EE 321 - Fall 2020
Project 2: Dimensionality Reduction and PCA, Due Date: October 2, 2020

This project introduces basic concepts in machine learning that include classification, dimensionality, and visualization. Submit a project report that includes a brief description of what you did, and provide the deliverables listed in each of the sections below. Make sure the report is clearly organized with all figures clearly numbered and labeled. Each figure should be referred to in the text and should describe the takeaways from the figure. Remember that the project grade depends on both the deliverables and the quality of the project report.

To get started, load into the programming environment, the data file, `gen_data.csv`. Note that in what follows, I will provide some sample code using MATLAB. Python users should be able to follow along with appropriate modifications.

1 Examine the Data

First, load the datafile into MATLAB. Use the following command:

```
>> data = readmatrix("gen_data.csv");
```

This will load the data stored in the `.csv` file into the variable `data`.

Next, we need to understand the data. How many dimensions do the data have? What do the data look like?

Let's answer these questions one at a time:

- To find the dimensions of the data, use the command:

  ```
  >> [r, c] = size(data)
  ```

  Notice that this line does not have a semicolon at the end. Leaving out the semicolon displays the result (also called an 'echo'). The result displays the dimensions of the array. This is a two-dimensional array, a matrix, with 200 rows and 3 columns. Though the matrix is two-dimensional, the data contained are in three dimensions, since we have 200 points (coordinates) in three dimensional space described.

- Let's visualize the data. Since we have three dimensional data, we use the following command:

  ```
  >> scatter3(data(:,1),data(:,2),data(:,3));
  ```

  which provides a plot similar to the one shown in Figure 1. What does your data look like?

2 Dimensionality Reduction

Let's now try to determine if we really need three dimensions to represent this data. The first step is to perform principal component analysis (PCA).

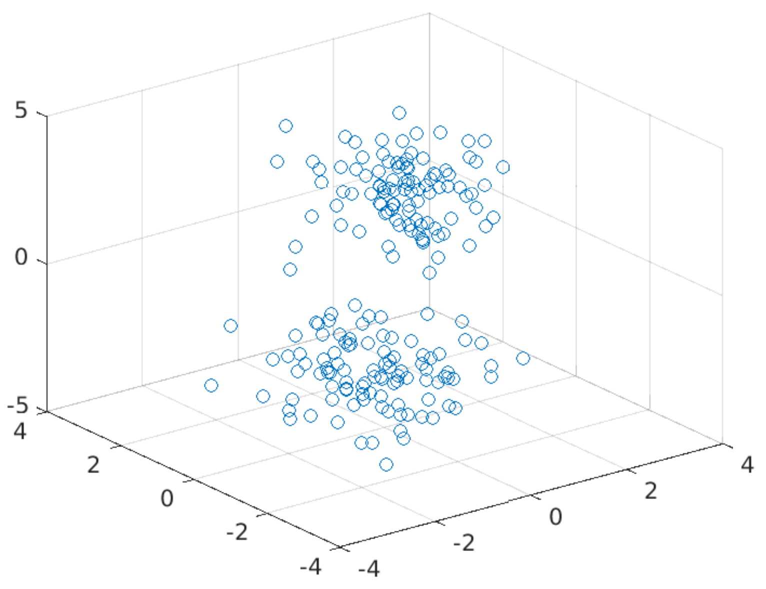

Figure 1: Original data in three dimensions.

2.1 PCA

PCA or Principal Component Analysis, is an application of linear algebra, that finds a basis, with the smallest number of orthogonal elements, that can fully represent the data. The most popular method of finding the principal components, the approach we use here, is using eigendecomposition.

In what follows, we describe the procedure to perform the PCA.

- First, we need to de-mean the data to remove any energy bias during computation. Start by computing the mean of the data. Use the command:

  ```
  >> m = mean(data);
  ```

  This should yield a column vector with three elements, that is, a single point in the same 3-D space. Now, subtract this mean from each element in the set:

  ```
  >> dm = data - repmat(m,r,1);
  ```

- Next, we need to find the covariance matrix. The covariance matrix, defined as:

  $$\mathbf{C}_d = E\left[\mathbf{dm}^T\mathbf{dm}\right], \tag{1}$$

  where **dm** is the demeaned version of the data, **d**. requires infinitely many samples to average over (you will see this in STAT 389, at least). Since we only have 200 samples, we instead approximate the covariance matrix using a scatter matrix (also called empirical covariance matrix) given by:

$$\hat{\mathbf{C}}_d = \frac{1}{r-1}\mathbf{dm}^T\mathbf{dm}, \tag{2}$$

computed as:

```
>> cd=(1/(r-1))*(dm.'*dm);
```

- Eigendecomposition is then performed on the empirical covariance matrix:

  $$\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} = \hat{\mathbf{C}}_d, \tag{3}$$

  where $\mathbf{Q}$ is the matrix of eigenvectors of $\hat{\mathbf{C}}_d$, and $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues of $\hat{\mathbf{C}}_d$. Note that the eigenvalues of $\hat{\mathbf{C}}_d$ are called the singular values of $\mathbf{dm}$. The eigenvectors of $\hat{\mathbf{C}}_d$ are the principal components of $\mathbf{dm}$. Implement the eigendecomposition in MATLAB as follows:

  ```
  [Qd, Dd] = eig(cd);
  ```

2.2 Dimensionality

Note that the matrix $\mathbf{\Lambda}$ is of size 3×3, with three non-zero diagonal elements, λ_1, λ_2, and λ_3, with $\lambda_1 > \lambda_2 > \lambda_3$, without loss of generality. This indicates that the original data lives in three dimensions. Note, however, that the matrix $\hat{\mathbf{C}}_d$ is highly ill-conditioned. A matrix is ill-conditioned when the condition number, $\kappa = \lambda_{\max}/\lambda_{\min} \gg 1$. What is the condition number?

These eigenvalues also represent how well each principal component represents the data. This is sometimes called the variability of the data (applicable when the data is random). In all cases, each eigenvalue provides a measure of the energy of the data projected onto the particular principal component. The energy in the l-th eigenvalue is given by:

$$E_l = \frac{\lambda_l^2}{\sum_k \lambda_k^2}. \tag{4}$$

For the data here, the three eigenvalues case are 7.0727, 0.7590, and 0.5595, representing 98.3%, 1.1%, and 0.6% of the energy. Clearly, most of the data can be represented single principal component. If we do this, we will reduce the data from three dimensions to single dimension. However, in the process, some data will be misrepresented, in this case, less then 0.5%. For most applications, a loss of 10% of the energy or less, is acceptable. If >90% of the energy is contained in n components, the dimensionality of the system can be reduced to n.

Once the number of reduced dimensions is determined, the next step is to represent the data in the lower dimension space. To do this, set the eigenvalues that are not needed for representation, as well as their corresponding eigenvectors to give us $\tilde{\mathbf{\Lambda}}$ and $\tilde{\mathbf{Q}}$, respectively. Reconstruct the reduced dimension data as:

$$\mathbf{d}_r = \tilde{\mathbf{Q}}\mathbf{d}. \tag{5}$$

This can be implemented in MATLAB using the following commands:

- First, sort the eigenvalues in descending order:

```
>> eig_value = diag(Dd);
>> [~,ind] = sort(eig_value);
>> eig_val_sorted = eig_value(ind);
>> eig_vec_sorted = Qd(:,ind);
```

- Next, select the eigenvector corresponding to the highest valued eigenvalue (we pick only the first because that has the highest energy contribution) and reconstruct:

```
>> num_comp = 1;
>> prin_comps = eig_vec_sorted(:,1:num_comp);
>> reconstructed_data = data*prin_comps;
```

- Finally, visualize the reconstructed data:

```
>> figure
>> scatter(reconstructed_data,zeros(size(reconstructed_data)));
```

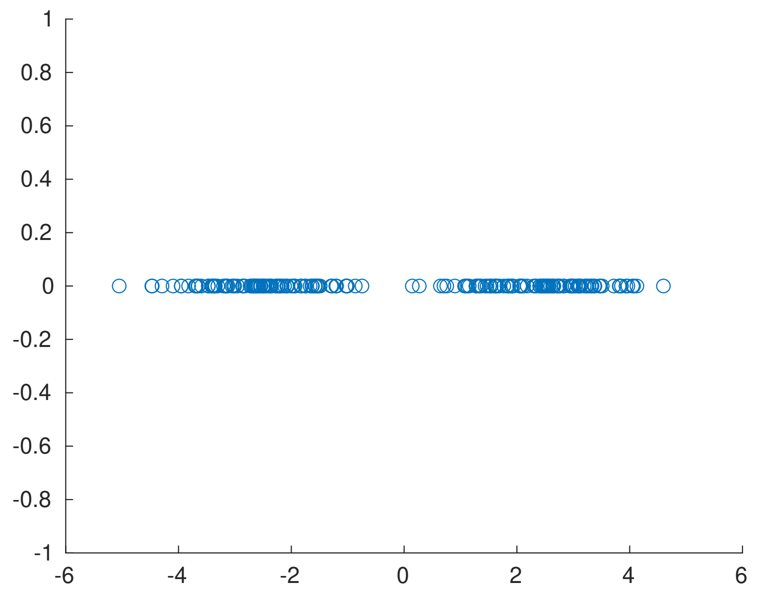

Figure 2: Reconstructed data projected down onto one dimension.

In this lower dimensional representation (see Figure 2), what can you infer about the data?

3 Higher Dimensional Data

Load the new dataset `gen_data2.csv`. For this dataset:

- Find the dimensionality of the data.
- Visualize the data.
- Apply PCA and find the minimum number of dimensions needed to represent >90% of the energy of the data.
- Reconstruct the lower dimensional data and visualize it.
- Interpret the data.

4 Keystroke Data

For this exercise, the CMU keystroke benchmark dataset [1] is used. It consists of static keystroke typing times from 51 subjects, each typing a password "tie5Roanl" 50 times over 8 sessions. Various timing features (e.g., the key press up-down times and hold times) were extracted from the raw data. These keystroke timings are the input features for this dataset. The data is stored in `keystroke_data.csv`. For this dataset:

- Find the dimensionality of the data.
- Apply PCA and find the minimum number of dimensions needed to represent >90% of the energy of the data.
- Reconstruct the lower dimensional data and visualize it.
- Interpret the data.

References

[1] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *2009 IEEE/IFIP International Conference on Dependable Systems Networks*, 2009, pp. 125–134.