TI2736-C Datamining
Assignment 6: Dimensionality Reduction

Delft University of Technology, February–February 2016

Thomas Abeel, Marcel Reinders
Zekeriya Erkin, Julian Kooij
Chantal Olieman, Gijs Weterings, Alex Salazar

Pattern Recognition and Bioinformatics Group

TU Delft

6 Dimensionality Reduction

In the following two exercises you will create an algorithm to perform dimension reduction on datasets. To test the algorithm you will reduce the dimensions of a 2D Gaussian dataset as well as a set of images of faces. The method for calculating the eigenvectors will be the Power Iteration method, which will be implemented first.

Exercise 6.1. Power Iteration

Consider a square $(D \times D)$ matrix M, and that we know that the $D \times 1$ column vector $\mathbf{v}$ is an eigenvector of M, with eigenvalue $\lambda \in \mathbb{R}$.

Question 0.1. What property defines the relation between the M, $\mathbf{v}$, and λ? Write down this equation.

The Power Iteration method is a relatively simple method for calculating eigenvectors for the square $(D \times D)$ matrix M. The process works as follows:

1. Construct a vector $\mathbf{v}$ of ones of length $D \times 1$.
2. Until convergence, compute:
$$\mathbf{v}_{k+1} = \frac{M\mathbf{v}_k}{||M\mathbf{v}_k||} \tag{1}$$
Where $||\mathbf{x}||$ is the (L2) norm of $\mathbf{x}$.
3. Output vector v as the principal eigenvector of M.
4. Compute M^* as:
$$\lambda = \mathbf{v}^\top M\mathbf{v} \tag{2}$$
$$M^* = M - \lambda \times \mathbf{v}\mathbf{v}^\top \tag{3}$$
5. If more eigenvectors are required, go to step 1 with M^* as input.

Step 1. First we will implement the `Matrix` class in `Matrix.java`. Implement the missing code in the methods in `Matrix.java`.

Step 2. Next we will create the `powerIteration` method in `main.java`. This method will calculate `nrVectors` of eigenvectors from the square matrix `m`. Use the steps described above to do so. The stopping criterion in the second step should be when the L2 norm of the difference between $\mathbf{v}_{k+1}$ and $\mathbf{v}_k$ of two consecutive iterations is smaller than the convergence parameter `e`.

Step 3. In the `powerIterationTest` method in `main.java`, create a matrix and let it read the data from `"data/matrix.txt"`. Use the `powerIteration` method to calculate two eigenvectors from this matrix (and set `e` to something like `10E-5`). Check that the resulting eigenvectors are roughly equal to:

$$\mathbf{v}_1 = \begin{bmatrix} 0.4472 \\ 0.8944 \end{bmatrix} \tag{4}$$
$$\mathbf{v}_2 = \begin{bmatrix} 0.8944 \\ -0.4472 \end{bmatrix} \tag{5}$$

Question 3.1. How many eigenvectors could there possibly be in a $D \times D$ matrix?

Question 3.2. Are the eigenvalues of these eigenvectors increasing or decreasing as you compute more vectors? What do these eigenvalues say about the eigenvectors?

Question 3.3. We have seen a very similar method before, what was it used for then? We did not need to normalize the vector then, why is that?

Exercise 6.2. Principal Component Analysis

Next we will use Principal Component Analysis to find the principal components in some dataset. Principal components can be seen as vectors along which most variance is found in the data.

Step 1. Complete the `pca` method in `main.java`. Create a matrix that reads data from `"data/guassian.txt"`. This size of this matrix is $N \times 2$, where N is the number of points in the dataset. Then, use a `PCAPlotter` object to plot the data.

Question 1.1. By just looking at the plotted data, what can you say about the direction of the first principal component? What about the second principal component?

Step 2. The principal components of a dataset can also be seen as the eigenvectors of the covariance matrix. Compute the covariance matrix of the Gaussian dataset as follows:

$$\text{cov}(X) = \frac{1}{N}(X - \bar{\mathbf{x}})^\top (X - \bar{\mathbf{x}}) \tag{6}$$

where $\bar{\mathbf{x}}$ is the mean row of $N \times D$ dimensional data matrix X. Next, compute the eigenvectors of this covariance matrix and also plot the vectors using the same `PCAPlotter` object.

Question 2.1. Which of these eigenvectors captures the most variance? Was this the first principal component or the second?

Step 3. Complete the `pcaFaces` method. This method should create a matrix object and let it read in `"data/faces.txt"`. This matrix is of size $N \times D$, where N is the number of images in the dataset and D is the number of pixels per image (in this case 32×32, so $D = 1024$). Similarly to before, compute the covariance matrix. Use this covariance matrix to compute the first 10 principal components.

Note: Calculating such large matrices is hard for Java, it is perhaps useful to add some output in `powerIteration` to show progress.

Step 4. Visualize the principal components using `ImageFrame` objects.

Question 4.1. What do these principal components mean, in terms of faces?