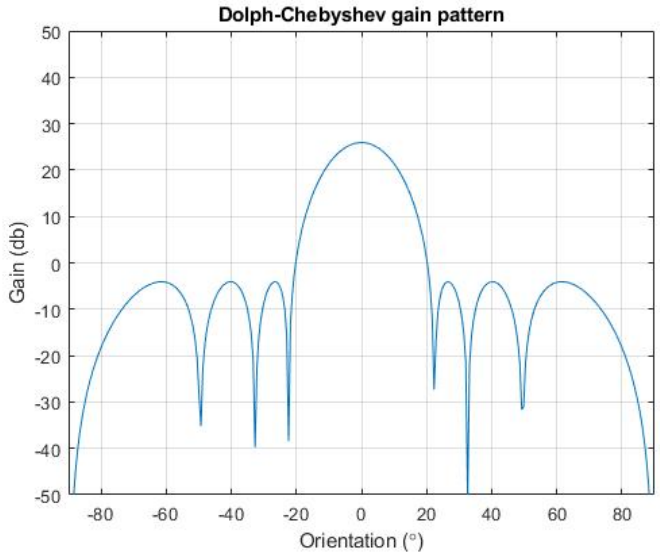

Figure 16: Dolph-Chebyshev weights based radiation pattern of Linear Array

References

[1] https://stats.stackexchange.com/questions/123672/coordinate-descent-soft-thresholding-update-operator-for-lasso

[2] https://www.cs.cmu.edu/ ggordon/10725-F12/slides/25-coord-desc.pdf

[3] Tibshirani, R. J. (2017). Dykstra's algorithm, admm, and coordinate descent: Connections, insights, and extensions.

[4] Tseng, P. (2001). Convergence of a block coordinate descent method for non differentiable minimization.

[5] https://blog.mlreview.com/l1-norm-regularization-and-sparsity-explained-for-dummies-5b0e4be3938a

[6] Wang, Y. , He, X. , Wang, J. , Berezin, S. and Mathis, W. (2015) Antenna Array Pattern Synthesis via Coordinate Descent Method. Journal of Electromagnetic Analysis and Applications, 7, 168-177. doi: 10.4236/jemaa.2015.75018.