Dan-Dominic Staicu
311CAb
2023
Numerical Methods 2nd Homework

Task 1 - Singular Value Decomposition (SVD)

This task implements a a function used for image compression using using Singular Value Decomposition (SVD); the function returns a new image which is an approximation of the original image. In mathematics, SVD is a factorization of a real or complex matrix that generalizes the eigendecomposition of a square normal matrix to any matrix.

"photo" is the input image that has to be compressed;
"k" is the compression level, which specifies the number of singular values to retain from the SVD;

Choosing a smaller k will result in higher compression, but will also result in more loss of image quality. Conversely, a larger k will give a result closer to the original image, but with less compression. So, the choice of k is a trade-off between compression level and preservation of image quality.

The function uses the SVD function in order to perform Singular Value Decomposition on the matrix. This will result in 3 matrices:

* U: a matrix containing the left singular vectors
* S: a diagonal matrix containing the singular values
* V: a matrix containing the right singular vectors

After that, it calculates the new matrices U_k, S_k, and V_k which are the first k columns of U, the first k singular values of S, and the first k columns of V respectively.

Task 2 - Image Compression using Principal Component Analysis (PCA)

This function implements a function used for image compression using Principal Component Analysis (PCA); the function returns a new image which is an approximation of the original image.

This algorithm calculates the principal components using SVD.

The original image is represented as a matrix which is then normalized by subtracting the mean of each row.

The algorithm then computes the Singular Value Decomposition (SVD) of the matrix.

The normalized matrix is then scaled by the square root of the number of columns in the image minus one, and the SVD of this scaled matrix (denoted as Z) is calculated. The singular value decomposition splits the original matrix into three separate matrices (U, S, and V).

Next, the function constructs a new matrix (W) from the first 'pcs' columns of the matrix V. 'pcs' is a parameter that denotes the number of principal components.

The matrix Y is then calculated by multiplying the transpose of W with the normalized photo matrix. The new approximation of the photo matrix, 'new_X', is calculated by multiplying W with Y and adding back the mean of each row.

Finally, this approximation matrix is converted back to an 8-bit unsigned integer type, to return a valid image. The result is an approximation of the original image, but using less data, demonstrating the power of PCA for image compression.

Task 3

This function implements a function used for image compression using Principal Component Analysis (PCA); the function returns a new image which is an approximation of the original image.

This algorithm calculates the principal components using covariance matrix;

The original image is represented as a matrix which is then normalized by subtracting the mean of each row.

The algorithm then computes the covariance matrix of the normalized image matrix. In statistics, the covariance matrix is a matrix that captures the variance of a dataset along with the covariance of the dataset's variables. The covariance matrix is used here to reveal the underlying data structure of the image.

Eigenvalues and eigenvectors of the covariance matrix are then calculated. The eigenvectors of the covariance matrix represent the principal components of the image. The eigenvectors are sorted in descending order by their corresponding eigenvalues. The eigenvector with the largest eigenvalue is the first principal component, the eigenvector with the second largest eigenvalue is the second principal component, and so on. The eigenvalues represent the variance of the data along the corresponding eigenvector. The eigenvector with the largest eigenvalue has the largest variance, and thus contains the most information about the data.

The eigenvalues are sorted in descending order and the corresponding eigenvectors are rearranged accordingly. These eigenvectors, also known as principal components, are the directions in which the original data varies the most. By keeping only the principal components corresponding to the largest eigenvalues, we effectively compress the image data.

A new matrix (V) is created from the first 'pcs' columns of the sorted eigenvectors. 'pcs' is a parameter that denotes the number of principal components.

The matrix Y is then calculated by changing the basis of the normalized photo matrix using the transpose of V. The new approximation of the photo matrix, 'new_X', is calculated by multiplying V with Y and adding back the mean of each row.

Finally, the approximation matrix is converted back to an 8-bit unsigned integer type, to return a valid image. The result is an approximation of the original image, but using less data, demonstrating the power of PCA for image compression.

The **“pcs”** parameter is really important in tasks 2 and 3. More principal components result in a better quality image but less data reduction. Conversely, fewer principal components result in more data reduction but a potentially lower quality image.

Task 4 - Handwritten Digit Recognition

This task implements a handwritten digit recognition algorithm using PCA (Principal Component Analysis) for data size reduction and k-Nearest Neighbors for classification; data used come from MNIST set that contains images of handwritten digits, totaling 70000 samples (60000 for training and 10000 for testing); Each photo is black-and-white and has a size of 28x28 pixels.

This implementation uses 6 functions:

1. prepare_data loads training data from "mnist.mat" file and prepares them for analysis. This involves converting each image into a one-dimensional vector of 784 elements (28x28 pixels) and storing them in a matrix for further processing

2. visualise_image unction allows the visualization of a specific image from the dataset. Given an index, it retrieves the corresponding image vector from the dataset, reshapes it back into a 28x28 matrix, and displays it using a grayscale colormap.

3. magic_with_pca function applies PCA on the dataset. First, it calculates the mean of each pixel across the training images and then subtracts this mean from each pixel, thereby normalizing the data. Then, it calculates the covariance matrix and, finally, the eigenvalues and eigenvectors of this matrix. These eigenvectors, sorted by their corresponding eigenvalues, represent the principal components. The function returns a reduced dataset by projecting the original data onto the space spanned by the chosen number of principal components.

4. prepare_photo prepares a test image for prediction. It takes an input image, inverts its colors (if necessary), reshapes it into a 784-length vector, and normalizes it by subtracting the mean calculated in the magic_with_pca function. The function returns the reduced vector after projecting it onto the same principal component space.

5. KNN function implements the k-Nearest Neighbors algorithm to make the final prediction. Given a test image, it calculates the Euclidean distance to each image in the training set, finds the 'k' nearest neighbors, and assigns the most common label among these neighbors as the predicted label for the test image.

6. classify_image combines all the previous functions to produce a prediction. It takes a test image as input, applies all the data preparation, PCA, and KNN steps, and finally, returns the predicted digit.

The number of principal components selected in PCA directly impacts the quality of the reconstructed image. However, selecting a very high number of principal components can lead to a significant increase in the computational complexity of the algorithm. In this case, we chose a number of 23 principal components, which allows a reasonable trade-off between image quality and computational complexity.