Project description

In the project, you will make use of what you have learned about programming and data science on the course. This part also requires learning on your part, this is not only applying what you already know.

In the project, you will mainly implement a k-means clustering algorithm according to the specification in this project description and then apply your own implementation on clustering data sets. You will report the findings, and make some experiments with different choices of parameters in the clustering algorithm.

The project will be graded. The tentative grading table is in the end of this project document.

Background on k-means clustering algorithm

K-means algorithm is a classical algorithm for clustering numerical, multivariate data. The data vectors that are clustered have d dimensions and there are N vectors in the data set. The data matrix has thus the size N x d (N rows, d columnns). The k-means algorithm represents the data with k so called prototypes, which are similarly represented to the data vectors (d dimensions). The purpose of the prototypes is to model individual data vectors with a local average of the data vectors. You can represent the set of k prototypes (cluster centers) in a matrix that collects the k prototypes together as a k x d matrix (k rows, d dimensions). The parameter k is specified by the user. You intialize the prototype matrix by k random entries from the data. If the data has numeric, continuous values, you should normalize the data to have zero mean and unit variance, in order to variables have the same magnitude.

After initialization, the algorithm itself consists of two steps.

In the first step, you allocate each of the data vectors to (exactly) one prototype according to the distance between the data vector and all the prototypes. The allocation of a data vector is done to the prototype closest to the data vector, in other words, to the prototype that has the minimum distance to the data vector. This creates a partitioning of the data vectors to k sets of data.

In the second step, new prototype vectors are calculated. The prototype vector is estimated from the data vectors associated with it by calculating the average of the data vectors belonging to that cluster. This will change slightly which data vectors are allocated to the cluster in the next steps of the algorithm.

The first and second steps are repeated (or iterated) until there are no changes to the allocation of data vectors to the prototypes. Rephrasing, the prototypes will stay the same as in the previous iteration. The results of the k-means algorithm are the prototypes (cluster centers) and the allocation of data vectors to clusters.

You will use two kinds of distance functions: Euclidean distance function for continuous, numeric attributes and Jaccard distance for data, which consists of values 0's and 1's only.

Mathematically, the Euclidean distance between a data vector $\boldsymbol{x}$ and the prototype vector $\boldsymbol{m}$ is expressed as follows:

$$d_E(\boldsymbol{x}, \boldsymbol{m}) = \sqrt{\sum_{i=1}^{d} (x_i - m_i)^2}$$

Alternatively, if the data only has binary (or Boolean) entries (0's and 1's), one can use the Jaccard distance instead. The Jaccard distance is expressed as follows:

$$d_J(\boldsymbol{x}, \boldsymbol{m}) = 1 - \frac{|\, \boldsymbol{x} \cap \boldsymbol{m} \,|}{|\, \boldsymbol{x} \cup \boldsymbol{m} \,|}$$

where the $\cap$ stands for the intersection of the binary values (elementwise), and the $\cup$ stands for the union of the binary values (elementwise), and the $|\cdot|$ stands for the number of ones in this case. For instance, if the data vector is $\boldsymbol{x} = (\mathbf{0}, \mathbf{0}, \mathbf{1})$ and the prototype vector is $\boldsymbol{m} = (\mathbf{1}, \mathbf{0}, \mathbf{1})$, then $\boldsymbol{x} \cap \boldsymbol{m} = (\mathbf{0}, \mathbf{0}, \mathbf{1})$ and $\boldsymbol{x} \cup \boldsymbol{m} = (\mathbf{1}, \mathbf{0}, \mathbf{1})$ and $d_J(\boldsymbol{x}, \boldsymbol{m}) = 1 - \frac{1}{2} = \frac{1}{2}$.

Parts of the implementation

For all functions, documentation strings are required listing the author, one sentence description of the purpose of the function and the description of the function (input) arguments and the return values. Each file (if you have many) must also have the author information and one sentence description in the beginning.

Task 1: Implement the k-means clustering algorithm, as described above. Implement the Euclidean distance for computing the similarity between the data and the cluster prototypes, as well as Jaccard distance for 0-1 data. The distance function used is selected in and returned by a dispatch function, which will either return the Euclidean distance function or the Jaccard distance function. The terminating condition for the k-means algorithm is that the cluster assignment of the points to clusters remains the same between the iterations. Grading criteria: correctness, readability of the code, documentation, adherence to specifications (60 points)

Task 2: Familiarize yourself with the model selection criterion silhouette coefficient and its readily available implementation in Scikit-learn library. You can find the documentation of the function sklearn.metrics.silhouette_score on the library web page at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html# and its user page at: https://scikit-learn.org/stable/modules/clustering.html#silhouette-coefficient The value of the Silhouette coefficient is larger for better clustering solutions and can be used for selecting how many clusters is a good choice for your model. (task not graded)

Task 3: Analyze the data set measurements.dat by trying out different clustering models and selecting the most appropriate number of clusters according to a model selection criterion. Try out different numbers of clusters ranging from 2 to 20 and calculate silhouette coefficients for each solution. Plot the silhouette coefficient values against the number of clusters and make an argument for your choice of the number of clusters. Grading criteria: correctness, readability of your code, documentation, adherence to specifications (20 points)

Task 4: Analyze the data set dna_amp_chr_17.dat by trying out different clustering models and selecting the most appropriate number of clusters according to a model selection criterion. Try out different numbers of clusters ranging from 2 to 20 and use the pre-computed distances between the data vectors to provide the distances (see links in Task 2). Plot the silhouette coefficient values against the number of clusters and make an argument for your choice of clusters. Grading criteria: correctness, readability of your code, documentation, adherence to specifications (20 points)