Assignment 5: CS 754, Advanced Image Processing

Due: 16th April before 11:55 pm

Remember the honor code while submitting this (and every other) assignment. All members of the group should work on and understand all parts of the assignment. We will adopt a zero-tolerance policy against any violation.

Submission instructions: You should ideally type out all the answers in Word (with the equation editor) or using Latex. In either case, prepare a pdf file. Create a single zip or rar file containing the report, code and sample outputs and name it as follows: A5-IdNumberOfFirstStudent-IdNumberOfSecondStudent.zip. (If you are doing the assignment alone, the name of the zip file is A5-IdNumber.zip). Upload the file on moodle BEFORE 11:55 pm on the due date. The cutoff is 10 am on 17th April after which no assignments will be accepted. Note that only one student per group should upload their work on moodle. Please preserve a copy of all your work until the end of the semester. If you have difficulties, please do not hesitate to seek help from me.

Instructions for Coding Questions

1. Make a subfolder in the submission folder. Name the folder 'media'.
2. The directory structure should look like :

```
A5-<Roll_No_1>-<Roll_No_2>-<Roll_No_3>
    |
    |________media
    |________<other_file_1>
    |________<other_file_2>
    |________------------
    |________------------
    |________<other_file_n>
```

3. Read ANY image/video in ANY code from this folder(media) itself.
4. ALL the images/videos required for ANY code should be present in the folder 'media' itself, if your final compressed submission folder size DOES NOT EXCEED THE MOODLE SIZE LIMIT.
5. The TAs will copy all the images/video to the folder 'media' at the time of evaluation, if your final compressed submission folder DOES EXCEED THE MOODLE SIZE LIMIT. In this case leave the 'media' folder blank.
6. Please ensure that all the codes run at the click of a single go (RUN button) in MATLAB.
7. Please ensure that all the asked result images/videos, plots and graphs pop up at the click of a single go (RUN button) in MATLAB, while running the corresponding code for any question.
8. The result images/videos, plots and graphs should match those present in the report.

Questions

1. Implement the ALM algorithm for robust PCA. For this, create a matrix $\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}$ which is the sum of a low rank matrix $\boldsymbol{L}$ of rank r and a sparse matrix $\boldsymbol{S}$ with $s = f_s n_1 n_2$ non-zero elements where $f_s \in [0, 1]$. Create $\boldsymbol{L}$ using truncated SVD of Gaussian random matrices. The non-zero entries of $\boldsymbol{S}$ should be drawn from $\mathcal{N}(0, 9)$ and they should exist at randomly chosen indices. For this experiment, let $n_1 = 800, n_2 = 900$. Vary $r \in \{10, 30, 50, 75, 100, 125, 150, 200\}$ and $f_s \in \{0.01, 0.02, 0.03, 0.04, 0.06, 0.08, 0.1, 0.15\}$. Each value of (r, f_s), execute the algorithm 15 times and record the success probability. Plot the success probability as an image whose X and Y axes are r and f_s respectively (lower probability in darker color and higher probability in brighter color). Plot a colorbar using the `colorbar` function in MATLAB. Note that we define a reconstruction to be successful if $\|\boldsymbol{L} - \hat{\boldsymbol{L}}\|_F / \|\boldsymbol{L}\|_F \leq 0.001$ and $\|\boldsymbol{S} - \hat{\boldsymbol{S}}\|_F / \|\boldsymbol{S}\|_F \leq 0.001$ where $\hat{\boldsymbol{L}}, \hat{\boldsymbol{S}}$ are estimates of $\boldsymbol{L}, \boldsymbol{S}$ respectively. (Note, typically a single run of RPCA took just about 1.5-2 seconds on my desktop.) For any one successful and one unsuccessful (r, f_s) configuration, plot the ground truth and estimated low-rank and sparse matrices as separate grayscale images. Note that for RPCA, we are minimizing $\|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1$ subject to $\boldsymbol{M} = \boldsymbol{L} + \boldsymbol{S}$ where $\lambda = 1/\sqrt{\max(n_1, n_2)}$. [30 points]
Solutions: See homework folder for sample code. I set λ set as mentioned in the problem statement, and $\mu = 0.25 n_1 n_2 / (4\|\boldsymbol{M}\|_1)$. The RPCA algorithm should be correctly implemented with soft-thresholding and singular value thresholding done correctly. For correct implementation, there are 20 points. For executing the algorithm and displaying the results for different values of rank and sparsity, there are 10 marks. The error values to be displayed are relative error in L and S, not relative error in $L + S$ as there can be many combinations of L and S that add up to the same M, but we are specifically interested in estimating L and S. If the error is computed only over M and not over L or S, then 5 points are to be deducted. For rank of 50 and 5 percent sparsity, the relative errors in L and S are less than 10^{-4} in my code.

2. Read the wiki article on L1-norm PCA: `https://en.wikipedia.org/wiki/L1-norm_principal_component_analysis`. List any three fundamental ways in which robust PCA that we did in class differs from L1-norm PCA. [15 points]
Solution: Consider a data matrix X of size $d \times N$. Every column of X is a d-dimensional vector and there are N such vectors. There are major differences between ℓ_1-PCA and RPCA. (1) RPCA directly finds the low rank and sparse components that add up to the matrix X, whereas L1-norm PCA finds the low rank eigenvector matrix Q which maximizes $\|XQ\|_1$. (2) RPCA has theoretical performance guarantees, which are not mentioned in the wikipedia article for L1-norm PCA. (3) RPCA assumes that the non-zero elements of S are uniformly distributed, and its analysis under the case where all values in a single row or column are non-zero, is not presented. L1-PCA requires no such assumption on S. (4) The exact solution of RPCA can be computed faster than that of L1-norm PCA which requires at least $O(N^{rK-K+1})$ time where r is the rank of X and K is the number of columns in Q. The per-iteration cost of RPCA is significantly lower than this as it involves only soft-thresholding and singular-value thresholding which is at most $O(d^3)$ or $O(N^3)$. 5 points each for the three points of difference.

3. Perform a google search on any one advancement in the theory of RPCA. Choose any one paper, and explain how it advances the theory of RPCA. For this, write the statement of one key theorem from that paper and explain how it advances over the RPCA theory done in class. Mention the application where the theory is valid. [15 points]
Answer: The chosen paper is 'Real time principal components pursuit' at `https://www.ece.iastate.edu/~hanguo/RealTimeRPCA.pdf`. The original RPCA assumes that the support set of the sparse component S is randomly distributed. This is not true of moving objects in an actual scene where the object motion is correlated. Moreover, the support set of S will be highly clustered as well. The paper under consideration works in an incremental (or online) fashion, where the sparse part S is estimated for every new frame that arrives, making use of the correlation between the sparse part of neighboring frames. The key idea of this work is as follows: Given an initial estimate of the principal directions of the low rank part L, we causally keep estimating the sparse part at each time by solving a noisy compressive sensing type problem. The principal directions of the low rank part are updated every $K$the frame. In between the two updates, if new Principal Component directions appear, the 'noise' seen by the Compressive Sensing step may increase. This problem is solved, in part, by utilizing the time correlation model of the low rank part. This approach still

requires the singular vectors of the low rank part to be spread out, but it does not require i.i.d.-ness of either the sparse part or the low rank part. The key equations are (6) and (10) in the paper to estimate the sparse part in the new frames. There is no theorem stated in this paper. However the authors have many follow-up papers, such as `https://proceedings.mlr.press/v80/narayanamurthy18a/narayanamurthy18a.pdf` where a recovery guarantee for the support of S in every tth frame is given in Theorem 2.1 (the support set is denoted by $\mathcal{T}_t$). The advantage over the basic RPCA seen in class is (a) incremental updates, and (b) updates for correlated sparse part.

4. Consider that you learned a dictionary $\boldsymbol{D}$ to sparsely represent a certain class $\mathcal{S}$ of images - say handwritten alphabet or digit images. How will you convert $\boldsymbol{D}$ to another dictionary which will sparsely represent the following classes of images? Note that you are not allowed to learn the dictionary all over again, as it is time-consuming.

   (a) Class $\mathcal{S}_1$ which consists of images obtained by applying a known affine transform $\boldsymbol{A_1}$ to a subset of the images in class $\mathcal{S}$, and by applying another known affine transform $\boldsymbol{A_2}$ to the other subset. Assume that the images in $\mathcal{S}$ consisted of a foreground against a constant 0-valued background, and that the affine transformations $\boldsymbol{A_1}, \boldsymbol{A_2}$ do not cause the foreground to go outside the image canvas.

   (b) Class $\mathcal{S}_2$ which consists of images obtained by applying an intensity transformation $I^i_{new}(x,y) = \alpha(I^i_{old}(x,y))^2 + \beta(I^i_{old}(x,y)) + \gamma$ to the images in $\mathcal{S}$, where α, β, γ are known.

   (c) Class $\mathcal{S}_4$ which consists of images obtained by downsampling the images in $\mathcal{S}$ by a factor of k in both X and Y directions.

   (d) Class $\mathcal{S}_5$ which consists of images obtained by applying a blur kernel which is known to be a linear combination of blur kernels belonging to a known set $\mathcal{B}$, to the images in $\mathcal{S}$.

   (e) Class $\mathcal{S}_6$ which consists of 1D signals obtained by applying a Radon transform in a known angle θ to the images in $\mathcal{S}$. [4 x 5 = 20 points]

   Solution: Wherever reshaping is required, it must be mentioned otherwise 1.5 points to be deducted for each part where it is missing. Each answer must argue clearly why the suggested transformation will accurately create a new dictionary from D for the transformed images.

   (a) Class $\mathcal{S}_1$ which consists of images obtained by applying affine transform $\boldsymbol{A_1}$ to a subset of the images in class $\mathcal{S}$, and applying affine transform $\boldsymbol{A_2}$ to the other subset. Assume appropriate zero-padding and increase in the size of the image canvas owing to the translation. In this case, the dictionary $\boldsymbol{D_1}$ is created by (i) reshaping the column vectors of $\boldsymbol{D}$ to form images, (ii) applying the affine transforms, (iii) reshaping all these zero-padded images to form column vectors. The resultant dictionary will thus have twice the number of columns as $\boldsymbol{D}$.

   (b) Class $\mathcal{S}_2$ which consists of images obtained by applying an intensity transformation $I^i_{new}(x,y) = \alpha(I^i_{old}(x,y))^2 + \beta(I^i_{old}(x,y)) + \gamma$ to the images in $\mathcal{S}$, where α, β, γ are known. In this case we see that $\text{vec}(I^i_{new}) = \alpha(\boldsymbol{D\theta}).^2 + \beta\boldsymbol{D\theta} + \gamma = \alpha(\sum_{k=1}^K \boldsymbol{d_k}\theta_k) \cdot (\sum_{l=1}^K \boldsymbol{d_l}\theta_l) + \beta\sum_{k=1}^K \boldsymbol{d_k}\theta_k + \gamma\boldsymbol{1}$. Therefore, $\text{vec}(I^i_{new})$ can be expressed as a sparse linear combination of the columns of a dictionary $\boldsymbol{D_3}$ which contains the columns of $\boldsymbol{D}$, element-wise products of all pairs of columns of $\boldsymbol{D}$ (including pairs with identical members), and a column vector containing all ones.

   (c) Reshape the columns of $\boldsymbol{D}$ to form a 2D image. Then apply downsampling by a factor of k in both directions, and reshape the downsampled 2D images to form a vector and reassemble the dictionary. This gives rise to dictionary $\boldsymbol{D_4}$.

   (d) Class $\mathcal{S}_5$ which consists of images obtained by applying a blur kernel which is known to be a linear combination of blur kernels belonging to a known set $\mathcal{B}$, to the images in $\mathcal{S}$. Let $L = |\mathcal{B}|$. Here we have $\text{vec}(I_{5,i}) = \text{vec}[(\sum_{l=1}^L b_l) * I_i] = \text{vec}[(\sum_{l=1}^L b_l) * \text{reshape}(\boldsymbol{D\theta})] = \text{vec}[(\sum_{l=1}^L b_l) * \text{reshape}(\sum_{k=1}^K \boldsymbol{d_k}\theta_k)]$ $= \text{vec}[(\sum_{l=1}^L \sum_{k=1}^K b_l * \text{reshape}(\boldsymbol{d_k})\theta_k]$. The new dictionary $\boldsymbol{D_5}$ is created by convolving a reshaped version of each column vector of $\boldsymbol{D}$ with every blur kernel belong to $\mathcal{B}$.

   (e) Class $\mathcal{S}_6$ which consists of 1D signals obtained by applying a Radon transform in a known angle θ to the images in $\mathcal{S}$. The new dictionary $\boldsymbol{D_6}$ is obtained as follows: $\boldsymbol{d_{6,i}} = \mathcal{R}_\theta(\text{reshape}(\boldsymbol{d_i}))$ where 'reshape' converts a column vector to a 2D image.

5. Explain how you will minimize the following cost functions efficiently. In each case, mention any one application in image processing where the problem arises. [4 x 5 = 20 points]

   (a) $J_1(\boldsymbol{A_r}) = \|\boldsymbol{A} - \boldsymbol{A_r}\|_F^2$, where $\boldsymbol{A}$ is a known $m \times n$ matrix of rank greater than r, and $\boldsymbol{A_r}$ is a rank-r matrix, where $r < m, r < n$.

   (b) $J_2(\boldsymbol{R}) = \|\boldsymbol{A} - \boldsymbol{RB}\|_F^2$, where $\boldsymbol{A} \in \mathbb{R}^{n\times m}, \boldsymbol{B} \in \mathbb{R}^{n\times m}, \boldsymbol{R} \in \mathbb{R}^{n\times n}, m > n$ and $\boldsymbol{R}$ is constrained to be orthonormal. Note that $\boldsymbol{A}$ and $\boldsymbol{B}$ are both known.

   (c) $J_3(\boldsymbol{A}) = \|\boldsymbol{C} - \boldsymbol{A}\|_F^2 + \lambda\|\boldsymbol{A}\|_1$, where matrix $\boldsymbol{C}$ is known.

   (d) $J_4(\boldsymbol{A}) = \|\boldsymbol{C} - \boldsymbol{A}\|_F^2 + \lambda\|\boldsymbol{A}\|_*$, where matrix $\boldsymbol{C}$ is known.

   Solution:

   (a) The solution is obtained as follows from SVD: $\boldsymbol{A} = \boldsymbol{USV}^T$. Then $\boldsymbol{A_r} = \boldsymbol{U(1:r)S(1:r,1:r)V(1:r)}^T$.

   (b) We have $\boldsymbol{R} = \boldsymbol{UV}^T$ where $\boldsymbol{AB}^T = \boldsymbol{USV}^T$, i.e. from SVD of $\boldsymbol{AB}^T$.

   (c) We have $\boldsymbol{A} = \text{soft}(\boldsymbol{C}; \lambda)$ where soft(.) stands for the soft thresholding operation done in class.

   (d) $\boldsymbol{A} = \boldsymbol{U}\text{soft}(\boldsymbol{S}; \lambda)\boldsymbol{V}^T$ where $\boldsymbol{C} = \boldsymbol{USV}^T$. Here soft$(\boldsymbol{S}; \lambda)$ stands for soft thresholding on the singular values in $\boldsymbol{S}$.