CS754 Assignment 4 Report

Arpon Basu
Shashwat Garg

Spring 2022

Contents

Introduction

Welcome to our report on CS754 Assignment 4. We have tried to make this report comprehensive and self-contained. We hope reading this would give you a proper flowing description of our work, methods used and the results obtained.

Also note that we installed the `Image Processing Toolbox` in MATLAB for this assignment. Thus the grader is urged to install it if she wishes to run the code on her on her machine. Also note that some of our code may take a while to run because of the intensive nature of the computations involved.

Hope you enjoy reading the report. Here we go!

1 Problem 1

2 Problem 2

Let $D = \{d_1, d_2, ..., d_n\}$ be our dictionary, and let $I \in \mathcal{S}$ be an image in our dataset. Then by the theory of dictionary learning, $I = \sum_{i=1}^{n} \alpha_i d_i$ for some sequence of scalars $\alpha_1, \alpha_2, \ldots, \alpha_n$. In this problem, we are given various transformations T which are applied on the images of $\mathcal{S}$, and we have to propose a modified dictionary D_T such that our transformed images are still expressible as linear combination of "atoms" in D_T.
(a) From the theory taught in class and common image processing literature, we know that derivative filters on an image are obtained through **convolutions** of the image with some known convolution filter, ie:- $I_{\text{filter}} = I * G$, where G is the filter matrix. Now, it's not hard to see that a matrix convolution is a linear mapping, ie:- the **derivative filter 'f' is itself a linear transform** on the image.Then

$$f(I) = f(\sum_{i=1}^{n} \alpha_i d_i) = \sum_{i=1}^{n} f(\alpha_i d_i) = \sum_{i=1}^{n} \alpha_i f(d_i)$$

where all the equalities follow by the properties of linear transforms.
Thus our transformed dictionary is $D_T := \{f(d_1), f(d_2), \ldots, f(d_n)\}$, where f represents our derivative filter.
(b) Once again, note that rotation is a linear transform. In particular, the rotation of an image (matrix) I anticlockwise by an angle θ can be given as below

$$I_{\text{rotated}}(x, y) = I(x\cos\theta + y\sin\theta, y\cos\theta - x\sin\theta)$$

or equivalently as

$$I_{\text{rotated}}([x \;\; y]^T) = I(R_\theta^{-1}[x \;\; y]^T)$$

where the rotation matrix R_θ represents anticlockwise rotation by an angle θ

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Thus, once again, we have a linear transform in our hands and thus the new atoms of our transformed dictionary will just be the linear transform applied on the old dictionary itself, ie:- rotated versions of the old atoms. We will have two such dictionaries (for two distinct angles of rotation). Note that we must take the union of both dictionaries as an image, say rotated by β won't be expressible as a linear combination of dictionary atoms rotated by the angle α.
Thus our transformed dictionary is $D_T := \{d_{1\alpha}, d_{2\alpha}, \ldots, d_{n\alpha}, d_{1\beta}, d_{2\beta}, \ldots, d_{n\beta}\}$, where $d_{k\theta}$

represents the k^{th} atom rotated by θ.
(c) Assuming the relation holds pixel-wise, we have

$$I_{\text{new}}(x,y) = \alpha(I_{\text{old}}(x,y))^2 + \beta I_{\text{old}}(x,y) + \gamma$$

But we have

$$I_{\text{old}}(x,y) = \sum_{i=1}^{n} \alpha_i d_i(x,y)$$

$$\implies I_{\text{new}}(x,y) = \alpha(\sum_{i=1}^{n} \alpha_i d_i(x,y))^2 + \beta \sum_{i=1}^{n} \alpha_i d_i(x,y) + \gamma$$

$$\implies I_{\text{new}}(x,y) = \alpha(\sum_{i=1}^{n} \alpha_i^2 d_i^2(x,y) + 2 \sum_{1\leq i<j\leq n} \alpha_i \alpha_j d_i(x,y) d_j(x,y)) + \beta \sum_{i=1}^{n} \alpha_i d_i(x,y) + \gamma$$

Thus our new dictionary is clear:

- First class of items: $\{d_1^{\circ 2}, d_2^{\circ 2}, ..., d_n^{\circ 2}\}$, where $^{\circ 2}$ denotes **elementwise squaring**.
- Second class of items: $\{d_i \odot d_j\}_{1\leq i<j\leq n}$, where $\odot$ denotes **elementwise multiplication**.
- Third item: $J_{d\times d}$, where $d_i \in \mathbb{R}^{d\times d}$, where J is the matrix of all ones, ie:- $J =$ `ones(d)`, in MATLAB notation.

Thus our transformed dictionary is $D_T := \{d_1^{\circ 2}, d_2^{\circ 2}, ..., d_n^{\circ 2}, d_1 \odot d_2, \ldots, d_{n-1} \odot d_n, J_{d\times d}\}$.
(d) Blur kernel, like derivative filters, are also convolutions with kernel matrices and hence are linear transforms.
Thus our transformed dictionary is $D_T := \{f(d_1), f(d_2), \ldots, f(d_n)\}$, where f represents our blur kernel transformation.
(e) Similar to the idea used in part (b), we have to take an union of all the dictionaries, each applied with a single blur kernel. Thus let $\mathcal{B} := \{b_1, b_2, ..., b_l\}$ be the set of our blur *transforms*. Then our l dictionaries are $\{b_1(d_1), b_1(d_2), \ldots, b_1(d_n)\}$, $\{b_2(d_1), b_2(d_2), \ldots, b_2(d_n)\}$, ..., $\{b_l(d_1), b_l(d_2), \ldots, b_l(d_n)\}$, and
Thus our final dictionary is $D_T := \{b_1(d_1), \ldots, b_1(d_n), \ldots, b_l(d_1), b_l(d_2), \ldots, b_l(d_n)\}$.
(f) Note that the meaning of the notation for this part is slightly different from all other parts of this problem in the fact that our images and dictionary atoms are now assumed to the vectorized forms of the images they represented. Now, note that the Radon transform matrix can be denoted as R_θ, which is a square matrix with entries dependent on θ, which when multiplied with our image vectors gives us their Radon transform. Once again, as above, since this is a linear transform of the vectors, our new dictionary will be the Radon transform of the dictionary atoms. **Thus our new dictionary $D_T := \{Rd_1, Rd_2, \ldots, Rd_n\}$, where R is the Radon transform matrix corresponding to the angle θ.**
(g) Note that translating an image by a certain offset (x, y) basically means that we **pad the image, on the left and downwards, with x columns and y rows of zeros respectively**. Now, since we simply added an extra layer of padding of zeros, the linear combination relations still hold as the zero rows and columns obviously satisfy any linear relation and the old pixels also satisfy the relation too. Thus if we denote translation by the first offset to be t_1 and the second offset to be t_2, then, as we have seen many times before our **new dictionary will be $D_T := \{t_1(d_1), \ldots, t_1(d_n), t_2(d_1), t_2(d_2), \ldots, t_2(d_n)\}$.**

3 Problem 3

3.1 Part 1

We use the **Eckart-Young-Mirsky theorem** to find the **best rank r approximation to the given matrix** $A \in \mathbb{R}^{m\times n}$, ie:- A_r, which, as we have seen in the lecture, states that the best (in

terms of the Frobenius norm of the difference in between those two matrices) r-rank approximation is given by $A_r = U\Sigma_r V^*$, where $A = U\Sigma V^*$ is the **Singular Value Decomposition** of A, and Σ_r contains the r maximum singular values in Σ, ie:- we retain the r largest diagonal values of Σ and set the rest to zero.
Thus the minimizer of the objective function $J(A_r) = \|A - A_r\|_F^2$; $\text{rank}(A_r) = r$ is

$$\boldsymbol{A_r = U\Sigma_r V^*}$$

where $A = U\Sigma V^*$ is the SVD decomposition of A and Σ_r contains the r maximum singular values in Σ.
Applications of low rank approximations:

- **Robust PCA**
  - Note that in a surveillance video, the background remains stationary while our object of interest moves. Thus the background of the image can be treated as it's low rank component, which can then be identified using the algorithm outlined above.
  - In a similar vein as the idea above, one can remove occlusion by **completing** a low rank matrix to effectively "cut-out" the occluding object in the image.

3.2 Part 2

We algebraically simplify the objective function $J(R) = \|A - RB\|_F^2$ first to find out the optimum orthonormal R matrix. For that, note the following facts from linear algebra:

- $\|A\|_F^2 = \text{tr}(AA^T)$
- $\text{tr}(AB) = \text{tr}(BA)$, and thus any rearrangement of a matrix multiplication will still yield the same trace.
- $\text{tr}(A) = \text{tr}(A^T)$ for any square matrix A

Then

$$\begin{aligned}
\|A - RB\|_F^2 = \text{tr}((A - RB)(A - RB)^T) &= \text{tr}((A - RB)(A^T - B^T R^T)) \\
&= \text{tr}(AA^T - RBA^T - AB^T R^T + RBB^T R^T) \\
&= \text{tr}(AA^T) - \text{tr}(RBA^T) - \text{tr}((RBA^T)^T) + \text{tr}(RBB^T R^T) \\
&= \text{constant} - \text{tr}(RBA^T) - \text{tr}(RBA^T) + \text{tr}(R^T RBB^T) \\
&= \text{constant} - 2 \cdot \text{tr}(RBA^T) + \text{tr}(BB^T) \\
&= \text{constant} - 2 \cdot \text{tr}(RBA^T)
\end{aligned}$$

Thus minimizing $\|A - RB\|_F^2$ is equivalent to maximizing $\text{tr}(RBA^T)$.
Now, let BA^T be equal to C, and let $C = U\Sigma V^*$ be the **Singular Value Decomposition** of C. Then

$$\text{tr}(RBA^T) = \text{tr}(RC) = \text{tr}(RU\Sigma V^*) = \text{tr}(V^* RU\Sigma)$$

Now, note that since R, U and V are all orthonormal, $M = V^* RU$ is orthonormal too, and thus

$$\text{tr}(V^* RU\Sigma) = \text{tr}(M\Sigma) = \sum_{i=1}^{n} m_{ii}\sigma_i \leq \sum_{i=1}^{n} |m_{ii}|\sigma_i$$

with the last inequality following by the triangle inequality (and note that $|\sigma_i| = \sigma_i$ since singular values are non-negative).
However, since M is orthonormal, ie:- all of it's column's norms are 1, we have that $|m_{ii}| \leq 1$, and

thus the maxima of $\text{tr}(V^*RU\Sigma)$ is achieved when all $|m_{ii}| = 1$. But that can happen only when $M = I$, and thus

$$V^*RU = I$$
$$\implies \boldsymbol{R = VU^*}$$

Thus the best orthonormal transform that maps B to A is given by VU^*, where U and V are obtained from the SVD of $BA^T = U\Sigma V^*$.
This problem is also widely encountered in image processing literature. We ourselves came across one such example in class, namely, in 3D Tomography under unknown angles, wherein we try to determine what rotation best maps one image to another, and there is where we note that orthonormal matrices basically represent unscaled rotation in higher dimensions, from which the problem of minimizing $\|A - RB\|_F^2$ under an orthonormal R constraint arises.

4 Problem 4

5 Problem 5