Final Project (Robust PCA) - MATP 4820

Thomas Shweh - Eric Tran

Collaboration
Thomas Shweh (shweht - 661774756): Quadratic Penalty, Graphs, Write up
Eric Tran (trane2 - 661646071): Quadratic Penalty, ADMM, Write up

Quadratic Penalty Method

Given the constrained problem, we decided to implement a quadratic penalty method. Penalty methods replace a constrained optimization problem by a series of unconstrained problems; for this reason we decided to separate our original model into two subproblems for each update: **L** and **S**.

Subproblems
We can model the Robust PCA problem in the form

$$\min_{\mathbf{L},\mathbf{S}} \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 \qquad \text{s.t. } \mathbf{L}+\mathbf{S}=\mathbf{X}$$

We can develop and iteration of the problem in form of an accelerated proximal gradient of this problem. Hence we will need to solve subproblems in the form

$$\min_{\mathbf{L},\mathbf{S}} \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 + \frac{\beta}{2}\|\mathbf{L}+\mathbf{S}-\mathbf{Z}\|_F^2$$

Where β is the penalty parameter. But we are going to express the equation in terms of μ, a value that we were are going to use every iteration to accelerate the convergence. Thus our problem becomes

$$\min_{\mathbf{L},\mathbf{S}} \mu\|\mathbf{L}\|_* + \mu\lambda\|\mathbf{S}\|_1 + \frac{1}{2}\|\mathbf{L}+\mathbf{S}-\mathbf{Z}\|_F^2$$

With $\mathbf{X}=(\mathbf{L},\mathbf{S})$, we can separate the function into two components $h(\mathbf{X}) = \mu\|\mathbf{L}\|_* + \mu\lambda\|\mathbf{S}\|_1$ and $g(\mathbf{X}) = \frac{1}{2}\|\mathbf{L}+\mathbf{S}-\mathbf{Z}\|_F^2$. Since the nuclear norm is non differentiable we will work with the gradient of g because it is differentiable. So $\nabla g(\mathbf{X}) = (\nabla_{\mathbf{L}} g(\mathbf{X}), \nabla_{\mathbf{S}} g(\mathbf{X}))$ where

$\nabla_{\mathbf{L}} g(\mathbf{X}) = \nabla_{\mathbf{S}} g(\mathbf{X}) = \mathbf{L} + \mathbf{S} - \mathbf{Z}$. Now finding the Lipshitz constant we have

$$\begin{aligned}
\|\nabla g(\mathbf{X}_1) - \nabla g(\mathbf{X}_1)\| &= \left\| \begin{bmatrix} \mathbf{L}_1 + \mathbf{S}_1 - \mathbf{Z} \\ \mathbf{L}_1 + \mathbf{S}_1 - \mathbf{Z} \end{bmatrix} - \begin{bmatrix} \mathbf{L}_2 + \mathbf{S}_2 - \mathbf{Z} \\ \mathbf{L}_2 + \mathbf{S}_2 - \mathbf{Z} \end{bmatrix} \right\| \\
&= \left\| \begin{bmatrix} \mathbf{L}_1 - \mathbf{L}_2 + \mathbf{S}_1 - \mathbf{S}_2 \\ \mathbf{L}_1 - \mathbf{L}_2 + \mathbf{S}_1 - \mathbf{S}_2 \end{bmatrix} \right\| \\
&\leq \left\| \begin{bmatrix} \mathbf{L}_1 - \mathbf{L}_2 \\ \mathbf{S}_1 - \mathbf{S}_2 \end{bmatrix} \right\| + \left\| \begin{bmatrix} \mathbf{S}_1 - \mathbf{S}_2 \\ \mathbf{L}_1 - \mathbf{L}_2 \end{bmatrix} \right\| \\
&= 2\|\mathbf{X}_1 - \mathbf{X}_2\|
\end{aligned}$$

Which leaves us with a Lipschitz constant of 2. Now we can express each iteration in the general form of the proximal gradient. This form is

$$x^+ = \arg\min_u \left(h(u) + \frac{L}{2}\|u - G(y)\|^2 \right)$$

So since each iteration we are updating $\mathbf{L}, \mathbf{S}$ together we have

$$\begin{aligned}
\mathbf{L}_{k+1}, \mathbf{S}_{k+1} &= \arg\min_{\mathbf{X}} \left(h(\mathbf{X}) + \frac{L}{2}\|\mathbf{X} - G(Y_k)\|^2 \right) \\
&= \arg\min_{\mathbf{L},\mathbf{S}} \left(\mu\|L\|_* + \mu\lambda\|\mathbf{S}\|_1 + \left\| \begin{bmatrix} \mathbf{L} \\ \mathbf{S} \end{bmatrix} - \begin{bmatrix} G_k^{\mathbf{L}} \\ G_k^{\mathbf{S}} \end{bmatrix} \right\|^2 \right) \\
&= \arg\min_{\mathbf{L},\mathbf{S}} \left(\|L\|_* + \frac{1}{2\frac{\mu}{2}}\|\mathbf{L} - G_k^{\mathbf{L}}\|^2 + \mu\lambda\|\mathbf{S}\|_1 + \frac{1}{2\frac{\mu}{2}}\|\mathbf{S} - G_k^{\mathbf{S}}\|^2 \right)
\end{aligned}$$

Since the problem above is separable, we now have 2 problems that we can solve for. Essentially we can express the problems of finding $\mathbf{L}_{k+1}, \mathbf{S}_{k+1}$ in terms of the shrinkage operator. For $G_k^{\mathbf{L}} = U\Sigma V^T$ or the SVD, we can express each iteration as such

$$\begin{aligned}
\mathbf{L}_{k+1} &= U\mathcal{S}_{\frac{\mu}{2}}[\Sigma]V^T \\
\mathbf{S}_{k+1} &= \mathcal{S}_{\frac{\lambda\mu}{2}}[G_k^{\mathbf{S}}]
\end{aligned}$$

Our penalty term for our problem is, where $\bar{\mu}$ is the floor value for μ.

$$\frac{1}{2\bar{\mu}}\|\mathbf{L} + \mathbf{S} - \mathbf{Z}\|_F^2$$

Shrinkage Operator

The shrinkage operator is derived from the soft-thresholding rule found from Theorem 2.1 (referred to in the assignment/from LMS) which shows:

$$\mathcal{S}_{\tau\lambda} = \arg\min_x \frac{1}{2\tau}||X - Y||_F^2 + \lambda||X||_1$$

Because this expressions is convex, we understand that there is a unique minimizer, X, that can be found element-wise. Because of this we know that $Y_{r,c} - X_{r,c} \in \tau\lambda\partial|X_{r,c}|$ where r, c indicates the element. From here, we can reason upon inspection that when $X_{r,c} = 0 \to |Y_{r,c}| \le \lambda$ and $X_{r,c} \ne 0 \to \partial|X_{r,c}|$. This yields:

$$X_{r,c} = Y_{r,c} - \tau\lambda sign(Y_{r,c}) = \mathcal{S}_\tau$$

This is a summary of how we derived the shrinkage operator based on the paper from LMS (suggested in the assignment - Theorem 2.1) and our additional research (see references). This will be used for our proximal mapping.

Stopping Condition

We can derive the stopping conditions based off of the KKT conditions of the original problem. This will give us the primal and dual feasibility. The original KKT conditions of the problem are

$$\mathbf{L} + \mathbf{S} - \mathbf{Z} = 0 \qquad Y^* \in \partial\|L\|_* \qquad Y^* \in \partial\|\lambda S\|_1$$

Thus the primal feasbility is the violation on the constraint on the problem. We will measure this error to be

$$\|\mathbf{Z} - \mathbf{L} - \mathbf{S}\|_F / \|\mathbf{Z}\|_F$$

and we will terminate once we are within a certain tolerance. From our extensive testing we found that $1e^{-4}$ was a good tolerance to terminate for this condition.

Our second stopping condition happens when we have a violation of dual feasibility. This is measured by the expression

$$\text{dist}(\partial\|L\|_*, \partial\|\lambda S\|_1)/\|\mathbf{Z}\|_F$$

Where the distance is $\min\{\|x - y\|_F | x \in \partial\|L\|_*, y \in \partial\|\lambda S\|_1\}$. However computing this distance is expensive because of the projection onto $\partial\|L\|_*$. So instead we will be using an approximation of $\mu_{k-1}\|S_k - S_{k-1}\|_F$. Thus our second stopping condition is

$$\min(\mu_k, \sqrt{\mu_k})\|S_k - S_{k-1}\|_F / \|\mathbf{Z}\|_F$$

We have found that $1e^{-}6$ is a good tolerance to stop for this measure of error.

Our last stopping condition involves with the parameter μ. For every iteration we decrease μ by a factor of 0.9. This helps with convergence and allows us to terminate our loop at a point with accurate results. However, once we will terminate the loop if it appears problem will not converge. This is if μ reaches the floor $\bar{\mu}$, which we set at the value of $10^{-9} \times \mu_0$.

Alternating Direction Method of Multipliers

To minimize the problem using an ADMM, we will use the augmented Lagrangian which takes this form in its most general case:

$$\mathcal{L}(L,S,Y,\mu) = f(x)_0 + \langle f(x), Y\rangle + \tfrac{\mu}{2}||f(x)||^2$$

Subproblems
Applying this general case to our minimization problem where the two subproblems will be iteratively solving $\mathbf{S}$ and $\mathbf{L}$ such that $\mathbf{X} = \mathbf{L} + \mathbf{S}$ yields the partial augmented Lagrangian:

$$\mathcal{L}(L,S,Y,\mu) = ||L||_* + \lambda||S||_1 + \langle X-L-S, Y\rangle + \tfrac{\mu}{2}||X-L-S||_F^2$$

Rather than iteratively solving $\mathbf{L}$ and $\mathbf{S}$ separately at each step, the ADMM will alternately solve $\mathbf{L}$ and $\mathbf{S}$ where $\mathbf{S}$ will be fixed to solve $\mathbf{L}$ and $\mathbf{L}$ will be fixed to solve $\mathbf{S}$. We will find these updates by taking the minimum of the Lagrangian with respect to $\mathbf{L}$ and $\mathbf{S}$ such that:

$$(L_{k+1}, S_{k+1}) = \arg\min_{L,S} ||L||_* + \lambda||S||_1 + \langle X-L-S, Y\rangle + \tfrac{\mu}{2}||X-L-S||_F^2$$

By deriving each component, we yield L_{k+1} and S_{k+1}:

$$\begin{aligned}
(L_{k+1}) &= \arg\min_L ||L||_* + \lambda||S||_1 + \langle X-L-S, Y\rangle + \tfrac{\mu}{2}||X-L-S||_F^2 \\
&= \arg\min_L ||L||_* + \frac{1}{2\mu_k^{-1}}||L-(X-S+\mu_k^{-1}Y)||_F^2 \\
(S_{k+1}) &= \arg\min_S ||L||_* + \lambda||S||_1 + \langle X-L-S, Y\rangle + \tfrac{\mu}{2}||X-L-S||_F^2 \\
&= \arg\min_S \lambda||S||_1 + \frac{1}{2\mu_k^{-1}}||S-(X-L+\mu_k^{-1}Y)||_F^2
\end{aligned}$$

Recognizing the form of these equations, recall the shrinkage operator (derived by the Theorem 2.1 of the paper from LMS) from the Quadratic Penalty Method above, such that we that apply the shrinkage operator to represent L_{k+1} and S_{k+1} in closed form. This will yield minimizers in the same form that we found in the Quadratic Penalty Method such that we are able to map the non-differentiable terms:

$$\begin{aligned}
L_{k+1} &= U\mathcal{S}_{\mu_k^{-1}}[\Sigma]V^\top \\
S_{k+1} &= \mathcal{S}_{\mu_k^{-1}}[X - L_{k+1} + \mu_k^{-1}Y]
\end{aligned}$$

Note the application of the shrinkage operator here is with the assumption that $U\Sigma V^\top = X - S + \mu_k^{-1}Y$ for the derivation of L_{k+1}. These closed-form solutions are our subproblems for our algorithm. Refer to shrinkage operator from Quadratic Penalty Method (as they are derived from the same logic).
Iterative Update

In addition to updating out subproblems, in each iteration we update Y_{k+1} with the addition of the primal feasibility multiplied by a factor of the penalty parameter (μ) such that $Y_{k+1} = Y + \mu(X - L - S)$. The penalty parameter is derived from the size of the data set and then scaled by 10. We chose this as our penalty parameter as we wanted the variable to fit relative to the data and multiplying it by a factor of 10 allowed our results to be clean. This provided the best results after extensive testing.

Stopping Condition

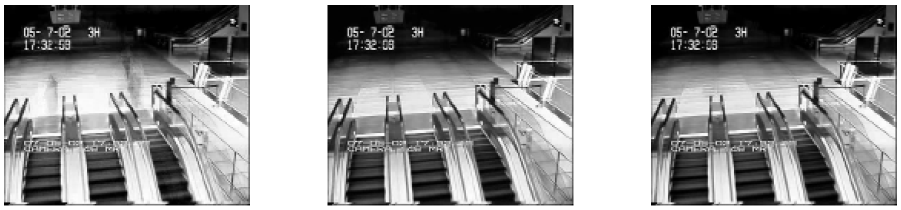
Our stopping condition is similar to the one we used in our Quadratic Penalty Method which defined the condition as $\dfrac{||X - L - S||_F}{||X||_F} < tol$ where tol is a tolerance defined as $1.0e - 4$. Refer to first stopping condition from Quadratic Penalty Method (as they are derived from the same logic).

Comparison of Algorithms

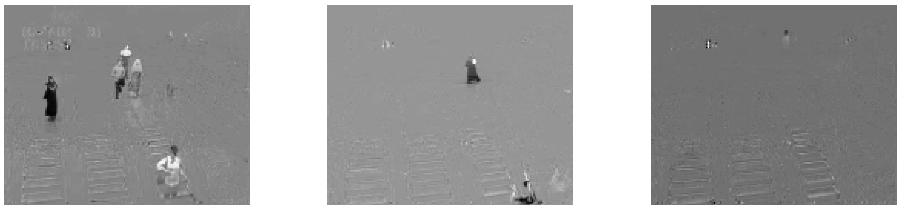
To run our code, be sure that the sample data, admm_solver.m, penalty_solver.m, and test_model_RPCA.m scripts are in the same directory and run test_model_RPCA.m. This will call both scripts for both provided datasets (random data and escalator). Below is a sample output:

Output

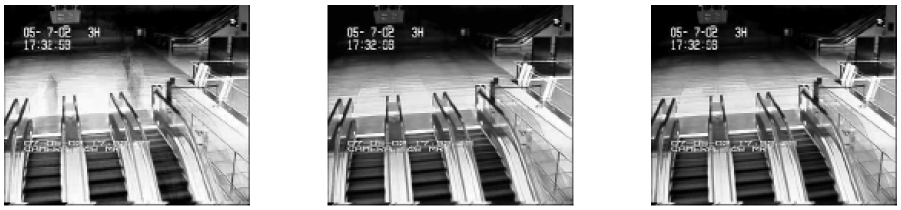
```
ADMM Solver:                          Penalty Solver:
Random Data Test                      Random Data Test
||L-Lopt||/||Lopt|| = 3.2518e-05      ||L-Lopt||/||Lopt|| = 4.6753e-05
||S-Sopt||/||Sopt|| = 9.6964e-04      ||S-Sopt||/||Sopt|| = 8.9496e-04
Elapsed Time: 0.989668 seconds        Elapsed Time: 6.650571 seconds
Iterations: 13                        Iterations: 99
Final residual: 9.5248e-05            Final residual: 9.3958e-05
--------------------                  --------------------
Escalator Data Test                   Escalator Data Test
See Output Images                     See Output Images
Elapsed Time: 65.793886 seconds       Elapsed Time: 27.844079 seconds
Iterations: 220                       Iterations: 113
Final residual: 9.9682e-05            Final residual: 9.3651e-05
```



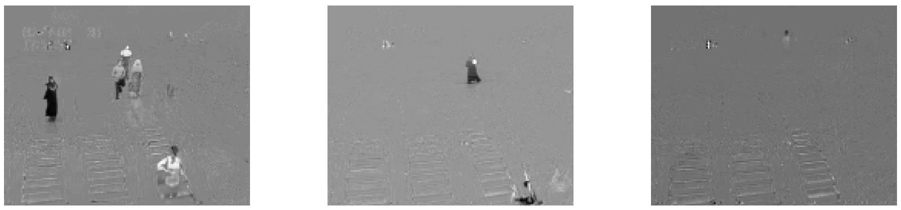
(a) L3 ADMM



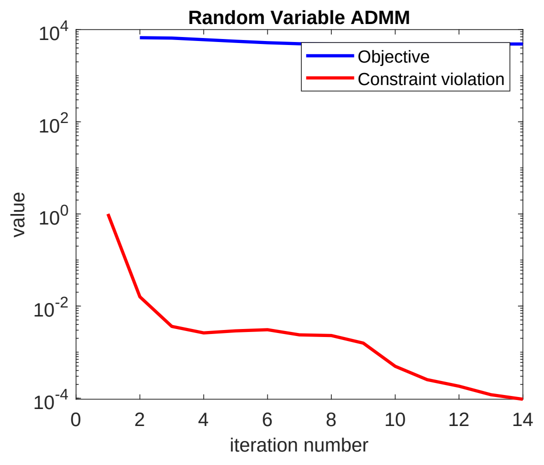
(b) S3 ADMM



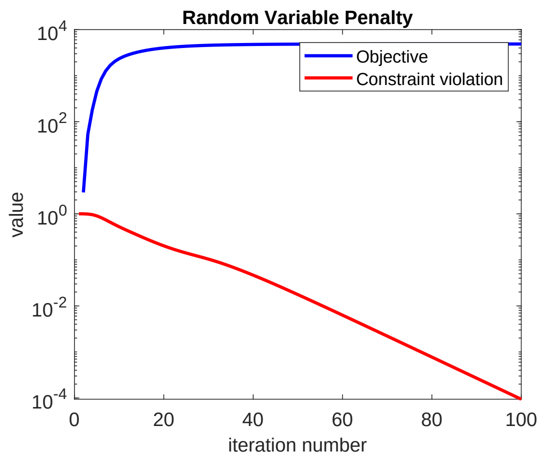
(c) L3 Penalty



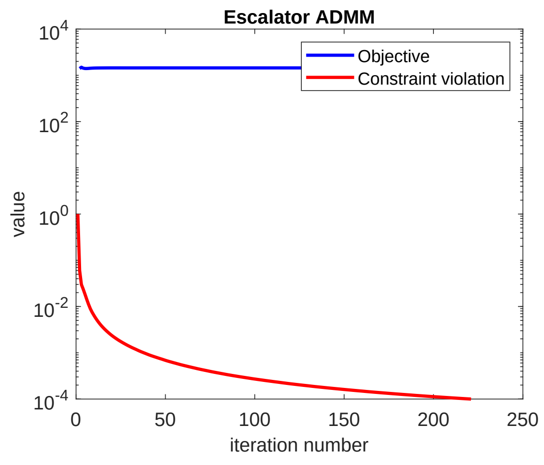
(d) S3 Penalty



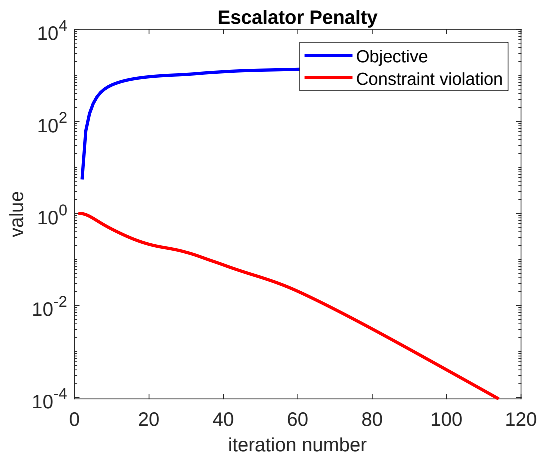
(e) Random Variable ADMM



(f) Random Variable Penalty



(g) Escalator ADMM



(h) Escalator Penalty

Discussion

Looking at the graphs for each of the four cases we see an obvious trend. For the quadratic penalty method, we can see a clear trend on how the objective value converges the more iterations we do. For both the random variable test and the escalator test we can see that objective value converges as we do more iterations. Since the y axis was plotted at a exponential scale, with ever index representing an order of 10 bigger. This means that it increases exponentially at first and then levels off. This makes sense because we added a penalty parameter to this particular problem. For the constraint violation, which is how much the dual feasibility has been violated, we can see a linear decrease in error on this scale. This means that as more iterations are being performed, we are constantly reducing the error for the dual feasibility on both tests. What is interesting is that even though the objective is outputting similar values the error is still decreasing on this scale.

For ADMM we see a different trend in our graphs. The objective value is near flat on both test cases. Since this scale is based on orders of 10, it implies that the objective might have increased but not as much for each specific case even though it converges. On the random variable test for ADMM the objective decreases and on the escalator the objective increases. This makes sense because unlike the penalty method, there was no penalty term which contributed to it's exponentially convergent behavior. As for the the constraint violation we see that for the two examples it's decreases look logarithmic on this scale. Meaning that error is near convergent on $1e^{-}4$ for both cases. For the random variable it appears that the there are iterations like 1 and 10, where the error decreases more drastically.

Based on the console output of our code, our ADMM Solver performed better on the Random Data Sample and the Penalty Solver performed better on the escalator data. Because the results and final residual between both tests for both data samples are similar to each (percent error and image slices for random and escalator data, respectively), the performance of our algorithms are based off of the elapsed time and number of iterations. We expected that the ADMM Solver would be better in both cases, we think that the reason our ADMM algorithm was outperformed by the Penalty algorithm was due to lack of fine tuning of the variable μ. When forming the Penalty method we spent a lot of time fine tuning the warm start and iteration updates so that it would terminate within reasonable time and with good results. We did not have the time to do the same for ADMM. In the ADMM algorithm we do not iteratively update μ which could have a negative affect on how efficiently our algorithm handles the subproblems. If we had more time, we would have continued optimizing the μ parameter. However, when plotting the constraint violations between each algorithm, the constraint violation of the ADMM algorithm initially converges at a must faster rate (see graphs). This supports our claim that the closed form solutions and iterative steps for ADMM are in fact more efficient and that fine tuning and further experimentation with parameters, such as μ, would give us results that would reflect this. In the case of Robust PCA, we believe that ADMM is generally a better algorithm.

References

In addition to lecture notes and previous homework assignments:

- Balandat, M., Krichene, W., Lam, C. and Lam, K., 2012. Robust Principal Component Analysis. [online] People.eecs.berkeley.edu.
- Cai, J., Candes, E. and Shen, Z., 2010. A singular value thresholding algorithm for matrix completion. SIAM Journal on optimization,.
- Lin, Z., Chen, M. and Ma, Y., 2011. The Augmented Lagrange Multiplier Method For Exact Recovery Of Corrupted Low-Rank Matrices. [online] Arxiv.org.