Mohammed J. Zaki

Dmcourse /
Assign1

Assign 1: Due Date: Tue 13th Sep, 11:59:59PM (i.e., just before midnight on Tue)

Download the Airfoil Data Set dataset from the UCI Machine Learning Repository . The dataset has 6 real valued attributes. We will treat all attributes equally for this assignment, which consists of two parts. Part I must be done by students in both sections, namely CSCI4390 and CSCI6390. For the second part, Part II-4390 must be done by those in CSCI4390, and Part II-CSCI6390 must be done by students registered for CSCI6390.

Part I (both CSCI-4390 and CSCI-6390)

Mean vector and total variance

Compute the mean vector μ for the 6-dimensional data matrix, and then compute the total variance $\text{var}(D)$; **see Eq. (1.4)** for the latter.

Covariance matrix (inner and outer product form)

Compute the sample covariance matrix Σ as **inner products** between the attributes of the centered data matrix (see **Eq. (2.30)** in chapter 2). Next compute the sample covariance matrix as sum of the **outer products** between the centered points (see **Eq. (2.31)**).

Correlation matrix as pair-wise cosines

Compute the correlation matrix for this dataset using the formula for the cosine between centered attribute vectors (see **Eq. (2.25)**). Which attributes are the most correlated, the most anti-correlated, and least correlated? Create the scatter plots for these interesting pairs using **matplotlib** and visually confirm the trends, i.e., describe how each of the three cases results in a particular type of plot.

Part II-4390 (To be coded only by CSCI-4390)

Dominant Eigenvector

Compute the dominant eigenvalue and eigenvector of the covariance matrix Σ via the power-iteration method. One can compute the dominant eigen-vector/-value of the covariance matrix iteratively as follows.

Let

$$x_0 = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

be the starting vector in R^d, where d is the number of dimensions.

In each iteration i, we compute the new vector:

$$x_i = \Sigma\, x_{i-1}$$

We then find the element of x_i that has the maximum absolute value, say at index m. For the next round, to avoid numerical issues with large values, we re-scale x_i by dividing all elements by x_{im}, so that the largest value is always 1 before we begin the next iteration.

To test convergence, you may compute the norm of the dif ference between the scaled vectors from the current iteration and the previous one, and you can stop if this norm falls below some threshold. That is, stop if

$$\|x_i - x_{i-1}\|_2 < \epsilon$$

For the final eigen-vector , make sure to normalize it, so that it has unit length.

Also, the ratio $\frac{x_{im}}{x_{i-1,m}}$ gives you the largest eigenvalue. If you did the scaling as described above, then the denominator will be 1, but the numerator will be the updated value of that element before scaling.

Once you have obtained the dominant eigenvector , u_1, project each of the original data points x_i onto this vector, and print the coordinates for the new points along this "direction".

Part II-6390 (To be coded only by CSCI-6390)

First Two Eigenvectors and Eigenvalues

Compute the first two eigenvectors of the covariance matrix Σ using a generalization of the above iterative method.

Let X_0 be a $d \times 2$ (random) matrix with two non-zero d-dimensional column vectors with unit length. We will iteratively multiply X_0 with Σ on the left.

The first column will not be modified, but the second column will be orthogonalized with respect to the first one by subtracting its projection along the first column (see section 1.3.3 in chapter 1). That is, let a and b denote the first and second column of X_1, where

$$X_1 = \Sigma\, X_0$$

Then we orthogonalize b as follows:

$$b = b - \left(\frac{b^T a}{a^T a}\right) a$$

After this b is guaranteed to be orthogonal to a. This will yield the matrix X_1 with the two column vectors denoting the current estimates for the first and second eigenvectors.

Before the next iteration, normalize each column to be unit length, and repeat the whole process. That is, from X_1 obtain X_2 and so on, until convergence.

To test for convergence, you can look at the distance between X_i and X_{i-1}. If the difference is less than some threshold ϵ then we stop.

Once you have obtained the two eigenvectors: u_1 and u_2, project each of the original data points x_i onto those two vectors, to obtain the new projected points in 2D. Plot these projected points in the two new dimensions.

What to Turn In

Write a script named **assign1-LAST-FIRST.py** that takes as input the data filename, and the epsilon parameter for convergence, where **LAST** and **FIRST** are your last and first names, respectively. You may assume that the data file resides in the local directory where the script will be called from. Save your output to either a text file and name it **assign1-LAST-FIRST.txt**. The output should comprise the mean vector, total variance, covariance matrix via inner and via outer product formulas, correlation matrix, the observations, the dominant eigen-vectors and eigenvalues. The scatter plots can be submitted as PDF or JPG files following the same convention, namely **assign1-LAST-FIRST-figN.jpg/pdf**, where 'N' denotes the fig number, and the extension is eit her '.jpg' or '.pdf'. Your comments on th e plots can be in a separate txt file, or include them in the PDF .

Email your script and output file to datamining.rpi@gmail.com . The subject of the email should be **assign1-LAST-FIRST**.

Your script must use Python 2.7 or Python 3 (please specify which version you are using in the script itself, as a comment on the first line). Please note that you can use built-in NumPy/Python functions for reading and parsing the text input, but you should **NOT** use any of the built-in functions for this assignment. You may however verify your answers by comparing to the result s from the built-in methods like **cov** or **eigen**, and so on.

Tutorial on Python and NumPy

For those not that familiar with python, you may google for tutorials, e.g. Python tutorial. For information on the NumPy module/package use: NumPy tutorial

Policy on Academic Honesty

You are free to discuss how to tackle the assignment, but all coding must be y our own. Please do not cut, copy and/or paste from anyone else, including code on the web. Any students caught violating the academic honesty principle will get an automatic **F** grade on the course and will be referred to the dean of students for disciplinary action.

Page last modified on September 09, 2016, at 01:15 PM