Problem 1 (15 Points): Dimensionality Reduction via PCA

Suppose you have the raw data points in 2-dimensional space shown in the following table:

data	x	y
1	5.51	5.35
2	20.82	24.03
3	-0.77	-0.57
4	19.30	19.38
5	14.24	12.77
6	9.74	9.68
7	11.59	12.06
8	-6.08	-5.22

You want to reduce the data into a 1-dim space. You are given the first principal component (0.694, 0.720).

1. **4 Points.** What is the representation for data#1 and data#8 in the first principal space?
2. **4 Points.** What are the xy coordinates in the raw space reconstructed using this first principal for data#1 and data#8?
3. **4 Points.** What is the representation for data#1 and data#8 in the second principal space?
4. **3 Points.** What is the reconstruction error if you use two principal components to represent raw data?

Problem 2 (20 Points): SVD: Image Compression

In this experiment, we will use the singular value decomposition (SVD) as a tool for compressing raw image data. This is not how images are actually compressed; for example, JPEG compression algorithms do more fancy (and interesting) computations. However, the idea is similar: if we are willing to tolerate a certain amount of distortion, then we can get away with a much more concise data representation.

1. **3 Points.** Read the given image file (`'mandrill_color.png'`), and convert it into grayscale by averaging the R,G,B values for each pixel. Your image is now a 288×288 matrix; call it $\mathbf{X}$.
2. **4 Points.** Perform an SVD of $\mathbf{X}$ to obtain the decomposition $\{\mathbf{U}, \Sigma, \mathbf{V}\}$. Plot the singular values (i.e., the diagonal entries of Σ) in decreasing order.
3. **5 Points.** Choose $k = 10$, and reconstruct an approximation of $\mathbf{X}$ using the top k singular values and vectors, $\mathbf{U}_k$, $\mathbf{V}_k$, and Σ_k. Display this approximation, and calculate how many numbers you needed to store this approximate image representation. Divide by the original size of $\mathbf{X}$ to get the compression ratio.
4. **8 Points.** Repeat this experiment for $k = 20, 40, 60$. Display these images, and report their compression ratios in the form of a table. Is there any benefit in going for higher k?

Problem 3 (20 Points): PCA: Best Places to Live

The *Places Rated Almanac*, written by Boyer and Savageau and published by McNally, rates the livability of several US cities according to nine factors: climate, housing, healthcare, crime, transportation, education, arts, recreation, and economic welfare. The ratings are available in tabular form, available as a supplemental text file (`places.txt`). Except for housing and crime, higher ratings indicate better quality of life.

Let us use PCA to interpret this data better.

1. **2 Points.** Read the data and construct a table with 9 columns containing the numerical ratings. (Ignore the last 5 columns – they consist auxiliary information such as longitude/latitude, state, etc.)
2. **2 Points.** Replace each value in the matrix by its base-10 logarithm. (This pre-processing is done for convenience since the numerical range of the ratings is large.) You should now have a data matrix X whose rows are 9-dimensional vectors representing the different cities.

3. **4 Points.** Perform PCA on the data. Remember to center the data points first by computing the mean data vector μ and subtracting it from every point. With the centered data matrix, do an SVD and compute the principal components.
4. **3 Points.** Write down the first two principal components v_1 and v_2. Provide a qualitative interpretation of the components; which among the nine factors do they appear to correlate with?
5. **3 Points.** Project the data points onto the first two principal components. (That is, compute the highest 2 scores of each of the data points.) Plot the scores as a 2D scatter plot. Which cities correspond to outliers in this scatter plot?
6. **6 Points.** Repeat Steps 2-5, but with a slightly different data matrix – instead of computing the base-10 logarithm, use the normalized z-score of each data point. How do your answers change?

Problem 4 (20 Points): Manifold Learning: Order the Faces

The dataset (`face.mat`) contains 33 faces of the same person ($Y \in \mathbb{R}^{112\times92\times33}$) in different angles. You may create a data matrix $X \in \mathbb{R}^{n\times p}$, where $n = 33, p = 112 \times 92 = 10304$ (e.g., `X=reshape(Y,[10304,33])';` in MATLAB).

1. **5 Points.** Explore the MDS-embedding of the 33 faces on top two eigenvectors: order the faces according to the top 1st eigenvector and visualize your results with figures.
2. **5 Points.** Explore the ISOMAP-embedding of the 33 faces on the $k = 5$ nearest neighbor graph and compare it against the MDS results. Note: you may try Tenenbaum's Matlab code (`isomapII.m`).
3. **5 Points.** Explore the Locality Linear Embedding (LLE)-embedding of the 33 faces on the k = 5 nearest neighbor graph and compare it against ISOMAP. Note: you may try the following Matlab code (`lle.m`).
4. **5 Points.** Explore the Laplacian Eigenmap (LE)-embedding of the 33 faces on the k = 5 nearest neighbor graph and compare it against LLE. Note: you may try the following Matlab code (`le.m`).

Problem 5 (25 Points): Random Projections

In this problem, we numerically verify the Johnson-Lindenstrauss Lemma. Recall its statement: for any set $\mathbf{X}$ of n points in d dimensions, there exists a matrix $\mathbf{A}$ with merely $m = 4 \log n/\epsilon^2$ rows such that for all $\mathbf{u}, \mathbf{v} \in \mathbf{X}$:

$$(1-\epsilon)\|\mathbf{u}-\mathbf{v}\|_2^2 \le \|\mathbf{A}\mathbf{u}-\mathbf{A}\mathbf{v}\|_2^2 \le (1+\epsilon)\|\mathbf{u}-\mathbf{v}\|_2^2$$

In particular, m is independent of d. Moreover, $\mathbf{A}$ can be constructed by choosing $m \times d$ i.i.d. entries from a zero mean Gaussian with variance $1/m$.

1. **2 Points.** Construct any data matrix $\mathbf{X}$ of your choice with parameters $n = 10, d = 5000$ (For instance, this could be any n columns of the identity matrix $\mathbf{I}_{d\times d}$). Fix $\epsilon = 0.1$ and compute the embedding dimension m by plugging in the formula above.
2. **7 Points.** Construct a random projection matrix $\mathbf{A}$ of size $m \times d$, and compare all pairwise (squared) distances $\|\mathbf{u}-\mathbf{v}\|_2^2$ with the distances between the projections$\|\mathbf{A}\mathbf{u}-\mathbf{A}\mathbf{v}\|_2^2$. Does the Lemma hold (i.e., for every pair of data points, is the projection distance is within 10% of the original distance)?
3. **8 Points.** Repeat the above steps by increasing d as a factor 2 each time with m and n fixed. Make d larger and larger until your system runs out of memory. Verify that the Lemma holds in each case.
4. **8 Points.** Repeat the above steps by increasing n as a factor 2 each time with d fixed. Make n larger and larger until your system runs out of memory. Verify that the Lemma holds in each case.