Project: 02953 Convex Optimization

Anton Ruby Larsen [s174356]

May 13, 2023

Contents

1 Introduction

In this project we will implement the alternating direction of multipliers method(ADMM) for the l_1-reguralized least squares problem also known as LASSO. We will implement a serial and a parallel version and test the algorithm on some problems. First we will describe the theory behind the general framework of ADMM and then specialize it for the LASSO problem. The theory will be based on the paper [1] and the lecture notes [2]. Afterwards the results will be presented.

2 Theory

Before diving into ADMM we will motivate the method by reviewing the dual and the method of multipliers.

2.1 Dual Method

Consider the problem

$$\min_x f(x) \quad \text{subject to } Ax = b \tag{1}$$

where we assume f is strictly convex and closed. The Lagrangian is the given by

$$\mathcal{L}(x, \lambda) = f(x) + \lambda^T (Ax - b)$$

The dual method then performs dual gradient ascent by repeating the following two steps

$$x^{(k+1)} = \operatorname*{argmin}_x \mathcal{L}\left(x, \lambda^{(k)}\right) \tag{2a}$$

$$\lambda^{(k+1)} = \lambda^{(k)} + \eta_k \left(Ax^{(k+1)} - b\right) \tag{2b}$$

If f is decomposable, then 2a can be decomposed and solved in parallel. One huge problem with the method is that f must be strictly convex. To relax this constraint we introduce the method of multipliers.

2.2 Method of multipliers

We introduce the augmented Lagrangian which alters the objective of 1.

$$\min_x f(x) + \frac{\rho}{2}\|Ax - b\|_2^2 \quad \text{subject to } Ax = b \tag{3}$$

The extra addition to the objective makes the program strongly convex as long $f(x)$ is convex. Furthermore we also observe that the extra term will only change the objective when 3 is infeasible and hence do not alter the solution. To optimize the program we follow the same steps as in the Dual method. The downside with this method is that even though $f(x)$ is decomposable we cannot decompose the problem due to $\|Ax - b\|_2^2$ entangles everything. To combine the best of the two methods we introduce the alternating direction of multipliers method.

2.3 Alternating direction of multipliers method

ADMM splits the problem into a problem of two variables

$$\min_x f(x) + g(z) \quad \text{subject to } Ax + Bz = c \tag{4}$$

where $f(x)$ and $g(z)$ are convex functions and $\rho > 0$. This give the Lagrangian

$$\mathcal{L}_\rho(x, z, \lambda) = f(x) + g(z) + \lambda^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 \tag{5}$$

We again optimize by dual ascent but now we have three steps due to the extra variable.

$$x^{(k+1)} = \operatorname*{argmin}_x \mathcal{L}_\rho\left(x, z^{(k)}, \lambda^{(k)}\right) \tag{6a}$$

$$z^{(k+1)} = \operatorname*{argmin}_z \mathcal{L}_\rho\left(x^{(k+1)}, z, \lambda^{(k)}\right) \tag{6b}$$

$$\lambda^{(k+1)} = \lambda^{(k)} + \rho\left(Ax^{(k+1)} + Bz^{(k+1)} - c\right) \tag{6c}$$

We often see the method in what is called scaled form. We obtain the scalled form by substituting λ with $\mu = \frac{\lambda}{\rho}$. This abbreviates especially 6a and 6b due to the fact

$$\lambda^T(AxBz - c + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 = \frac{\rho}{2}\|Ax + Bz - c + \mu\|_2^2 - \frac{\rho}{2}\|\mu\|_2^2$$

This gives us the scaled form of the dual ascent

$$x^{(k+1)} = \operatorname*{argmin}_x f(x) + \frac{\rho}{2}\left\|Ax + Bz^{(k)} - c + \mu^{(k)}\right\|_2^2 \tag{7a}$$

$$z^{(k+1)} = \operatorname*{argmin}_z g(z) + \frac{\rho}{2}\left\|Ax^{(k+1)} + Bz - c + \mu^{(k)}\right\|_2^2 \tag{7b}$$

$$\mu^{(k+1)} = \mu^{(k)} + Ax^{(k+1)} + Bz^{(k+1)} - c \tag{7c}$$

2.3.1 Convergence

Necessary and sufficient conditions for the ADMM method are primal feasibility

$$Ax^* + Bz^* - c = 0 \tag{8}$$

and dual feasibility

$$0 \in \partial f\left(x^*\right) + A^T\mu^* \tag{9a}$$

$$0 \in \partial g\left(z^*\right) + B^{T*} \tag{9b}$$

The primal feasibility we calculate for every iterations as

$$r^{(k+1)} = Ax^{k+1} + Bz^{k+1} - c \tag{10}$$

It is not straight forward to calculate the dual feasibility from 9a and 9b but in section 3.3 in [1] it is shown that we can calculate the dual feasibility as

$$s^{k+1} = \rho A^T B \left(z^{k+1} - z^k\right) \tag{11}$$

Further they state in [1] that a primal and dual feasibility tolerance are given by

$$\begin{aligned} \epsilon^{\text{pri}} &= \sqrt{p}\epsilon^{\text{abs}} + \epsilon^{\text{rel}} \max\left\{\left\|Ax^k\right\|_2, \left\|Bz^k\right\|_2, \|c\|_2\right\} \\ \epsilon^{\text{dual}} &= \sqrt{n}\epsilon^{\text{abs}} + \epsilon^{\text{rel}} \left\|A^T\mu^k\right\|_2 \end{aligned} \tag{12}$$

Lastly they recommend ϵ^{rel} to be in $[10^{-3}, 10^{-4}]$ and we can choose ϵ^{abs} freely.

2.3.2 Consensus ADMM

If our problem can be split up into B parts we can write it as

$$min_x \sum_{i=1}^{B} f_i(A_i^T x + b_i) + g(x) \tag{13}$$

To use ADMM we introduce a variable z and a constraint.

$$\min_{x_1,\dots x_B,z} \sum_{i=1}^{B} f_i\left(A_i^T x_i + b_i\right) + g(z) \text{ subject to } x_i = z, i = 1,\dots B \tag{14}$$

We can now write out the ADMM steps

$$x_i^{k+1} = \underset{x_i}{\text{argmin}}\ f_i\left(x_i\right) + \mu_i^{kT}\left(x_i - z^k\right) + (\rho/2)\left\|x_i - z^k\right\|_2^2 \tag{15a}$$

$$z^{k+1} = \underset{z}{\text{argmin}}\ g(z) + \sum_{i=1}^{B}\left(-\mu_i^{kT} z + (\rho/2)\left\|x_i^{k+1} - z\right\|_2^2\right) \tag{15b}$$

$$\mu_i^{k+1} = \mu_i^k + \rho\left(x_i^{k+1} - z^{k+1}\right) \tag{15c}$$

The primal and dual feasibility we can calculate by

$$\left\|r^k\right\|_2^2 = \sum_{i=1}^{N}\left\|x_i^k - \bar{x}^k\right\|_2^2, \quad \left\|s^k\right\|_2^2 = N\rho^2\left\|\bar{x}^k - \bar{x}^{k-1}\right\|_2^2 \tag{16}$$

2.4 LASSO

We now want to apply ADMM to the l_1-regularized least squares problem also known as LASSO

$$\min_x \|Ax - b\|_2^2 + \tau\|x\|_1 \tag{17}$$

We recognize

$$f(x) = \|Ax - b\|_2^2 \qquad g(z) = \tau\|z\|_1$$

So we can write 17 as

$$\min_{x,z} f(x) + g(z) \quad \text{ subject to } x = z \tag{18}$$

We formulate the x-step in the dual ascent

$$x^{(k+1)} = \underset{x}{\operatorname{argmin}} \ \|Ax - b\|_2^2 + \frac{\rho}{2} \left\| x - z^{(k)} + \mu^{(k)} \right\|_2^2 \tag{19}$$

This is equivalent to the least squares problem

$$\min_x \left\| \begin{bmatrix} A \\ \sqrt{\rho} I \end{bmatrix} x - \begin{bmatrix} b \\ \sqrt{\rho}(z^{(k)} - \mu^{(k)}) \end{bmatrix} \right\|_2^2 \tag{20}$$

with the closed form solution

$$\begin{aligned} x^{(k+1)} &= (A^T A + \rho I)^{-1} \begin{bmatrix} A^T & \sqrt{\rho} I \end{bmatrix} \left[b(z^{(k)} - \mu^{(k)}) \right] \\ &= (A^T A + \rho I)^{-1} (A^T b + \rho(z^{(k)} - \mu^{(k)})) \end{aligned} \tag{21}$$

Next we have the z-step.

$$\begin{aligned} z^{k+1} &= \underset{z}{\operatorname{argmin}} \ \tau \|z\|_1 + \frac{\rho}{2} \left\| z - x^{(k+1)} - \mu^{(k)} \right\|_2^2 \\ &= \underset{z}{\operatorname{argmin}} \ \frac{\tau}{\rho} \|z\|_1 + \frac{1}{2} \|z - v\|_2^2 \end{aligned} \tag{22}$$

We split 22 into N problems

$$z_n = \underset{z_n \in R}{\operatorname{argmin}} \ h(z_n) = \frac{\tau}{\rho} |z_n| + \frac{1}{2}(z_n - v_n)^2 \tag{23}$$

23 is convex and differentiable everywhere except at $z = 0$.

$$\frac{\mathrm{d}h}{\mathrm{d}z} = \begin{cases} \frac{\tau}{\rho} + z - v, & z > 0 \\ -\frac{\tau}{\rho} + z - v, & z < 0 \end{cases} \tag{24}$$

We see from 24 that for the optimal solution z^* to be positive $\frac{\tau}{\rho} + z - v = 0$, hence it must hold that $v > \frac{\tau}{\rho}$. Similarly if z^* is negative then $-\frac{\tau}{\rho} + z - v = 0$, and hence $v < -\frac{\tau}{\rho}$.If neither $v < -\frac{\tau}{\rho}$ or $v > \frac{\tau}{\rho}$ then z^* must equal zero. 22 can hence be solved by

$$\begin{aligned} z^* &= \begin{cases} v - \frac{\tau}{\rho}, & v > \frac{\tau}{\rho} \\ 0, & |v| \leq \frac{\tau}{\rho} \\ v + \frac{\tau}{\rho}, & v < -\frac{\tau}{\rho} \end{cases} \\ &= S_{\frac{\tau}{\rho}}(v) \end{aligned} \tag{25}$$

where 25 is the proximal operator known as the shrinkage operator. Lastly we have the μ-step which is given by

$$\mu^{(k+1)} = \mu^{(k)} + x^{(k+1)} - z^{(k+1)} \tag{26}$$

In algorithm 1 we have summarized the algorithm

Algorithm 1 An ADMM algorithm for the LASSO problem

1: **procedure** LASSOADMM(A, b, τ)
2: $x^{(k+1)} = (A^T A + \rho I)^{-1}(A^T b + \rho(z^{(k)} - \mu^{(k)}))$
3: $z^{(k+1)} = S_{\frac{\tau}{\rho}}(x^{(k+1)} + \mu^{(k)})$
4: $\mu^{(k+1)} = \mu^{(k)} + x^{(k+1)} - z^{(k+1)}$
5: **end procedure**

2.4.1 Distributed LASSO

If we are given data as

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_B \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_B \end{bmatrix} \tag{27}$$

and want to fit LASSO in parrallel we can apply the consensus framework. By doing so we obtain the following steps described at page 65 in [1].

$$x_i^{(k+1)} = (A_i^T A_i + \rho I)^{-1}(A_i^T b_i + \rho(z^k - \mu_i^k)) \tag{28a}$$
$$z^{(k+1)} = \mathrm{S}_{\tau/(\rho B)}(\bar{x}^{k+1} - \bar{\mu}^{(k)}) \tag{28b}$$
$$\mu_i^{(k+1)} = \mu_i^{(k)} + x_i^{(k+1)} - z^{(k+1)} \tag{28c}$$

where $\bar{x}^{k+1} = \frac{1}{B}\sum_{i=1}^{B} x_i^{(k+1)}$ and $\bar{\mu}^{k+1} = \frac{1}{B}\sum_{i=1}^{B} \mu_i^{(k+1)}$

3 Results

We have implemented a LASSO and a distributed LASSO using the ADMM algorithm in matlab. The code can be found in the attached zip file. In the following we will test the implementations.

3.1 Serial LASSO

We have tested the algorithm on a problem with 3000 samples and 500 variables. We solve the problem using 50 different log spaced regularization values in the interval $[10^{-7}, 10^{1}]$. We see in figure 1 that when we increase τ the sparsity increased. This makes sense because the higher l_1-regularization the more our program is penalized by having many non-zero variables.

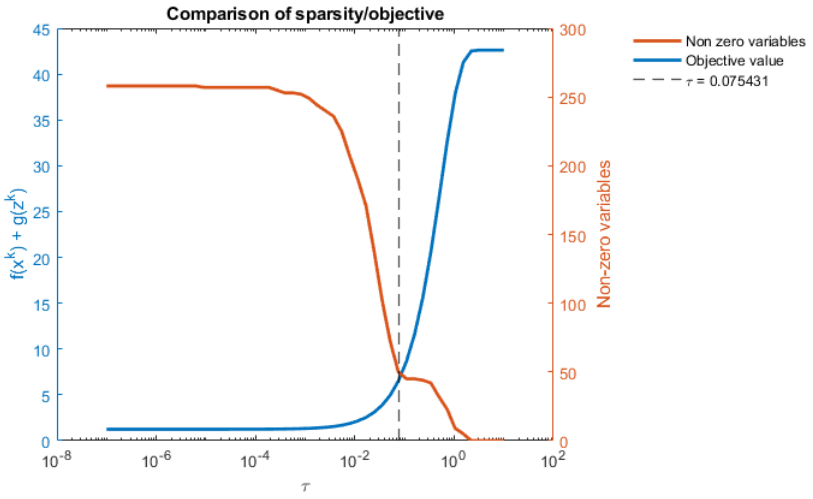

Figure 1: The relation between sparsity and objective value for different τ

The trade-off between sparsity and objective value we saw in figure 1 can be understood as a Pareto optimal set. We have plotted this relation in figure 2. We have selected the point of maximum curvature as an optimal τ. This τ is indicated with a dashed line.

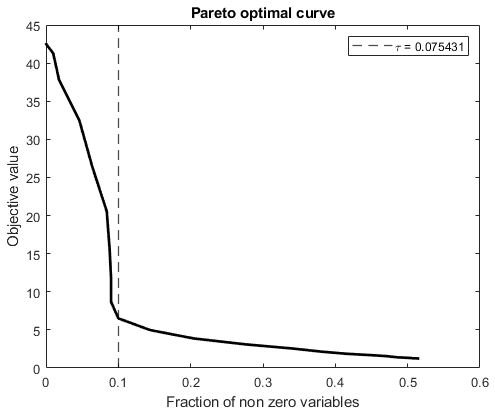

Figure 2: The Pareto curve for sparsity and objective value

Next we investigate how the ADMM algorithm converges under different values of τ. In figure 3 we investigate how the model behaves with $\tau = 10^{-7}$. We see that the algorithm is primal feasible from the start because practically no regularization is done and hence $z = x$ is trivially met. On the other hand our z variable must converge to a stable value for our program to be dual feasible as given in 11.

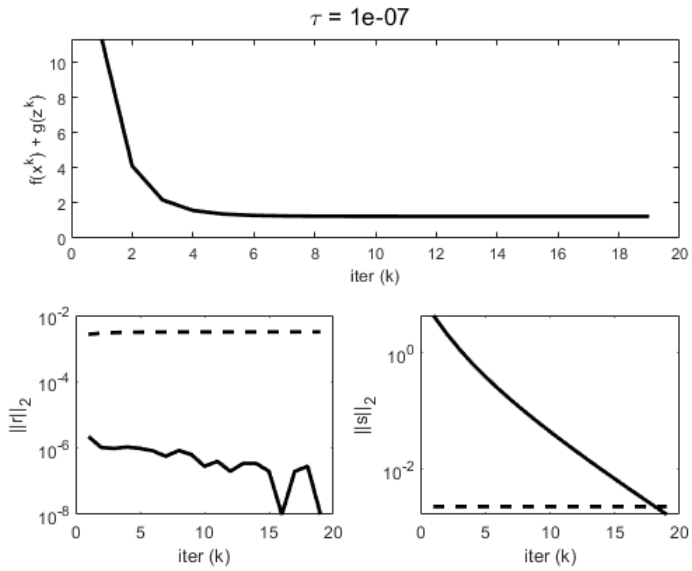

Figure 3: Convergence statistics for $\tau = 10^{-7}$

On the other end of the spectrum we have a heavily regularized program with $\tau = 10$. The convergence for such a program is plotted in figure 4. Here we see the opposite picture where the program is dual feasible from the start due to z being set to all zero in every iteration. The primal feasibility is then converging as x goes towards all zeros.

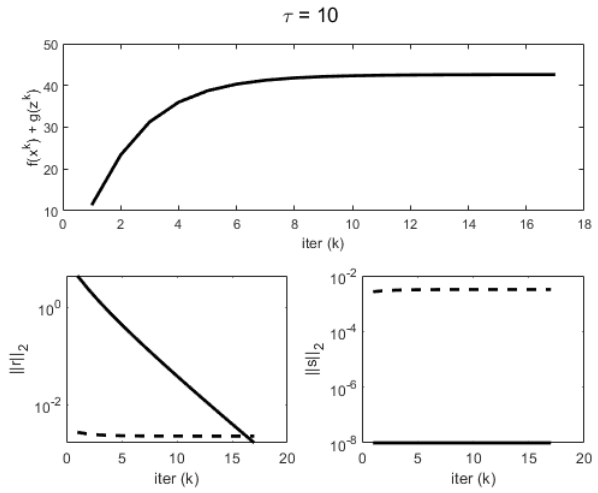

Figure 4: Convergence statistics for $\tau = 10^1$

Lastly we have the case where $\tau = 0.075431$ which is the optimum described earlier. The convergence for this value is plotted in figure 5. Here we see that the dual and primal feasibility converges simultaneously. We also observe that the objective value converges to a slightly higher value than in figure 3 due the sparsity of the solution but much lower than in figure 4 because some variables are allowed to be non zero.

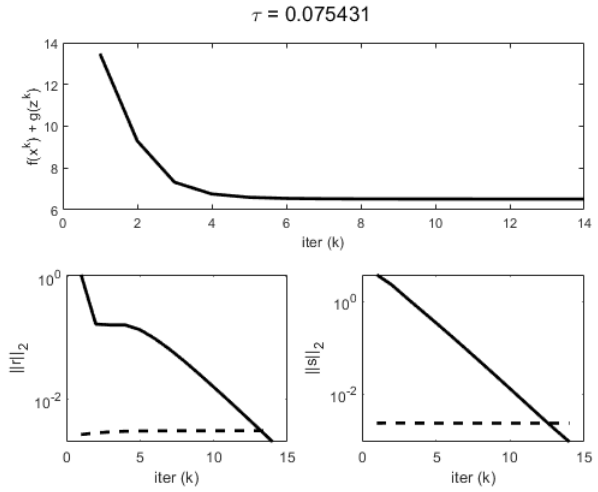

Figure 5: Convergence statistics for $\tau = 0.075431$

3.2 LASSO parallel

We have implemented the parallel framework in matlab and the setup is very slow due to inefficient data structures and little experience with parallel computing in matlab. We will therefore not compare the serial setup with the parallel because the serial is much faster. We will though compare the speed of the parallel framework using 1 to 8 workers. We see in figure 6 that the fastest is when using 4 workers which makes sense due to my laptop having 4 kernels.

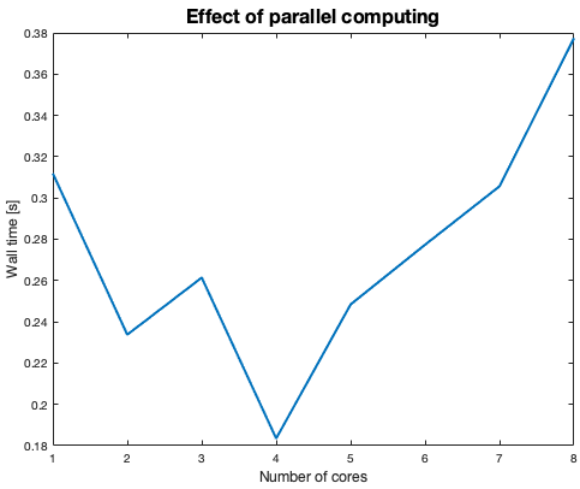

Figure 6: Wall time for different number of workers

4 Conclusion

We see that the ADMM implemented for LASSO works fine and the regularization has the wished effect. The parallel case works but not efficiently. In [1] they have in C implemented a distributed version of LASSO using ADMM. This version is much more efficient than ours so an improvement if more time was available was to implement our version in C.

Furthermore they also have a lot of tricks in [1] one can use to improve the ADMM algorithm. This is also something one could implement if more time was available.

References

[1] S. P. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers bibtex," January 2011.

[2] R. Tibshirani, "Lecture 21: November 14,"