Homework 5

Convex Optimization 10-725/36-725

Due Tuesday April 21 at 4:00pm
submitted to Mallory Deptola in GHC 8001
(make sure to submit each problem separately)

***Instructions: you will solve 4 questions, choosing **two of** Q1–Q3 and **two of** Q4–Q6.

1 Isomorphism of primal-dual feasible set [Veeru]

Given an LP:

$$\begin{aligned} \min_x \quad & c^T x \\ \text{subject to} \quad & Ax = b,\ x \geq 0, \end{aligned}$$

recall the set of (strictly) feasible primal-dual points, from the first lecture on primal-dual interior point methods:

$$\mathcal{F}_0 = \{(x, y, s) : Ax = b, A^T y + s = c, x > 0, s > 0\}.$$

Consider the map from $\mathcal{F}_0 \to \mathbb{R}^n_{++}$, defined by

$$(x, y, s) \mapsto x \circ s,$$

where $x \circ s = (x_1 s_1, \ldots x_n s_n)$ denote the elementwise product between x and s. Assume that $\mathcal{F}_0$ is nonempty and $A \in \mathbb{R}^{m \times n}$ has full row rank. In this problem you will prove that the above map is a bijection, i.e., it uniquely associates each element of $\mathcal{F}_0$ with an element of $\mathbb{R}^n_{++}$, and vice versa.

(a) Consider the primal and dual barrier problems:

$$\begin{aligned} \min_x \quad & c^T x - \tau \sum_{i=1}^n \log x_i \\ \text{subject to} \quad & Ax = b \end{aligned}$$

and

$$\begin{aligned} \max_{y,s} \quad & b^T y + \tau \sum_{i=1}^n \log s_i \\ \text{subject to} \quad & A^T y + s = c. \end{aligned}$$

Prove that both problems have unique solutions, for any barrier parameter $\tau > 0$. Note that you need to show the existence of the minimizers as well.

(b) Use part (a) and the KKT conditions to prove that, for all $\tau > 0$, there exists a unique $(x, y, s) \in \mathcal{F}_0$ such that

$$Ax = b,\ A^T y + s = c,\ (x \circ s) = \tau \mathbb{1},$$

where $\mathbb{1}$ is the vector of all 1s.

(c) Tweak the barrier problems in part (a) to show that the same result holds as in (b), but when $\tau\mathbb{1}$ is replaced by any vector that is positive in each component. Conclude that the map $(x, y, s) \mapsto x \circ s$ is indeed a bijection.

(d) Opposite in some sense to interior point methods are *penalty methods*, which approach the constraint set from the outside of the feasible region. For our original LP, consider the following modification:

$$\min_x \; c^T x + \tau \cdot \Big(\sum_{i=1}^m |a_j^T x - b_j| + \sum_{i=1}^m (x_i)_- \Big),$$

where $a_1, \ldots a_m$ denote the rows of A, $\tau > 0$ is a penalty parameter and $(u)_- = \max(0, -u)$ for $u \in \mathbb{R}$. Consider an increasing sequence of parameter values $\{\tau_k\}_{k=1,2,3,\ldots}$, with $\tau_k \to \infty$, and let $x^\star(\tau_k)$ denote a solution to the above penalized problem at $\tau = \tau_k$. Let $\bar{x}$ be a limit point of the sequence $\{x^\star(\tau_k)\}_{k=1,2,3,\ldots}$. Assume that the original LP has a solution $x^\star$. Prove that $\bar{x}$ is a solution to the original LP.

Hint: Let $f(x) = c^T x$, $p(x) = \sum_{i=1}^m |a_j^T x - b_j| + \sum_{i=1}^m (x_i)_-$, and $q(x, \tau) = f(x) + \tau \cdot p(x)$. Abbreviate $x_k = x^\star(\tau_k)$. You may assume without proof the following facts, for each $k = 1, 2, 3, \ldots$:

- $q(x_k, \tau_k) \leq q(x_{k+1}, \tau_{k+1})$;
- $p(x_k) \geq p(x_{k+1})$;
- $f(x_k) \leq f(x_{k+1})$;
- $f(x^\star) \geq q(x_k, \tau_k) \geq f(x_k)$.

Bonus: prove these facts!

2 Safe support vector elimination for SVMs [Junier]

We've already seen some pretty neat applications of KKT conditions, but one of the coolest applications of KKT analysis is that of eliminating covariates (and/or instances) in a lossless manner. That is, with a careful look at KKT conditions techniques like SAFE rules[1] allow us to safely eliminate covariates or instances without any optimization of the original problem. Such screening rules are extraordinarily useful since they allow us to better scale our optimization algorithms by solving a smaller problem after discarding covariates or instances. Although SAFE rules are typically employed in ℓ_1 penalized optimization, we shall use a similar analysis to rule out non-support vectors from a standard SVM problem below.

We consider the following constrained SVM problem:

$$\min_w J(w) \equiv \frac{1}{2}\|w\|^2 \quad \text{subject to} \quad H(w) \equiv \sum_{i \in \mathcal{N}} [1 - y_i f_w(x_i)]_+ \leq s, \tag{1}$$

where $\mathcal{N} = \{i \in \mathbb{N} : 1 \leq i \leq N\}$, $x_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$, $f_w(x) \equiv w^T x$ and $s > 0$. Furthermore, we shall define $w^*(s)$ as the optimal solution of (1) and $f^*(x|s) \equiv w^*(s)^T x$.

(a) Prove that $w^*(s)$ is the optimizer of :

$$\min_v J(v) \equiv \frac{1}{2}\|v\|^2 \quad \text{subject to} \quad H(v) \equiv \sum_{i \notin \mathcal{R}} [1 - y_i f_v(x_i)]_+ \leq s,$$

[1] http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-126.pdf

where $\mathcal{R} = \{i \in \mathcal{N} : y_i f^*(x_i|s) > 1\}$. In other words, show that the instances $i \in \mathcal{R}$ do not affect the solution of (1), and can hence be discarded. Hint: consider the Lagrangian of (1), and analyze stationarity and complementary slackness to show that they are not support vectors.

(b) Suppose that $s_a > s_b$.

1. Show that $\sigma \in [s_b, s_a] \implies J(w^*(\sigma)) \leq J(w^*(s_b))$.
2. Show that $\sigma \in [s_b, s_a] \implies w^*(s_a)^T(w^*(\sigma) - w^*(s_a)) \geq 0$. Hint: consider KKT conditions for (1), use convexity of $H(w)$ and primal feasibility of $w^*(\sigma)$ for (1) with $s = s_a$.
3. Show that we may discard an instance $i \in \mathcal{N}$ in the optimization of (1) with $s \in [s_b, s_a]$ if $g_{[s_b,s_a]}(i) > 1$ where:

$$g_{[s_b,s_a]}(i) \equiv \min_{w \in \Theta_{[s_b,s_a]}} y_i f_w(x_i), \tag{2}$$

and $\Theta_{[s_b,s_a]} \equiv \{w : J(w) \leq J(w^*(s_b)) \wedge w^*(s_a)^T(w - w^*(s_a)) \geq 0\}$.

(c) We reduce the screening rule of $g_{[s_b,s_a]}(i) > 1$ to an analytical formula below. Let $\gamma_b \equiv J(w_b^*)$ and $\gamma_a \equiv J(w_a^*)$.

1. Write out the Lagrangian of $g_{[s_b,s_a]}(i)$ with Lagrange multipliers of μ and ν for constraints $J(w) \leq \gamma_b$ and $w^*(s_a)^T(w - w^*(s_a)) \geq 0$ respectively.
2. Write the KKT conditions of $g_{[s_b,s_a]}(i)$, the optimizer z^* of (2), and the optimal criterion value in terms of μ and ν.
3. Show that:

$$g_{[s_b,s_a]}(i) = \begin{cases} -\sqrt{2\gamma_b}\|x_i\| & \text{if } -\frac{y_i f^*(x_i|s_a)}{\|x_i\|} \geq \frac{\sqrt{2}\gamma_a}{\sqrt{\gamma_b}} \\ y_i f^*(x_i|s_a) - \sqrt{\frac{\gamma_b - \gamma_a}{\gamma_a}\left(2\gamma_a\|x_i\|^2 - f^*(x_i|s_a)^2\right)} & \text{otherwise.} \end{cases}$$

Hint: use the sign of $-\frac{y_i f^*(x_i|s_a)}{\|x_i\|} - \frac{\sqrt{2}\gamma_a}{\sqrt{\gamma_b}}$ to guide whether $\nu = 0$.

4. Further simplify the screening rule of $g_{[s_b,s_a]}(i) > 1$ to:

$$y_i f^*(x_i|s_a) > 1 \quad \text{and} \quad y_i f^*(x_i|s_a) - \sqrt{\frac{\gamma_b - \gamma_a}{\gamma_a}\left(2\gamma_a\|x_i\|^2 - f^*(x_i|s_a)^2\right)} > 1.$$

3 Coordinate descent for the graphical lasso [Mattia]

Let $X \in \mathbb{R}^{n \times p}$ be a data matrix whose rows are independent observations from $\mathcal{N}(0, \Sigma)$. Normality theory tells us that for $Z \sim \mathcal{N}(0, \Sigma)$, $\Sigma^{-1}_{ij} = 0$ implies that the variables Z_i and Z_j are conditionally independent given all the other variables $\{Z_k\}_{k \notin \{i,j\}}$. If we believe that many pairs of features recorded in X are conditionally independent given the other features (which is often a reasonable belief if p is large), then we want an estimate of Σ such that Σ^{-1} is sparse. This goal can be achieved by solving the graphical lasso problem

$$\min_{\Theta \in \mathbb{S}^p_{++}} -\log\det\Theta + \operatorname{tr}(S\Theta) + \lambda\|\Theta\|_1, \tag{3}$$

where $\mathbb{S}^p_{++}$ is the set of symmetric $p \times p$ positive definite matrices. In the above, $S = X^T X/n$ is the emiprical covariance matrix, Θ serves as our estimate for Σ^{-1}, and $\|\Theta\|_1 = \sum_{i,j=1}^p |\Theta_{ij}|$.

a) Verify that the KKT stationarity condition for the graphical lasso problem above is

$$-\Theta^{-1} + S + \lambda\Gamma = 0$$

where $\Gamma_{ij} \in \partial|\Theta_{ij}|$. To do this, you may need to look up how to compute the gradient of the log-determinant of a positive definite matrix and the gradient of the trace of a linear transformation of a symmetric matrix! Let now $W = \Theta^{-1}$. Claim that the KKT stationarity condition above implies that $W_{ii} = S_{ii} + \lambda$ for every $i = 1, \ldots, p$.

Consider now partitioning W as

$$W = \begin{pmatrix} W_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}$$

where $W_{11} \in \mathbb{R}^{(p-1)\times(p-1)}$, $w_{12} \in \mathbb{R}^{p-1}, w_{21}^T \in \mathbb{R}^{p-1}$, and $w_{22} \in \mathbb{R}$ (and consider the same partition for the other matrices Θ, S, and Γ).

b) Using the fact that $W\Theta = I$ and the KKT stationarity condition above, show that

$$w_{12} = -W_{11}\theta_{12}/\theta_{22}$$

and therefore

$$W_{11}\frac{\theta_{12}}{\theta_{22}} + s_{12} + \lambda\gamma_{12} = 0.$$

c) Let now $\beta = \theta_{12}/\theta_{22} \in \mathbb{R}^{p-1}$. Write a lasso problem (with β as the optimization variable) such that $\beta = \theta_{12}/\theta_{22}$ is a solution.

Once the lasso problem of part c) is solved, it is easy to recover $w_{12} = -W_{11}\beta$ and $w_{21} = w_{12}^T$. Furthermore, θ_{12}, θ_{21}^T, and θ_{22} are directly obtained by solving

$$\begin{pmatrix} W_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} \Theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & 1 \end{pmatrix}.$$

By rearranging appropriately the entries of W, Θ, S, and Γ, one can then iterate the procedure above to solve for the entire matrix W.

d) Describe an algorithm to estimate Θ based on iterative blockwise minimization using the results of parts a), b), and c).

e) Explain why, strictly speaking, such an algorithm is in fact *not* a blockwise coordinate descent algorithm for the graphical lasso problem as formulated in equation (3).

We will now verify that the coordinate descent algorithm suggested by parts a), b), and c) (which optimizes over the matrix W and is known as 'glasso') is in fact a *proper* coordinate descent algorithm for the dual problem of the graphical lasso as formulated in equation (3).

f) Consider the graphical lasso dual problem

$$\min_{\tilde{\Gamma} \in \mathbb{S}^p} \; -\log\det(\tilde{\Gamma} + S) - p \;\; \text{subject to} \;\; \|\tilde{\Gamma}\|_\infty \leq \lambda. \tag{4}$$

Write the KKT conditions for the dual problem of equation (4). In particular, verify that the KKT stationarity condition of the primal graphical lasso problem of equation (3) can be retrieved from the KKT conditions of the dual problem (possibly after appropriate changes of variable). Finally, briefly clarify why the 'glasso' algorithm that you derived in part d) is a proper coordinate ascent algorithm for the dual problem of the graphical lasso.

Bonus: Derive the dual of the graphical lasso problem of equation (3).

4 Primal-dual interior point methods for LPs [Veeru]

For the following LP:

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{subject to} \quad & Ax = b,\ x \geq 0, \end{aligned}$$

implement the infeasible path-following algorithm. Assume that A has full row-rank. The method is described on slide 20 in the first lecture on primal dual interior point methods.

Run the algorithm for the A, b, c given in `Q4.mat` or `Q4.Rdata`. You may use the other algorithmic parameters given in `Q4.mat` or `Q4.Rdata` but you can experiment with the parameters for possibly faster convergence.

(a) Write down the key steps and any important implementation details, such as how you solve the Newton step and how you make sure you stay in the neighbourhood of the central path. Attach your code.

(b) Let $r_b = Ax - b, r_c = A^T y + s - c$ as in the slides. Plot the (i) residual $\|(r_b, r_c, x \circ s)\|_2$ vs. number of Newton steps and (ii) primal objective vs. number of Newton steps separately until the residual is reduced to $\leq$ 1e-8 times 1+max($\|b\|_2, \|c\|_2$). Verify whether the residual plot reflects R-linear convergence of the residual.

5 NMF via alternating gradient descent [Nicole]

Let $V \in \mathbb{R}^{m \times n}$ be a given matrix, and let $0 < k < \min\{m, n\}$ be a given integer. In nonnegative matrix factorization (NMF), we solve

$$\min_{W \in \mathbb{R}^{m \times k},\, H \in \mathbb{R}^{k \times n}} \|V - WH\|_F^2 \ \text{ subject to } \ W \geq 0,\ H \geq 0, \tag{5}$$

where the constraints $W \geq 0$, $H \geq 0$ are to be interpreted elementwise.

The NMF problem is a nonconvex problem, and therefore it is difficult to solve. Interestingly, there is an elegant algorithm for NMF described in Lee & Seung (2001), "Algorithms for non-negative matrix factorizations" (available here: `http://www.dm.unibo.it/~simoncin/nmfconverge.pdf`). Some motivation for NMF can be found in other papers written by these same authors, e.g., Lee & Seung (2001), "Learning the parts of objects by non-negative matrix factorization" (available here: `http://www.nature.com/nature/journal/v401/n6755/abs/401788a0.html`).

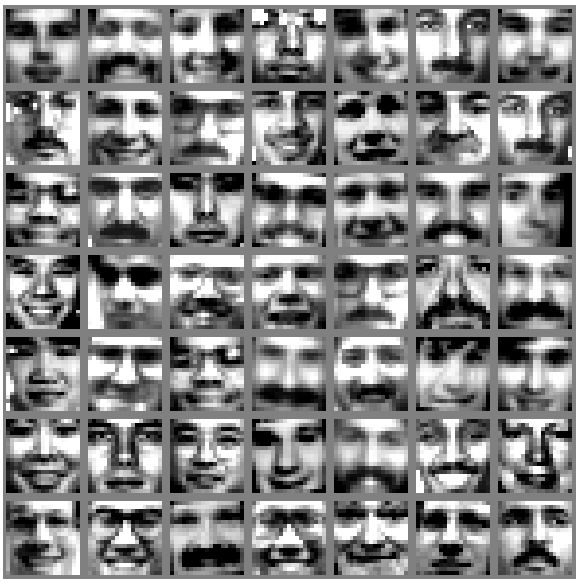
Your task is to implement this NMF gradient descent algorithm on the CBCL FACE database (taken from `http://cbcl.mit.edu/software-datasets/FaceData2.html`) . A sample of the original faces look like:

To generate plots throughout, see the provided scripts `compact_canvas.m` or `compact_canvas.R`.

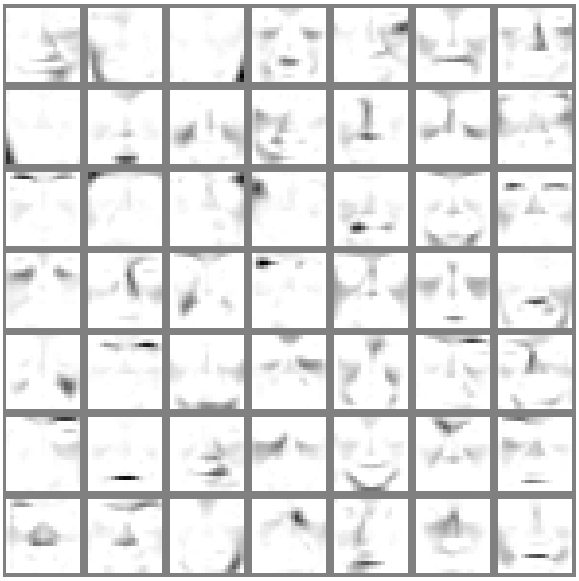
(a) The solution proposed in Lee & Seung alternately minimizes W and H using convex optimaztion techniques, i.e. we fix W and optimize over H, then fix H and optimize over W, and repeat. Why is this a good approach? (Hint: think about what is true about the problem if H is fixed)

(b) On the course website, you will find the file `face.txt`, which contains 2459 faces, each of which is a 19 x 19 grayscale image. Therefore the file can be read into memory as an array of dimension 361 x 2459, the columns being unraveled images of faces, taking values in between 0 and 255.

Apply the following preprocessing: for each face (column) V_i, normalize it so that the median is $m_i = \text{med}(V_i) = 0.5$, and the median absolute deviation from the median is $\text{med}(|V_i - m_i|) = 0.25$. Truncate any values above 1 and below 1e-4 (since we do not want exact zeros). Now, you should have a matrix V that looks like (sampled columns):



(c) Implement the algorithm described in Theorem 1 of Lee & Seung (2001) to optimize (5). Set $k = 49$, and initialize every entry of W, H uniformly at random in $[0, 1]$. Plot a sample of the columns of W, similar to the below. Also, plot the objective value across iterations. Does the objective function strictly decrease? Why?



(d) Each step of this (Lee & Seung) is in special case of a general descent step that we know very well. Which one? Why?

(e) What are alternatives, still staying within the realm of first-order methods, to ensure that the entries of W, H are nonnegative on each iteration? What could we do to speed up convergence?

(f) On the course website you will find the file `face_test.txt`, which contains 472 test faces in the same format as `face.txt`. Reconstruct these faces by optimizing H with your trained W from part (b) fixed. What is the average squared error of the reconstructed images (normalized by the number of images)? Do your reconstructed faces look similar to the test faces?

6 Robust PCA and ADMM [Yuxiang]

In this question, we will develop a fast algorithm for Robust Principal Component Analysis (RPCA) and use it for the problem of foreground-background segmentation.

RPCA is an interesting optimization formulation independently developed by two groups of authors: Candes, Li, Ma and Wright (`http://statweb.stanford.edu/~candes/papers/RobustPCA.pdf`) and Chandrasekaran, Sanghavi, Parrilo and Willsky (`http://arxiv.org/pdf/0906.2220.pdf`) in 2009.

Let $M \in \mathbb{R}^{m \times n}$ be the observed matrix and we hope to decompose it into the sum of a low-rank matrix L and sparse matrix S.

$$\min_{L,S} \|L\|_* + \lambda \|S\|_1 \text{s.t. } L + S = M \tag{6}$$

where $\|\cdot\|_*$ is the nuclear norm/trace norm and $\|\cdot\|_1$ is the elementwise 1-norm.

The above formulation is well-supported in theory, in that it can *exactly* recover the low-rank and sparse component under certain separation conditions even though the formulation uses convex relaxation of rank and sparsity. More specifically, if $M = L_0 + S_0$, and in addition L_0 is sufficiently low-rank and incoherent, S_0 is sufficiently sparse, then the optimal solutions of (6) with $\lambda = \frac{1}{\sqrt{\max m,n}}$ obeys $L^* = L_0, S^* = S_0$.

Dozens of extensions of the RPCA theory and hundreds of RPCA applications has been developed since its invention in 2009. In addition, many applicable optimization algorithms have been tried and Alternating Direction Method of Multipliers (ADMM) (also called Augmented Lagrange Multiplier algorithm in the original text), with the few tweaks that we will describe later, is among the state-of-the-art solvers.

6.1 Deriving and implementing ADMM algorithm for RPCA

Refer to the ADMM algorithm described in Section 5 of `http://statweb.stanford.edu/~candes/papers/RobustPCA.pdf`. Basically, it constructs the augmented Lagrangian (with an additional term $\frac{\mu}{2}\|M - L - S\|_F^2$ comparing to the Lagrangian), and then iteratively solves:

(1) Minimize the augmented Lagrangian w.r.t. L, and update L.

(2) Minimize the augmented Lagrangian w.r.t. S, and update S.

(3) Dual gradient ascent with stepsize μ.

You can follow the following steps.

1. Please write down the formula of the above updates. You will see that (1) and (2) will be familiar proximal maps.

2. Then implement the algorithm in a language of your choice. The input of the algorithm is M, λ, and the output is (L, S). Within the algorithm, there are a number of numerical parameters to set. Including weight μ, tolerance δ and "maxiter". In addition, we will use an optional modification of ADMM, where we increase μ in every iteration via

$$\mu^{(t)} = \min\{\mu^{(t-1)}\rho, \bar{\mu}\}.$$

   $\bar{\mu}$ is an predefined upper bound and $\rho \geq 1$ defines how fast we want μ to increase over the iterations. When $\rho = 1$, this becomes the plain ADMM.

   Default parameters: By default, you can set $\lambda = 1/\max m, n$, $\rho = 1.5$, $\mu^{(0)} = 1.25/\|M\|_2$ where $\|M\|_2$ is the spectral norm (or operator norm) of M (you can compute it by finding the largest singular value) and $\bar{\mu} = \mu^{(0)} * 1e7$.

   Initialization: You should initialize L, S and the Lagrange multiplier (dual variable) Y with all 0 matrices.

   Stopping conditions: "maxiter" is set to be 40 and $\delta = 1e{-}7$. You stop when $\|M{-}L{-}S\|_F \leq \delta\|M\|_F$ or the number of iterations exceed "maxiter".

3. Debug your code by generating a random low-rank matrix of dimension $m \times n$ and rank-r using codes that look like the following:

```
m=500;n=600;r=10;
L0=randn(m,r)*randn(r,n);
mask=rand(m,r)<0.1;
num=sum(mask(:));
S0=zeros(m,n);
S0(mask)=8*(rand(num,1)-0.5);
M=L0+S0;
```

run your algorithm and then compare the solution against L0 and S0. Your algorithm should converge pretty quickly.

4. When you are sure that your code above is correct, modify the code with following trick for further speed-up. Note that the most expensive subroutine of the algorithm is (1), where we need to call SVD where the complexity is $O(n^3)$. However, we know that all singular values smaller than $1/\mu$ will be shrinked to 0 anyway and we shouldn't be computing them at all. We can substantially save the computation using partial SVD, if we know the cut-off point where singular value drop below $1/\mu$. Let the cut-off-point be k, and the matrix we take SVD be A, you will use the "lansvd" in PROPACK (the package `PROPACK.zip` is available on the course website) with the syntex $[U, S, V] = lansvd(A, k, 'L')$. If you are using R, you can use propack.svd as in `http://cran.r-project.org/web/packages/svd/svd.pdf`. The time complexity will be reduced to $O(k^2 n)$. This could be the difference between infeasible and feasible for large scale applications.

   Here, we will use a heuristic to estimate k adaptively. Since we are increasing μ in every iteration, the rank of L will in general be monotonically increasing in every iteration. A pseudo-code is given below:

```
a. Initialize k=10;
b. In every iteration:
    Update L based on partial svd with parameter k.
    if (rank(L)<k),
        set k=rank(L+1)+1
    else
        set k=min(n,k+floor(0.05*n))
```

   Another note is that when $k > 0.5n$, partial SVD is slower than full SVD, in which case you should simply use SVD.

6.2 Running RPCA for foreground-background detection in Surveillance video

Now you will apply your implemented code to a real-life problem! Available on the course website is `data_RPCA.zip`. Extract this into a folder, and you will find 400 BMP files. These are frames of a short surveillance video in Singapore Changi airport.

You will load these images into a big data matrix where ith column contains the vectorized ith image. Change the color image to grayscale image so that we do not need to deal with 3 different color channels and make sure each entry will be the pixel value in the 0-255 scale.

Using the default parameter, plug in this data matrix as M and solve RPCA.

1. Report the rank of the returned matrix L and number of non-zero entries in the returned S.

2. Visualize L and S respectively by making a video out of them. In Matlab, you can do this by simply reshaping L and S into a $144 * 176 * 400$ array and run "implay". Make sure that you find the max and min in L and S, then scale the entries into $[0, 1]$, otherwise the video will not look correct.

   Please have them concatenated so that we have background on the left hand side and foreground on the right hand side. The video will then have size $144 * 352 * 400$.

3. Upload the video on YouTube using private sharing and include the link to the video in your report.

6.3 Bonus: Combining RPCA and Matrix completion

Modify your code to handle missing data as in Eqn (1.5) of `http://statweb.stanford.edu/~candes/papers/RobustPCA.pdf`.

Taking 10% random pixels as observed and the remaining 10% missing in the above dataset and run missing-data-RPCA. Make another video of your result and show it in YouTube.

Hint: To get the foreground of every frame, you can subtract the completed columns in L from M. Directly using S may not have the foreground in S at all due to the subsampling.