DSA5103
Optimization algorithms for data modelling

National University of Singapore
Data Science and Machine Learning Department
April 16, 2023

Assignment 3

Wang Jiangyi, A0236307J
e0732857@u.nus.edu

1 Intersection of two convex sets via ADMM

Intersection of two convex sets $C, D \subseteq \mathbb{R}^n$ can be found via solving:

$$\min_x \delta_C(x) + \delta_D(x)$$

1.1 Q(a): Formulate into 2-block separable structure

$$\begin{aligned} &\min_x \delta_C(x) + \delta_D(x) \\ \iff &\begin{cases} \min\limits_{x,z\in\mathbb{R}^n} \ \delta_C(x) + \delta_D(z) \\ \text{s.t. } x - z = 0 \end{cases} \end{aligned} \tag{1}$$

1.2 Q(b): Explicit formular of ADMM iteration

1. Augmented Lagrangian function ($\xi \in \mathbb{R}^n$):

$$\begin{aligned} L_\sigma(x, z; \xi) &= \delta_C(x) + \delta_D(z) + \langle \xi, x - z \rangle + \frac{\sigma}{2}||x - z||^2 \\ &= \delta_C(x) + \delta_D(z) + \frac{\sigma}{2}||x - z + \sigma^{-1}\xi||^2 - \frac{1}{2\sigma}||\xi||^2 \end{aligned}$$

2. Subproblem-x:

$$\begin{aligned} \hat{x} &= arg\min_x L_\sigma(x, z; \xi) \\ &= arg\min_x \delta_C(x) + \frac{\sigma}{2}||x - z + \sigma^{-1}\xi||^2 \\ &= arg\min_x \delta_C(x) + \frac{1}{2}||x - z + \sigma^{-1}\xi||^2 \\ &= \Pi_C(z - \sigma^{-1}\xi) \end{aligned}$$

3. Subproblem-z:

$$\begin{aligned} \hat{z} &= arg\min_z L_\sigma(x, z; \xi) \\ &= arg\min_z \delta_D(z) + \frac{\sigma}{2}||z - x - \sigma^{-1}\xi||^2 \\ &= arg\min_x \delta_D(z) + \frac{1}{2}||z - x - \sigma^{-1}\xi||^2 \\ &= \Pi_D(x + \sigma^{-1}\xi) \end{aligned}$$

4. To conclude, when we are <u>at the k-iteration</u>, that is, we have $(x^{(k)}, z^{(k)}; \xi^{(k)})$, the ADMM iteration is:

$$\begin{aligned} x^{(k+1)} &= \Pi_C(z^{(k)} - \sigma^{-1}\xi^{(k)}) \\ z^{(k+1)} &= \Pi_D(x^{(k+1)} + \sigma^{-1}\xi^{(k)}) \\ \xi^{(k+1)} &= \xi^{(k)} + \tau\sigma(x^{(k+1)} - z^{(k+1)}) \end{aligned}$$

1.3 Q(c): First two iterations of ADMM

1. Initialize $x^{(0)} = z^{(0)} = \xi^{(0)} = 0, \tau = \sigma = 1$.
2. Iteration 1:

$$\begin{aligned} x^{(1)} &= \Pi_C(z^{(0)} - \xi^{(0)}) = 1 \\ z^{(1)} &= \Pi_D(x^{(1)} + \xi^{(0)}) = 1.5 \\ \xi^{(1)} &= \xi^{(0)} + (x^{(1)} - z^{(1)}) = -0.5 \end{aligned}$$

2. Iteration 2:

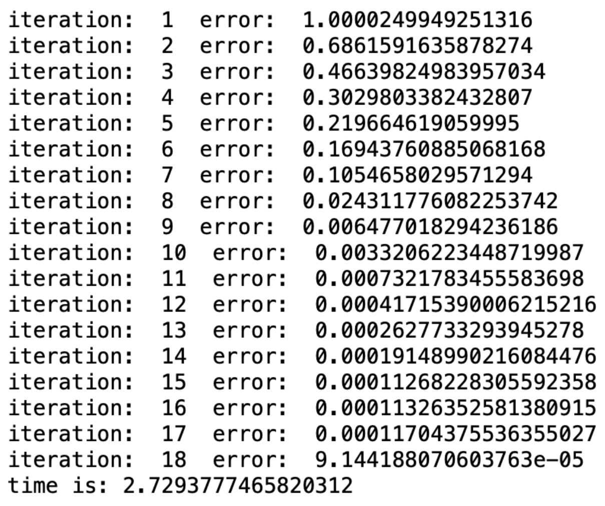
$$x^{(2)} = \Pi_C(z^{(1)} - \xi^{(1)}) = 2$$
$$z^{(2)} = \Pi_D(x^{(2)} + \xi^{(1)}) = 1.5$$
$$\xi^{(2)} = \xi^{(1)} + (x^{(2)} - z^{(2)}) = 0$$

2 Robust PCA via ADMM

Before show the result for each question, we clarify the initialization setting first. We **do not apply zero matrix initialization** for each (L_0, S_0, Z_0). Reason is, in Q(b), we are required to use the inverse of largest eigenvalue to initialize σ, forcing σ to be very small. With this initialization setting, soft-threshold operator will shrink almost every parameter to 0, which will terminate our ADMM algorithm since our initialization is also zero matrix. Therefore, we use Gaussian random variables (0 mean and 0.1 standard deviation) to initialize (L_0, S_0, Z_0) in the following experiments.

2.1 Q(a): Report naive implementation of ADMM

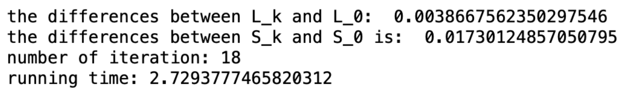
Here, we use 18 iteration to make ADMM converge. We report the iteration, termination signal $r^{(k)}$ and running time in Figure 1.



```
iteration:  1  error:  1.0000249949251316
iteration:  2  error:  0.6861591635878274
iteration:  3  error:  0.46639824983957034
iteration:  4  error:  0.3029803382432807
iteration:  5  error:  0.219664619059995
iteration:  6  error:  0.16943760885068168
iteration:  7  error:  0.1054658029571294
iteration:  8  error:  0.024311776082253742
iteration:  9  error:  0.006477018294236186
iteration:  10  error:  0.0033206223448719987
iteration:  11  error:  0.0007321783455583698
iteration:  12  error:  0.00041715390006215216
iteration:  13  error:  0.0002627733293945278
iteration:  14  error:  0.00019148990216084476
iteration:  15  error:  0.00011268228305592358
iteration:  16  error:  0.00011326352581380915
iteration:  17  error:  0.00011704375536355027
iteration:  18  error:  9.144188070603763e-05
time is: 2.7293777465820312
```

Figure 1: Iteration, Termination signal $r^{(k)}$, Running time Report (naive implementation)

Moreover, we report the difference between our solution and ground truth in Figure 2.



```
the differences between L_k and L_0:  0.0038667562350297546
the differences between S_k and S_0 is:  0.01730124857050795
number of iteration: 18
running time: 2.7293777465820312
```

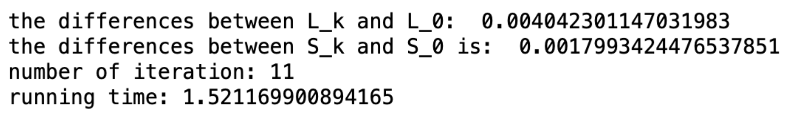
Figure 2: Difference between our solution and ground truth Report (naive implementation)

2.2 Q(b): Speed up ADMM via some tricks

Here, we actually make three different experiments on different choices of ρ. Before showing the experiment results, we state of conclusion first. We should choose ρ as an intermediate value. If ρ is too big, then our achieved solution will be very poor although the ADMM algorithm converges very fast. If ρ is too small (close to 1), then our ADMM algorithm will converge very slowly, although our achieved solution can converges to the ground truth as we want.

2.2.1 $\rho = 3$, which is an intermediate value (our interested scenario)

Here, we choose $\rho = 3$ to conduct experiments.



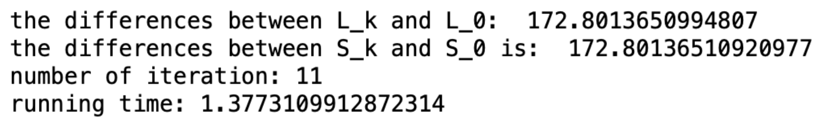
```
the differences between L_k and L_0:  0.004042301147031983
the differences between S_k and S_0 is:  0.0017993424476537851
number of iteration: 11
running time: 1.521169900894165
```

Figure 3: Error, Iteration, Running time Report ($\rho = 3$)

As shown in Figure 3, with good choice of ρ, the ADMM algorithm is able to converge faster. We only need 11 iteration in this speed-up version, while 18 iteration is required for naive implementation. Moreover, from the difference reported, we can verify that, the achieved solution correctly converges to the ground truth.

2.2.2 $\rho = 4$, which is too big (bad case)

Here, we choose $\rho = 4$ to conduct experiments.



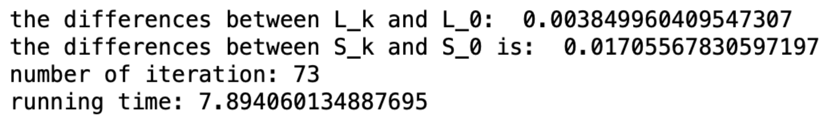
```
the differences between L_k and L_0:  172.8013650994807
the differences between S_k and S_0 is:  172.80136510920977
number of iteration: 11
running time: 1.3773109912872314
```

Figure 4: Error, Iteration, Running time Report ($\rho = 4$)

As shown in Figure 4, when we set ρ too big, our ADMM algorithm **will not converge to the ground truth** from the reported difference. Moreover, it can be also observed that, the ADMM algorithm converges very fast, which only requires 11 iteration for convergence.

2.2.3 $\rho = 1.1$, which is too small (bad case)

Here, we choose $\rho = 1.1$ to conduct experiments.



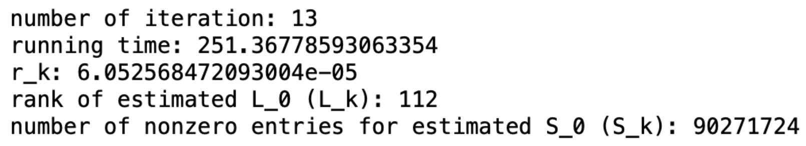
```
the differences between L_k and L_0:  0.003849960409547307
the differences between S_k and S_0 is:  0.01705567830597197
number of iteration: 73
running time: 7.894060134887695
```

Figure 5: Error, Iteration, Running time Report ($\rho = 1.1$)

As shown in Figure 5, when we set ρ too small, our ADMM converges **even more slowly compared with the naive implementation**. The reason is, our initialization for σ is the inverse of largest eigenvalue, which is a very small value in general. If the increasing rate ρ is also small, e.g., $\rho = 1.1$, then σ will keep small for a long time, which will restrict the convergence of our algorithm. However, our achieved solution can correctly converge to the ground truth.

2.3 Q(c): Application of foreground-background segmentation

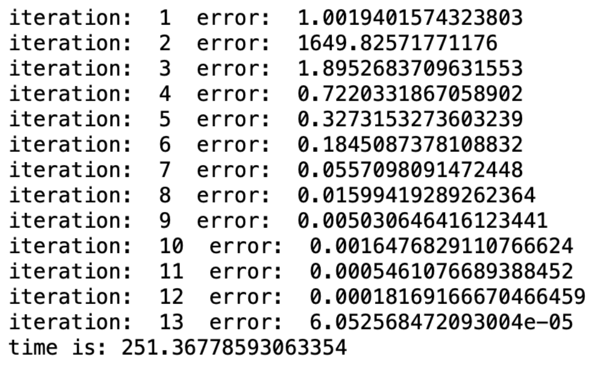
Based on the previous discussion, here we set $\rho = 3$ to achieve the trade-off between convergence speed and solution quality.



```
number of iteration: 13
running time: 251.36778593063354
r_k: 6.052568472093004e-05
rank of estimated L_0 (L_k): 112
number of nonzero entries for estimated S_0 (S_k): 90271724
```

Figure 6: Iteration, Running time, Error, Rank and Number of non-zero entries Report ($\rho = 3$)

Moreover, we report our whole learning trajectory in Figure 7:



```
iteration:  1  error:  1.0019401574323803
iteration:  2  error:  1649.82571771176
iteration:  3  error:  1.8952683709631553
iteration:  4  error:  0.7220331867058902
iteration:  5  error:  0.3273153273603239
iteration:  6  error:  0.1845087378108832
iteration:  7  error:  0.0557098091472448
iteration:  8  error:  0.01599419289262364
iteration:  9  error:  0.005030646416123441
iteration:  10  error:  0.0016476829110766624
iteration:  11  error:  0.0005461076689388452
iteration:  12  error:  0.00018169166670466459
iteration:  13  error:  6.052568472093004e-05
time is: 251.36778593063354
```

Figure 7: Whole Learning Trajectory ($\rho = 3$)

2.4 Q(d): Visualize M[:, 19], L[:,19], S[:,19]

Firstly, we show the total pictures M[:,19], which is the 20-th frame in the video in Figure 8.

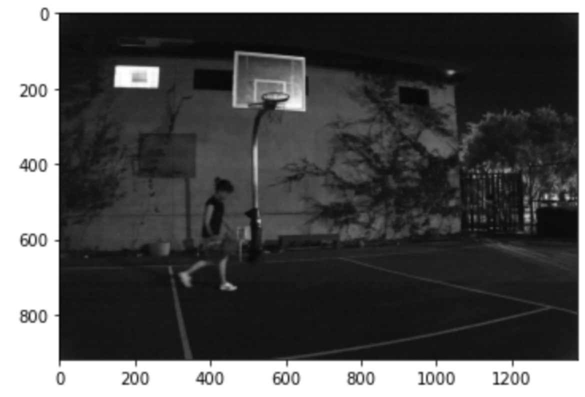
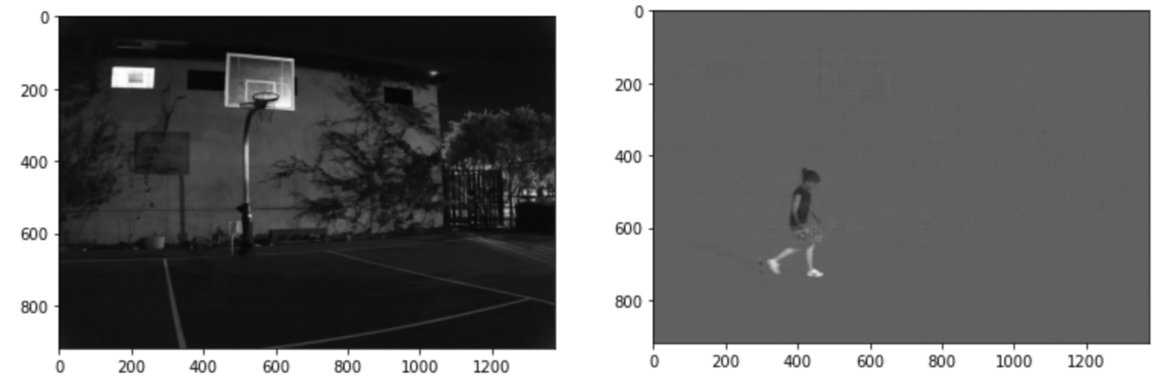


Figure 8: 20-th frame in the video, M[:,19]

Secondly, we show our separation result for the background and person.



(a) 20-th background frame

(b) 20-th person frame

Figure 9: 20-th Background and Person Frame, L[:,19] and S[:,19]

2.5 Q(e): Making a Background Video and Moving Object Video

For this part, we will attach our 'background.avi' and 'people.avi' file. We successfully apply 'cv2' to visualize the result as videos.