Section 4 Problem

In this problem, you will apply the knowledge of numerical algebra to compress an image. The idea comes from a row-rank approximation of SVD. A matrix $\mathbf{A} \in \mathbf{R}^{m \times n}$ can be compressed using the low-rank approximation property of the SVD. The approximation algorithm is given in Algorithm 1.

Algorithm 1: Low-Rank Approximation using SVD

Input: $\mathbf{A} \in \mathbf{R}^{m \times n}$ of rank r and approximation rank k, where $(k \leq r)$

Output: $\mathbf{A}_k \in \mathbf{R}^{m \times n}$, the optimal rank k approximation to $\mathbf{A}$.

1. Compute (thin) SVD of $\mathbf{A} = \mathbf{U\Sigma V}^T$, where $\mathbf{U} \in \mathbf{R}^{m \times r}, \mathbf{\Sigma} \in \mathbf{R}^{r \times r}, \mathbf{V} \in \mathbf{R}^{n \times r}$
2. $\mathbf{A}_k = \mathbf{U}(:, 1:k)\mathbf{\Sigma}(1:k, 1:k)\mathbf{V}(:, 1:k)^T$

Notice the low-rank approximation, $\mathbf{A}_k$, and the original matrix, $\mathbf{A}$, are the same size. To actually achieve compression, the truncated singular factors, $\mathbf{U}(:, 1:k) \in \mathbf{R}^{m \times k}$, $\mathbf{\Sigma}(1:k, 1:k) \in \mathbf{R}^{k \times k}$, $\mathbf{V}(:, 1:k) \in \mathbf{R}^{n \times k}$, should be stored. As $\mathbf{\Sigma}$ is *diagonal*, the required storage is $(m+n+1)k$ doubles. The original matrix requires storing mn doubles. Therefore, the compression is useful only if $k < \dfrac{mn}{m+n+1}$.

Recall from lecture that the SVD is among the most expensive matrix factorizations in numerical linear algebra. Many SVD approximations have been developed; a particularly interesting one from Halko (2011) is given in Algorithm 2. This algorithm computes a rank p approximation to the SVD of a matrix. This approximation rank is *different* than the compression rank from Algorithm 1.

Algorithm 2: Low-Rank Probabilistic SVD Approximation

Input: $\mathbf{A} \in \mathbf{R}^{m \times n}$ (usually $n \ll m$), approximation rank p, and number of power iterations q

Output: Approximate SVD of $\mathbf{A} \approx \mathbf{U\Sigma V}^T$

1. Generate $n \times p$ Gaussian test matrix $\mathbf{\Omega}$
2. Form $\mathbf{Y} = (\mathbf{AA}^T)^q\mathbf{A\Omega}$
3. Compute QR factorization of $\mathbf{Y} : \mathbf{Y} = \mathbf{QR}$
4. Form $\mathbf{B} = \mathbf{Q}^T\mathbf{A}$
5. Compute SVD of $\mathbf{B} = \tilde{\mathbf{U}}\mathbf{\Sigma V}^T$
6. Set $\mathbf{U} = \mathbf{Q}\tilde{\mathbf{U}}$

In this problem, you will use the Singular Value Decomposition (SVD) to compress the image given in `palm.png`. You will be given the function `get_rgb.m` that accepts the filename of an image and returns the RGB matrices defining the image, stacked vertically to form a single skinny matrix. Also, the function `plot_image_rgb.m` will take stacked RGB matrix and (optionally) a handle to an `axes` graphics object (default will use `gca`) and plot the corresponding image (the input to `plot_image_rgb.m` is the output of `get_rgb.m`). To "compress" the image, compress the stacked

RGB matrix and pass to `plot_image_rgb.m`. Another possibility involves compressing the RGB matrices individually. For this assignment, use the stacked version for the compression.

Task 1

- Download `get_rgb.m, plot_image_rgb.m`, and the image (Palm Drive).
- Consider an image of $m \times n$ pixels. The `R, G, B` matrices will be of dimension $m \times n$.
- Compress to ranks $[5, 10, 15, 20, 25, 50, 100, 200, \min(3m, n)]$ and visualize resulting images.
- Use `svd` for the SVD step in Algorithm 1.

Task 2

Implement the probabilistic SVD algorithm from Algorithm 2 and use this for the SVD step in Algorithm 1. Good values for the approximation rank will depend on the image you choose to compress. Notice that a rank p approximation to the SVD in Algorithm 2 only admits low-rank approximations of up to rank p (cannot go up to $\min(3m, n)$). In this case, the ranks $[5, 10, 15, 20, 25, 50, 100, 200, \min(3m, n)]$ cannot all be used. Choose the ranks $[5, 10, 15, 20, 25, 50]$ instead. Choose $q = 1, 5, 10$.

Task 3

Plot $v(k)$ for $k \in \{1, 2, \dots, \min(3m, n)\}$ for the stacked RGB matrix, where

$$v(k) = 1 - \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{j=1}^{\min(3m,n)} \sigma_j^2}$$

and σ_i `is the` ith singular value of the matrix. This gives an indication of the compressibility of the image. A fast singular value decay implies the image can be compressed to a small rank with minimal loss in quality. Use the singular values by generated by `svd` to complete this part.

Task 4

Repeat using the singular values generated by Algorithm 2. In this case, we choose the maximum value of k to be 50. Hence, in this case

$$v(k) = 1 - \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{j=1}^{50} \sigma_j^2}$$

and σ_i is the ith singular value from Algorithm 2 with $q = 1$.

Checkpoint

Please answer the following questions and put the answers in the EdX page:

(A) As the rank in Task 1 increases, the compressed image becomes ....

(B) As q in Task 2 increases, the compressed image becomes ....

(C) What is $v(1)$ from Task 3? Rounded the answer to the nearest thousandth.

(D) What is $v(1)$ from Task 4? Rounded the answer to the nearest thousandth.