Python exercises for the lecture “Models and Algorithms in Bioinformatics”

Organizers: Peter Meinicke, Sina Garazhian
Submission date: 04.01.2025, 12 p.m.

Sheet 3: K-means

Task 1: synthetic data, initialization

a) Implement the K-means ”hard clustering” algorithm as presented in the lecture. Test the process with four synthetically generated 2D clusters, which are from **Normal** distributions centered on the corners of the unit square where each distribution has a standard deviation of 1/2 in both dimensions. Use a corresponding function (e.g. numpy.random.randn) to generate 100 points randomly for each cluster. Visualize the data as a scatter plot (e.g. with matplotlib.pyplot.plot).

First, initialize $K = 4$ prototype vectors with randomly generated points in the unit square (e.g. with numpy.random.rand). Depict the decrease in the average quadratic error E_i with progressing iteration i as a function plot, the scheme should be terminated after 100 iterations at the latest. Start the procedure again with a different random initialization if a prototype vector receives no assignment. Also introduce a termination criterion that the algorithm stops if $(E_{i-1} - E_i)/E_{i-1} \leq 10^{-4}$. Show the resulting prototypes as additional points in the scatter plot and use a different representation for them (e.g. color, size) to make them stand out from the data points. What would theoretically be the expected value of the squared error E at the use of the ideal prototype vectors and the original assignment according to the generating process? Is this value practically achieved or do you observe a systematic deviation when you compare the theoretical value with the observed error value? Compare the measured errors on 50 random data records with the expected value! What may be the reason for systematic deviations?

b) Try an alternative initialization of the prototypes with K randomly selected data points. Tip: use the function numpy.random.permutation in Python to select these points from the data set. Can you observe an improvement over the previous procedure with regard to the mean error and the number of required iterations over 50 runs with different random initializations? Use the new initialization method in the following.

c) Test the procedure for different numbers of prototypes $K = 2, ..., 10$ and visualize the resulting error (best value from 20 random initializations) depending on K as a function plot. Does the curve give an indication of the actual number of clusters? Now try the whole thing with a standard deviation 1 and 1/4 when creating the data! How do the three function plots according to the different standard deviations differ with regard to the elbow criterion?

d) Implement the robust clustering variant based on minimization of the average city block distance between data points and prototypes. Now add a point $[10, 10]$ to the data record and repeat the test runs including visualization from a) and c) with the initialization from b) for both variants of the error criterion (Euclidean vs. City block). What can be observed? How does the added data point change the resulting prototypes and error function values (in particular for $K = 4$) for the two variants?

Task 2: Gene expression data

Now load (e.g. with numpy.genfromtxt) the data matrix of the GexprData.csv file in the `Data` directory. Note that the 384 data points appear as rows in the matrix. What do the data points represent (see reference)?.
Cluster the data and vary the number of prototypes again according to $K = 2, ..., 10$. For each K depict the minimum error over 20 random initializations in a function plot. Can you "guess" the number of existing clusters? Repeat the experiment with the robust K-means variant - do you observe any difference?

Reference:
K. Y. Yeung, C. Fraley, A. Murua, A. E. Raftery, W. L. Ruzzo, Model-based clustering and data transformations for gene expression data , Bioinformatics, Volume 17, Issue 10, October 2001, Pages 977-987 https://doi.org/10.1093/bioinformatics/17.10.977

Submission instructions

Please submit your solution by **4th January** using the corresponding homework folder in StudIP. Upload a **zip** compressed file containing all documents and data that you use in your solution. The file name should indicate the author, for example *Peter_Meinicke_Sheet3.zip*.
With your solution you should provide the Python source code together with the documentation within a **Jupyter Notebook** answering all the questions above. Use Markdown cells for your answers and explanations well separated from the Python code.
For the programming in Python/numpy do not use any code or functions from special bioinformatics libraries or toolboxes! Make sure that your code is readable and understandable for others, i.e. use descriptive names for the variable definitions according to the conventions on **Sheet 0**, do not use implicit variables (hard-coded values without a variable) and make use of comments!