Clusterization by the K-means method when K is unknown

Natalya Litvinenko, *Orken* Mamyrbayev, *Assem* Shayakhmetova*, and *Mussa* Turdalyuly

Institute of information and computational technologies of CSMES RK, Kazakhstan

Abstract. There are various methods of objects' clusterization used in different areas of machine learning. Among the vast amount of clusterization methods, the K-means method is one of the most popular. Such a method has as pros as cons. Speaking about the advantages of this method, we can mention the rather high speed of objects clusterization. The main disadvantage is a necessity to know the number of clusters before the experiment. This paper describes the new way and the new method of clusterization, based on the K-means method. The method we suggest is also quite fast in terms of processing speed, however, it does not require the user to know in advance the exact number of clusters to be processed. The user only has to define the range within which the number of clusters is located. Besides, using suggested method there is a possibility to limit the radius of clusters, which would allow finding objects that express the criteria of one cluster in the most distinctive and accurate way, and it would also allow limiting the number of objects in each cluster within the certain range.

1 Introduction

Nowadays artificial intelligence is a very popular tool in various fields of science - economics, public life and production. Since the beginning of the 21st century, Bayesian networks have been the most common direction of using artificial intelligence. Bayesian networks are widely used in such areas as economics, psychology, sociology, medicine, genetics, management theory, etc.

The theory of Bayesian networks is provided in [1,2,3]. In [2,4] you can see the graphical aspects of the construction of Bayesian networks.

In the process of modelling with the use of Bayesian networks, almost every researcher faces typical problems, usually associated with a lack of specific knowledge and experience in the studied area. We have listed below some of these problems:

• The user does not have the opportunity to assess correctly the probabilistic characteristics of the variables of some nodes of the Bayesian network;

• The user does not have the opportunity to assess correctly the probabilistic dependencies between individual nodes of the Bayesian network;

• It can be difficult to define the presence of the probabilistic dependencies between some nodes of the Bayesian network;

• Sometimes researcher may have some doubts about the necessity of determining Bayesian networks' separate nodes individual nodes of a Bayesian network.

Usually, these problems are solved by involving specialists in the studied area. But sometimes the effectiveness of such involvement is not clear. It is not always clear how the subjective knowledge of specialists reflects the reality.

In the research process, we want to see a special mechanism that allows either to control the decisions of specialists in the research area or even to refuse the involvement of such specialists.

Today researcher can find with ease required experimental data for almost every studied problem. We want to construct the mechanism, that allows using of big amount of data for corrections or partial construction of Bayesian networks. We want to implement the mechanism in an open program environment with the user-friendly interface.

Different computations in Bayesian networks are quite a challenging task, which requires the modern and powerful computer technologies. The processing of big amount of data in the construction of Bayesian networks significantly complicates an already difficult task. This implies new problems not only in the computational area but in mathematics. Therefore, our main goal is to implement the solve of different typical problems in an open program environment with the user-friendly interface.

The modern software market quickly responded to these problems. There are quite a lot of good software products, both for calculations in Bayesian networks and for construction of the Bayesian networks. We have listed below the most popular software products:

• **BayesiaLab**. Bayesia SAS - French software development company, founded in 2001, that specialized in artificial intelligence technologies. The BayesiaLab focuses on various aspects of decision-making process' support with Bayesian networks. The software allows you to easily construct Bayesian networks and conduct

various computations in already existing networks. For acquaintance with the basic principles of work, you can refer to [5].

AgenaRisk. Agena is a supplier of advanced software for solving problems related to risk analysis and decision-making support. In its research, the company focuses on the use of large-scale Bayesian networks. As BayesiaLab this software allows you to easily construct Bayesian networks and conduct various computations in already existing networks. For acquaintance with the basic principles of work with AgenaRisk, you can refer to [6,7].

- **Bayes Server**. The commercial British company, specialized in the development of intelligent systems based on artificial intelligence and machine learning. The main development – **BayesServer.** The first version was released in 2008. As of the March 2018 current version is 7.25.0.0. This software is commercial and fees are high enough. There are 2 trial versions – in the first one version users have significant restrictions on the networks' size and cannot save their projects, in the second one - user's work session is limited to 120 minutes.
- **Netica**. **Norsys Software Corp**. is the commercial company based in Vancouver, Canada. **Norsys** specializes in the software development for Bayesian networks. Price for the commercial edition in 2018 is 685 USD and 285 USD for the academic license. A free version is also available but has restrictions to the model size.
- **Hugin Expert**. **HUGIN EXPERT** is the leading developer and supplier of software for decision-making support in the field of artificial intelligence. The company was founded in 1989 by the world's leading researchers in the field of graphic models based on Bayesian networks.
- **BayesFusion**. BayesFusion, **LLC** was founded in 2015, Pittsburgh, USA. The company supplies software for solutions' modelling based on the principles of decision-making theory. The main software product is GeNIe Modeler, a tool for modelling and learning with the use of Bayesian networks. **GeNIe** is compatible with Win/Mac/Linux.
- **Matlab toolboxes**. Among the Matlab toolboxes oriented to work with Bayesian networks, there are two main packages: BNT (Bayes Net Toolbox) and BRML. **BNT toolbox** was developed by Kevin Murphy. The latest changes were made in October 2007. **BRNL toolbox** was developed by David Barber.
- **R toolboxes**. Among the R toolboxes oriented to work with Bayesian networks the most popular one is **gRain**. **gRain** was developed by Søren Højsgaard. The latest changes were made in October 2007.
- **C# libraries**. **Infer.NET** - a library for probabilistic calculations in graphics models that uses and provides modern algorithms for calculations in graphics models, as well as the accompanying algorithms and subprograms of linear algebra which are necessary for modern machine learning applications. As of the beginning of 2018, the current available version is 2.7 Beta (March 2018) with a non-commercial license. The group of developers - Tom Minka, John Winn, John Guiver, Yordan Zaykov.

The process of construction of Bayesian networks is usually called the Bayesian networks' learning. For acquaintance with the Bayesian networks' learning methods, you can refer to [1,8].

Probabilistic aspects of machine learning and the basis of various algorithms used in machine learning are described in [9]. We have described two types of the Bayesian networks' learning below:

Probabilistic aspects of machine learning and the basis of various algorithms used in machine learning are described in [9]. We have described two types of the Bayesian networks' learning below:

- **Controlled learning**. Under Bayesian networks' controlled learning we usually understood as different ways to determine the probabilistic characteristics of the separate nodes' variables of the network, as well as the probabilistic dependencies between separate nodes based on some array of experimental data.
- **Uncontrolled learning**. Under Bayesian networks' uncontrolled learning we usually understood methods of defining new nodes of the network and new dependencies between nodes based on some array of experimental data.

In the process of uncontrolled learning, the most common way of work is the use of clusterization algorithms. The task of clusterization is to combine different objects into groups (clusters) according to various characteristics. Objects with similar characteristics refer to the same cluster. The distance between objects is often determined by a certain metric that reflects the basic properties of the studied process. The rules by which objects belong to the same cluster can be different. The K-means method is one of the popular methods of clusterization. As well as the other methods of clusterization, the K-means method has its advantages and disadvantages. We have listed the main disadvantages below:

- We must know in advance the number of clusters that the set of objects will be divided to. In practice, we often have only a restriction on the number $K: K_{min} \leq K \leq K_{max}$.
- We must to select appropriate clusters' centers in the first approximation.
- We may have restrictions to clusters' radiuses.
- We may have restrictions to the maximum and minimum number of objects in a cluster.

The main advantage of this method is a rather fast clusterization.

Our goal in this work is to modify K-means method in such a way that the user could set additional requirements for the clusterization method and, at the same time, the time of clusterization should not increase significantly. We will develop the clusterization algorithm and implement it in program code. We assume to apply this software to grant project «Development and software implementation of a package for solving applied problems in Bayesian networks».

2 Problem statement

Suppose we have some set of experimental data. We will assume that the given set is stored in the form of a table of some database. When we were implementing this algorithm in the code, we used the ACCESS database.

Each researcher usually has at his disposal some data sets, which they can use to clarify the details of the Bayesian network that reflects the studied process. These data sets not always reflect correctly the studied process. It is possible that only 40-60% of the data in the set are useful. Therefore, we will have a question – how to find this useful part? Clusterization usually processes all the data array and sort all the objects into different clusters. The user, however, needs to extract from the data array only a part of the data most adequately reflecting the studied process. Users try to solve this problem by different methods. For example, in the process of clusterization they try to limit the cluster size to some small diameter, to exclude unloaded clusters, to limit the number of objects in clusters, etc. In other words, the user solves the problem of separating some criteria from the data array instead of solving the clusterization task. At the same time, he is ready to exclude from consideration most of the array's data.

However, it is not always possible to select a appropriate software that can perform clusterization with the additional requirements of the researcher (searching for criteria).

In this paper, we replaced the problem of separating a set of objects into clusters to the task of extracting criteria in the considered set of objects. By the criteria, we will further understand some part of the cluster that most clearly defines the property of the cluster. We expect that the criteria will more correctly reflect some categories of the considered object, that were not clear in the beginning. It would be wrong to select criteria after the stage of separating the array into clusters. For example, a cluster may do not have any criteria, or, conversely, a cluster may have several criteria each with its own center. It would be right to construct criteria instead of clusters' construction. Further, in the paper, we will not define the concept of criteria, we will use the cluster notion. But in the process of clusters' construction, we will introduce additional restrictions, for example, on the radius of the cluster, with the expectation that some of the constructed clusters will coincide enough with the criteria. The concept of criteria is a rather complex concept, and we will not consider it in this paper.

We need to separate the considered data set into clusters by the K-means method with additional significant restrictions.

In this paper the diameter D of the array means the distance in the selected metric d between the most distant objects. By the cluster diameter Dk we mean the distance between the most distant cluster objects, expressed as a number from 0 to 1 and as the ratio of the cluster's diameter to the array's diameter.

In this paper the radius R of the array means the distance in the selected metric d between the center of the array and the most distant object. By the cluster radius Rk we mean the distance between the cluster center and the most distant cluster object, expressed as a number from 0 to 1 and as the ratio of the cluster's radius to the array's radius.

Note that the definition of the array's radius here differs from the generally accepted definition. It is clear that R < D < 2*R.

2.1 General limitations

- The maximum number of records is limited to 1,000,000 (1 million records).
- The maximum number of fields in a record is limited to 100
- In our paper, we only consider the construction of clusters, therefore, we omit all issues, related with the choice of the metric (although this is a rather difficult question). In this paper we consider only the Euclidean metric:

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2 + \cdots + (x_n - y_n)^2}$$

- The maximum number of clusters is limited to 10000. In this paper, we will limit the number to 100 clusters.

2.2 Limitations in the process of clusters' construction

- User can set the lower limit of the number of constructed clusters ($KlaMin$). In the process of dividing the data array into clusters the number of clusters should be not less than $KlaMin$.
- User can set the upper limit of the number of constructed clusters ($KlaMax$). In the process of dividing the data array into clusters the number of clusters should be less than $KlaMax$.
- User can set the lower limit of the number of objects ($ObMin$) in each constructed cluster.
- User can set the upper limit of the number of objects ($ObMax$) in each constructed cluster
- User can set the maximum diameter ($Dmax$) of a cluster.
- User can set the maximum radius ($Rmax$) of a cluster.

3 Problem

We need to split the data array into clusters so that the number of clusters n_{Kla} satisfies the condition $n_{Kla} \in (KlaMIn, KlaMax)$, the number of objects n_{Ob} in each cluster satisfies the condition $n_{Ob} \in (ObMin, ObMax)$, the diameter of each cluster was limited by the value of $Dmax$, and the radius of each cluster was limited the value of $Rmax$. The conditions are set by the operator.

4 Stages of clusterization algorithm

The first stage – We define natural clusters in an objects' array. Then we exclude clusters with the number of objects less than $ObMin$**.**

The task of the first stage is to determine the natural clusters of the considered objects. In the next stage of clusterization the centers of the most loaded clusters will be taken as the first approximation. If we obtain too many such clusters (more than $KlaMax$), we take only $KlaMax$ clusters with the largest number of objects. If we obtain very few such clusters (less than $KlaMin$), we take only $KlaMin$ clusters with the largest number of objects (here we can also obtain empty clusters).

1) We significantly increase the maximum number of clusters. In this algorithm, the maximum number of clusters is 3 times increased (specified in the program settings).

Let denote $K1 = KlaMax * 3$. And let $K = K1$.

2) We apply normal clusterization by K-means method for the calculated $K = K1$. Here we do not have additional limitations, listed above. As initial cluster centers, we select randomly constructed centers of $K1$ clusters.

3) We analyze the obtained clusters. If there are some empty clusters (no one object contained) we count them. Let $K2$ – the number of empty clusters. We define $K = Max(KlaMin, K - Max(1, K2 * 0.5))$.

4) We repeat steps 2) and 3) until either the empty clusters disappear, or K becomes equal to $KlaMin$. For the initial coordinates of the clusters, we take the coordinates of the clusters calculated in the previous iteration. Thus, we reduce the number of clusters.

5) After the empty clusters are over, we will do the same operations with 'bad' clusters, i.e. with those that contain no more than $Nmin$ objects.

6) We count the number of 'bad' clusters. Let $K2$ – the number of 'bad' clusters. We define $K = Max(KlaMin, K - Max(1, K2 * 0.5))$.

7) We perform normal clusterization by K-means method for the calculated K. Here we don't have additional limitations which we listed above. For the initial coordinates of the clusters, we take the coordinates of the clusters calculated in the previous iteration.

8) We repeat steps 6) and 7) until either the empty clusters disappear, or K becomes equal to $KlaMin$.

9) We limit the radiuses and diameters of clusters with the values $Rmax$ and $Dmax$. And then we find the centers of new clusters.

The second stage – We determine the optimal number of clusters, considering the acceptable cluster radius $Rmax$, the acceptable cluster diameter $Dmax$, the acceptable number of objects in the cluster (less than $ObMax$), the acceptable number of clusters (more than $KlaMin$ and less than $KlaMax$).

In the process of defining clusters in the second stage, we require that the number of objects in each cluster is not more than $ObMax$. At a time, we will try to keep objects closest to the center of the cluster. For that we will slightly complicate the clusterization algorithm. At each iteration, an array of objects will be considered several times, starting with the closest objects to the cluster centers, and finishing with the most distant objects. We divide the maximum permissible cluster radius ($Rmax$) into p parts:

$EpsKL = Rmax/p$, where the number p is specified by the operator. We divide a maximally acceptable cluster radius ($Rmax$**)** into p parts: $EpsKL = Rmax/p$, where p is operator.

At the second stage, it is possible to consider the data array up to p times at each iteration, $EpsX = EpsKL * jj$. Here $jj \in (1, p)$. Thus, at first, we collect the objects closest to the center of the cluster into clusters. We can obtain the situation that for some $jj = jA$, no object belongs to any cluster. Then we stop the iteration. Some objects, maybe most of them, will not belong to any cluster.

Now we will describe the algorithm above more detailed. We define this algorithm as Alg.

Alg:

To use additional clusterization conditions, we modify the standard clusterization method.

For the initial centers of clusters, we take the centers K of the clusters obtained in the first stage. Now we will describe the clusterization algorithm:

1) Let N – number of objects in the array, L – object's dimension, K – current number of clusters, obtained in the first stage. We define following arrays and variables:

- $NK1(N)$ - to store the number of cluster which contains the object in the next iteration. At the beginning of each iteration, the array is filled with -1, which means that the object does not belong to any cluster.
- $NK2(K)$ – to store the number of objects in each cluster after the next iteration. At the beginning of each iteration, the array is filled with 0.
- $KOOR1(K, L)$ – coordinates of the cluster's centers calculated at the previous iteration. At the beginning of calculations, the array is filled with coordinates, obtained in the first stage.
- $KOOR2(K, L)$ – coordinates of the cluster's centers calculated at the end of the current iteration. At the beginning of calculations, the array is filled with 0.
- xNK – to store the number of nearest cluster.
- xRK – to store the distance to the nearest cluster.

2) At the beginning of the next iteration, we assume $NK1 = -1, NK2 = 0$.

3) At the beginning of the next consideration (jj – consideration), we calculate $EpsX = EpsK1 * jj$**.**

4) We choose the next object ja from the array, for which $NK1(ja) = -1$ (This object still not belongs to any cluster). We assume $xNK = -1, xRK = EpsX$.

5) We choose the next jb, not filled maximally cluster $(NK2(jb) < ObMax)$. We calculate the distance from the object to this cluster. If this distance less than xRK, we store this distance in xRK, and in xNK we store the current cluster's number.

6) We repeat the step 5, until we have considered all the clusters.

7) If $xNK = -1$, then the object is not belonging to any cluster. Otherwise

8) $NK1(ja) = jb, NK2(jb) = NK2(jb) + 1$.

9) If we still have objects, then we repeat step 4).
10) If we don't have objects anymore and at least one object belongs to any cluster and $jj < p$, then we do step 3).
11) We store the current coordinates of clusters' centers $KOOR1 = KOOR2$. In $KOOR2$ we calculate the coordinates of new clusters' centers. We compare new coordinates with the old ones. If the difference is big, we do step 2).
12) We have finished. It is the end of the block Alg.

Unfortunately, in this clusterization we may obtain 'bad' clusters (clusters, where the number of objects is less than $ObMin$) again. If we have obtained the 'bad' clusters, we remove half of them, but so that the number of 'good' clusters is bigger than $KlaMin$, and then we start again the block Alg for the new K.

We repeat these steps until there are no 'bad' clusters or until we have $K = KlaMin$.

In the first case, if we do not have 'bad' clusters anymore then we have the process of clusterization finished.

In the second case, if $K = KlaMin$, and we cannot reduce the number of clusters, we try to correct the 'bad' clusters by adding the nearest $ObMin$ objects in each 'bad' cluster at the beginning of each iteration. It is the third stage – the last clusterization of objects.

The third stage – We check the number of objects in clusters (bigger than $ObMin$ and less than $ObMax$).

In the process of cluster formation in the third stage we will already know the number of clusters. We only need to ensure the number of objects in each class is not less than $ObMin$. Also, we want to keep objects, which are nearest to the clusters' centers.

Now we will describe the clusterization's algorithm Alg2:

Alg2:

1) At the beginning of the next iteration, we assume $NK1 = -1, NK2 = 0$.
2) In each cluster we add $ObMin$ nearest objects:

a. At the beginning of the next consideration (jj – consideration) we calculate $EpsX = EpsK1 * jj$.

b. We choose the next object ja from the array, for which $NK1(ja) = -1$ (This object still not belongs to any cluster). We assume $xNK = -1, xRK = EpsX$.

c.We choose the next jb, not filled minimally cluster $(NK2(jb) < ObMin)$. We calculate the distance from the object to this cluster. If this distance less than xRK, we store this distance in xRK, and in xNK we store the current cluster's number.

We repeat the step c, until we have considered all the clusters.

d. If $xNK = -1$, then the object is not belonging to any cluster. Otherwise $NK1(ja) = jb$, $NK2(jb) = NK2(jb) + 1$.

e. If we still have objects and we have clusters, where the number of clusters is less than $ObMin$, then we repeat step b.

f. If we don't have objects anymore and at least one object belongs to any cluster and $jj < p$, then we do step a.

3) In each cluster we add other objects, but their amount is less than $ObMax$. Then again, we organize consideration of array P times.

a. At the beginning of the next consideration (jj – consideration) we calculate $EpsX = EpsK1 * jj$.

b. We choose the next object ja from the array, for which $NK1(ja) = -1$ (This object still not belongs to any cluster). We assume $xNK = -1, xRK = EpsX$.

c. We choose the next jb, not filled minimally cluster $(NK2(jb) < ObMax)$. We calculate the distance from the object to this cluster. If this distance less than xRK, we store this distance in xRK, and in xNK we store the current cluster's number.

d. We repeat the step c, until we have considered all the clusters.

e. If $xNK = -1$, then the object is not belonging to any cluster. Otherwise $NK1(ja) = jb, NK2(jb) = NK2(jb) + 1$.

f. If we still have objects and we have clusters, where the number of clusters is less than $ObMax$, then we repeat step b.

g. If we don't have objects anymore and at least one object belongs to any cluster and $jj < p$, then we do step a.

4) We store the current coordinates of clusters' centers $KOOR1 = KOOR2$. In $KOOR2$ we calculate the coordinates of new clusters' centers. We compare new coordinates with the old ones. If the difference is big, we do step 1).
5) We have finished. It is the end of the block Alg2.

3 Conclusions

In this paper we have discussed the modified K-means method of clusterization, which allows user, instead of accurate definition clusters' number, to define the range within which the amount of cluster should be located. The other new possibility is the limitation of the objects' number in each cluster, as well as limitation of cluster's radius, which would help user to find only the objects that characterize the cluster most. Using this method user can independently choose the range of number of clusters, number of objects in each cluster, and limitations for cluster radius. In this paper we also described the algorithm of this clusterization method. The Algorithm is implemented on the C# in the Visual Studio 2013.

Software

Our algorithms are implemented in program code on C# in Visual Studio 13. At the current moment, we have draft program version in an interactive module and our team is working on debugging. We are planning to apply this module in our grant project.

Acknowledgment

This work was supported by the «Institute of information and computational technologies» of Committee of science of the Ministry of Education and Science of Republic Kazakhstan, (ITR number is №AP05131293). Additionally, we would like to express our enormous gratitude to Prof. Maksat Kalimoldayev, Academic member of the National Academy of Sciences, the head of Institute of Information and Computational Technologies for his organizational assistance, valuable comments and financial support.

References

1. D. Barber, Bayesian Reasoning and Machine Learning. 686 (2017). URL: http://web4.cs.ucl.ac.uk/staff/D.Barber/textbook/020217.pdf
2. D. Barber, Machine Learning. A probabilistic Approach. 343 URL: https://pdfs.semanticscholar.org/7bc7/54bc548f32b9ac53df67e3171e8e4df66d15.pdf
3. M. Goncharov, Bayesian networks. 46 (2011) URL: http://www.businessdataanalytics.ru/download/BayesianNetworks.pdf
4. F.V. Jensen, T.D. Nielsen, Bayesian Networks and Decision Graphs. Springer. 447 (2007)
5. S. Conrady, L. Jouffe, Bayesian Networks & BayesiaLab. A Practical Introduction for Researches. ISBN: 978-0-9965333-0-0
6. AgenaRisk 7.0 User Manual. 204 (2016) URL: http://www.agenarisk.com
7. Getting Started with AgenaRisk. 31 (2013). URL: http://www.agenarisk.com
8. R.E. Neapolitan, Learning Bayesian Networks. 704. URL: http://www.cs.technion.ac.il/~dang/books/Learning%20Bayesian%20Networks(Neapolitan,%20Richard).pdf
9. K.P. Murphy, Machine Learning A Probabilistic Perspective. The MIT Press. 1067.