2 Principal Component Analysis

In this exercise, you will use principal component analysis (PCA) to perform dimensionality reduction. You will first experiment with an example 2D dataset to get intuition on how PCA works, and then use it on a bigger dataset of 5000 face image dataset.

The provided script, `ex7_pca.m`, will help you step through the first half of the exercise.

2.1 Example Dataset

To help you understand how PCA works, you will first start with a 2D dataset which has one direction of large variation and one of smaller variation. The script `ex7_pca.m` will plot the training data (Figure 4). In this part of the exercise, you will visualize what happens when you use PCA to reduce the data from 2D to 1D. In practice, you might want to reduce data from 256 to 50 dimensions, say; but using lower dimensional data in this example allows us to visualize the algorithms better.

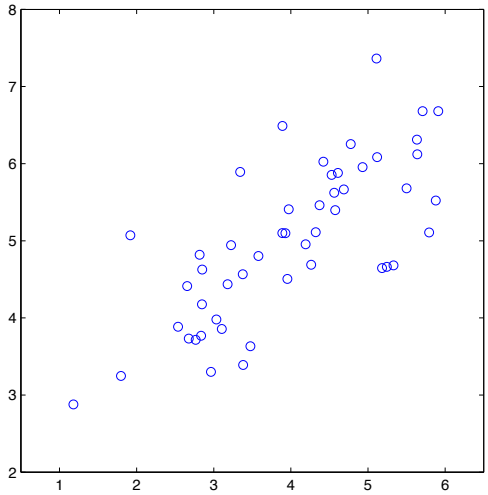

Figure 4: Example Dataset 1

2.2 Implementing PCA

In this part of the exercise, you will implement PCA. PCA consists of two computational steps: First, you compute the covariance matrix of the

data. Then, you use Octave's `SVD` function to compute the eigenvectors $U_1, U_2, \ldots, U_n$. These will correspond to the principal components of variation in the data.

Before using PCA, it is important to first normalize the data by subtracting the mean value of each feature from the dataset, and scaling each dimension so that they are in the same range. In the provided script `ex7_pca.m`, this normalization has been performed for you using the `featureNormalize` function.

After normalizing the data, you can run PCA to compute the principal components. You task is to complete the code in `pca.m` to compute the principal components of the dataset. First, you should compute the covariance matrix of the data, which is given by:

$$\Sigma = \frac{1}{m} X^T X$$

where X is the data matrix with examples in rows, and m is the number of examples. Note that Σ is a $n \times n$ matrix and not the summation operator.

After computing the covariance matrix, you can run SVD on it to compute the principal components. In Octave, you can run SVD with the following command: `[U, S, V] = svd(Sigma)`, where `U` will contain the principal components and `S` will contain a diagonal matrix.

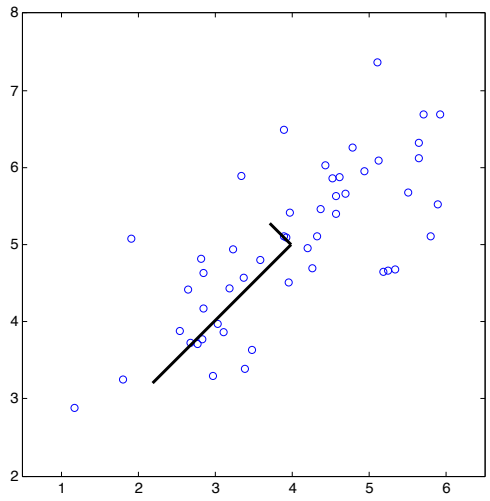

Figure 5: Computed eigenvectors of the dataset

Once you have completed `pca.m`, the `ex7_pca.m` script will run PCA on the example dataset and plot the corresponding principal components found

(Figure 5). The script will also output the top principal component (eigenvector) found, and you should expect to see an output of about `[-0.707 -0.707]`. (It is possible that Octave may instead output the negative of this, since U_1 and $-U_1$ are equally valid choices for the first principal component.)

2.3 Dimensionality Reduction with PCA

After computing the principal components, you can use them to reduce the feature dimension of your dataset by projecting each example onto a lower dimensional space, $x^{(i)} \rightarrow z^{(i)}$ (e.g., projecting the data from 2D to 1D). In this part of the exercise, you will use the eigenvectors returned by PCA and project the example dataset into a 1-dimensional space.

In practice, if you were using a learning algorithm such as linear regression or perhaps neural networks, you could now use the projected data instead of the original data. By using the projected data, you can train your model faster as there are less dimensions in the input.

2.3.1 Projecting the data onto the principal components

You should now complete the code in `projectData.m`. Specifically, you are given a dataset `X`, the principal components `U`, and the desired number of dimensions to reduce to `K`. You should project each example in `X` onto the top `K` components in `U`. Note that the top `K` components in `U` are given by the first `K` columns of U, that is `U_reduce = U(:, 1:K)`.

Once you have completed the code in `projectData.m`, `ex7_pca.m` will project the first example onto the first dimension and you should see a value of about `1.481` (or possibly `-1.481`, if you got $-U_1$ instead of U_1).

2.3.2 Reconstructing an approximation of the data

After projecting the data onto the lower dimensional space, you can approximately recover the data by projecting them back onto the original high dimensional space. Your task is to complete `recoverData.m` to project each example in `Z` back onto the original space and return the recovered approximation in `X_rec`.

Once you have completed the code in `projectData.m`, `ex7_pca.m` will recover an approximation of the first example and you should see a value of about `[-1.047 -1.047]`.

2.3.3 Visualizing the projections

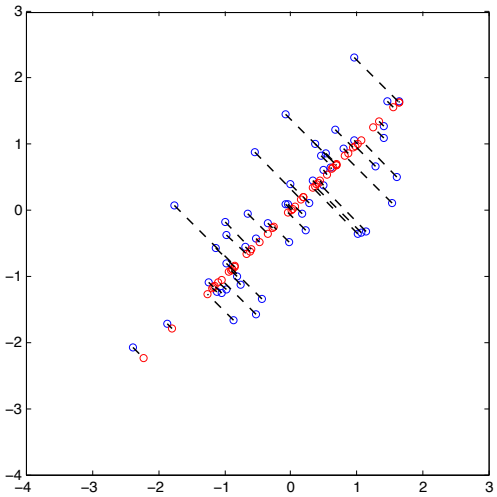

Figure 6: The normalized and projected data after PCA.

After completing both `projectData` and `recoverData`, `ex7_pca.m` will now perform both the projection and approximate reconstruction to show how the projection affects the data. In Figure 6, the original data points are indicated with the blue circles, while the projected data points are indicated with the red circles. The projection effectively only retains the information in the direction given by U_1.

2.4 Face Image Dataset

In this part of the exercise, you will run PCA on face images to see how it can be used in practice for dimension reduction. The dataset `ex7faces.mat` contains a dataset[2] `X` of face images, each 32×32 in grayscale. Each row of `X` corresponds to one face image (a row vector of length 1024). The next

[2]This dataset was based on a cropped version of the labeled faces in the wild dataset.

step in `ex7_pca.m` will load and visualize the first 100 of these face images (Figure 7).

Figure 7: Faces dataset

2.4.1 PCA on Faces

To run PCA on the face dataset, we first normalize the dataset by subtracting the mean of each feature from the data matrix `X`. The script `ex7_pca.m` will do this for you and then run your PCA code. After running PCA, you will obtain the principal components of the dataset. Notice that each principal component in `U` (each row) is a vector of length n (where for the face dataset, $n = 1024$). It turns out that we can visualize these principal components by reshaping each of them into a 32×32 matrix that corresponds to the pixels in the original dataset. The script `ex7_pca.m` displays the first 36 principal components that describe the largest variations (Figure 8). If you want, you can also change the code to display more principal components to see how they capture more and more details.

2.4.2 Dimensionality Reduction

Now that you have computed the principal components for the face dataset, you can use it to reduce the dimension of the face dataset. This allows you to use your learning algorithm with a smaller input size (e.g., 100 dimensions) instead of the original 1024 dimensions. This can help speed up your learning algorithm.

Figure 8: Principal components on the face dataset

Figure 9: Original images of faces and ones reconstructed from only the top 100 principal components.

The next part in `ex7_pca.m` will project the face dataset onto only the first 100 principal components. Concretely, each face image is now described by a vector $z^{(i)} \in \mathbb{R}^{100}$.

To understand what is lost in the dimension reduction, you can recover the data using only the projected dataset. In `ex7_pca.m`, an approximate recovery of the data is performed and the original and projected face images are displayed side by side (Figure 9). From the reconstruction, you can observe that the general structure and appearance of the face are kept while the fine details are lost. This is a remarkable reduction (more than 10×) in

the dataset size that can help speed up your learning algorithm significantly. For example, if you were training a neural network to perform person recognition (gven a face image, predict the identitfy of the person), you can use the dimension reduced input of only a 100 dimensions instead of the original pixels.

2.5 Optional (ungraded) exercise: PCA for visualization

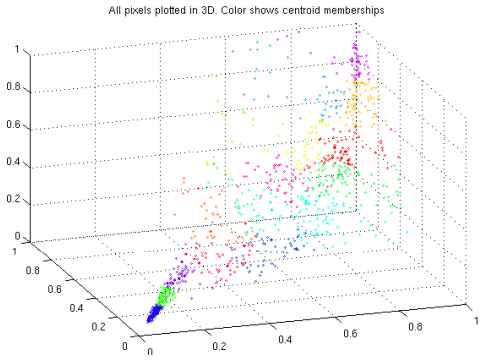

Figure 10: Original data in 3D

In the earlier K-means image compression exercise, you used the K-means algorithm in the 3-dimensional RGB space. In the last part of the `ex7_pca.m` script, we have provided code to visualize the final pixel assignments in this 3D space using the `scatter3` function. Each data point is colored according to the cluster it has been assigned to. You can drag your mouse on the figure to rotate and inspect this data in 3 dimensions.

It turns out that visualizing datasets in 3 dimensions or greater can be cumbersome. Therefore, it is often desirable to only display the data in 2D even at the cost of losing some information. In practice, PCA is often used to reduce the dimensionality of data for visualization purposes. In the next part of `ex7_pca.m`, the script will apply your implementation of PCA to the 3-dimensional data to reduce it to 2 dimensions and visualize the result in a 2D scatter plot. The PCA projection can be thought of as a rotation that selects the view that maximizes the spread of the data, which often corresponds to the "best" view.

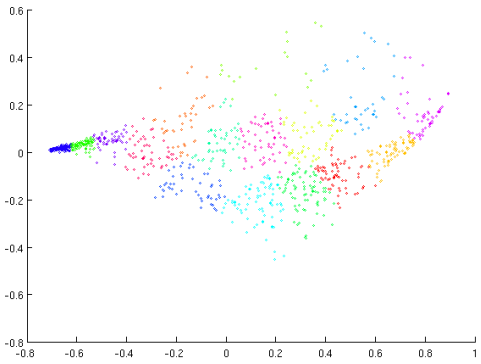

Figure 11: 2D visualization produced using PCA