Marcin PIETRZYKOWSKI, Marcin PLUCIŃSKI

Faculty of Computer Science and Information Technology, West Pomeranian University of Technology

doi:10.15199/48.2017.01.18

Mini-model method based on *k*-means clustering

Abstract. *Mini-model method (MM-method) is an instance-based learning algorithm similarly as the k-nearest neighbor method, GRNN network or RBF network but its idea is different. MM operates only on data from the local neighborhood of a query. The paper presents new version of the MM-method which is based on k-means clustering algorithm. The domain of the model is calculated using k-means algorithm. Clustering method makes the learning procedure simpler.*

Streszczenie. *Metoda mini-modeli (metoda MM) jest algorytmem bazującym na próbkach podobnie jak metoda k-najbliższych sąsiadów, sieć RBF czy sieć GRNN ale jej zasada działania jest inna. MM operuje tylko na danych z najbliższego otoczenia punktu zapytania. Artykuł prezentuje nową wersję metody MM, która bazuje na algorytmie k-średnich. Domena MM jest obliczana przy pomocy algorytmu k-średnich. Użycie algorytmu klasteryzacji uprościło procedurę uczenia.* ***(Metoda mini-modeli bazująca na algorytmie k-średnich)***

Keywords: mini-model, local self-learning, function approximation, instance-based learning, *k*-means
Słowa kluczowe: mini-modele, lokalne samouczenie, aproksymacja funkcji, algorytm bazujący na próbkach, algorytm *k*-średnich

Introduction

The paper presents new version of the mini-models method based on *k*-means clustering algorithm. The Mini-model method (MM-method) already was described in many publications. General idea was developed by Piegat [1] and depends on assumption that in the modeling task very often we are only interested in an answer to a specific query. Approach presented by global methods such as neural networks, neuro-fuzzy networks, polynomial approximation and other can be successfully replaced by a local constrained mini-model. MM operates only on the data from the local neighborhood of a query. Thus, learning process of MM-method is in fact the identification process of a mathematical function, which describes the dependence between input and output variables only in some part of the space. The local neighborhood of a query point is called a *mini-model area* or a *mini-model domain* and is defined in the input space of the problem. It includes the set of points which is used in the learning process. In the general case for *n*-dimensional space, the mini-model area has character of a *n-1*-dimensional convex geometric object. The previous publications describe MM based on: simplex, *n*-cube,*n*-orthoplex [2, 3], hyper-sphere or hyper-ellipsoid [4, 5, 6].

The paper presents MM-method modification in which the domain of the model is calculated using the *k*-means clustering algorithm. Clustering algorithm makes the MM-method simpler and gives lower computational complexity. It calculates local neighborhoods for all the query points at once at the beginning of the learning process. It gives an edge in the situation when we have to calculate answers for several query points. This approach frees us from creating MM-domains for different query points separately.

The MM-method belongs to the group of a supervised instance-based learning algorithms (memory-based learning) [7]. This kind of methods compares new problem instances with instances knowing from training, which are stored in the memory. Main drawback of this type of algorithms is dependency between complexity and data growth, but instance-based learning methods have great ability to adapt the model to new data. Examples of instance-based learning algorithm are the *k*-nearest neighbor algorithm [8], GRNN network [9] or RBF network [10]. The *k*-NN method can be considered as a special case of a mini-model method [2].

Mini-model domain

The Mini-models algorithms can be divided into two groups: algorithms for defining the local neighborhood of the query point and algorithms for mathematical modeling on the mini-model domain. Defining local neighborhood will be based on clustering algorithms, whilst for a mathematical modeling linear regression will be used.

MM-domain includes the set of points which is used in the learning process. In previously described versions of the method, finding a mini-model domain was a complex process. The MM has to find optimal domain among all available domains which fulfills initial criteria: domain has to be a convex set, number of samples contained inside the domain has to be within a specified range, etc. It gives several advantages: MM-domain was precisely described (it has hyper-spherical shape or a shape of a convex polytope), and the accuracy of the method was high. Authors propose that mini-models domain will be created by the division of the input space at once at the beginning of the calculations. The process will be common for all possible mini-models that can be calculated for a particular data set. It gives an edge in the situation when we have to calculate answer for several query points.

K-means algorithm [12, 11] is used for input space division into several subspaces. It is iterative, data-partitioning algorithm that assigns *n* observations to exactly one of *k* clusters. *k* is chosen *a priori* before the algorithm starts. Each cluster is defined by centroid. In the basic version centroid is computed as a mean of all data points belonging to the cluster. The algorithm proceeds as follows:

1. Choose *k* initial cluster centers (centroid).
2. Compute point-to-cluster-centroid distances of all observations to each centroid.
3. Assign each observation to the cluster with the closest centroid.
4. Compute the average of the observations in each cluster to obtain k new centroid locations.
5. Repeat steps 2 through 4 until cluster assignments do not change, or the maximum number of iterations is reached.

In order to improves the running time of the algorithm, and the quality of the final solution we can find intial centroid seeds by *k*-means++ algorithm [13]:

1. Select an observation uniformly at random from the data set. The chosen observation is the first centroid, and is denoted c_1.
2. Compute distances from each observation to c_1. Denote the distance between c_i and the observation m as $d(x_m, c_i)$.
3. Select the next centroid, c_2 at random from data set with

probability

$$\frac{d^2(x_m, c_1)}{\sum_{i=1}^{n} d^2(x_i, c_1)}$$

4. To choose center i:
   - Compute the distances from each observation to each centroid, and assign each observation to its closest centroid.
   - For $m = 1, \ldots, n$ and $p = 1, \ldots, i-1$, select centroid i at random from data set with probability

$$\frac{d^2(x_m, c_p)}{\sum_{h; x_h \in C_p} d^2(x_h, c_p)}$$

     where C_p is the set of all observations closest to centroid c_p and x_m belongs to C_p.
5. Repeat step 4 until k centroids are chosen.

In the basic version algorithm uses squared Euclidean distance, and each centroid is computed as the mean of the points in the cluster. Formula of distance measure is as follows:

$$d(x, c) = (x - c)(x - c)' \tag{1}$$

where x is an observation, c is a centroid and $(x - c)'$ is a transposition of vector $x - c$.

Besides squared Euclidean distance it is possible to use other distance measures. Authors limit their consideration to three additional measures: cityblock, cosine, correlation. Cityblock measure is $L1$ distance, i.e. sum of absolute differences. Centroid is computed as the component-wise median of the points in that cluster.

$$d(x, c) = \sum_{i=1}^{p} |x_i - c_i| \tag{2}$$

Cosine measure treats points as vectors. Distance is calculated as one minus the cosine of the included angle between points. Centroid is the mean of the points in that cluster, after normalizing those points to unit Euclidean length.

$$d(x, c) = 1 - \frac{xc'}{\sqrt{(xx')(cc')}} \tag{3}$$

Correlation measure treats points as sequence of values. Distance is calculated as one minus the sample correlation between points. Centroid is the component-wise mean of the points in that cluster, after centering and normalizing those points to zero mean and unit standard deviation.

$$d(x, c) = 1 - \frac{(x - \vec{\bar{x}})(c - \vec{\bar{c}})'}{\sqrt{(x - \vec{\bar{x}})(x - \vec{\bar{x}})'}\sqrt{(c - \vec{\bar{c}})(c - \vec{\bar{c}})'}} \tag{4}$$

where

$$\vec{\bar{x}} = \frac{1}{p}(\sum_{j=1}^{p} x_j)\vec{1}_p \tag{5}$$

$$\vec{\bar{c}} = \frac{1}{p}(\sum_{j=1}^{p} c_j)\vec{1}_p \tag{6}$$

and $\vec{1}_p$ is a row vector of p ones.

The *k*-means algorithm groups data points into several clusters. Each cluster can be interpreted as the separate mini-model domain. Each MM will use only this set of points which is contained in corresponding MM-domain (cluster created by k-means algorithm). We have to remember that MMs are not connected and to each other, and each MM have to be considered as separate one. This is consistent with previous versions of MM-method. The method do not create smooth surface in the whole modeled domain. This behavior is similar to *k*-NN algorithm which also belongs to instance-based learning methods.

Mathematical modeling on the mini-model domain

MM-method can use any method of mathematical modeling on the defined mini-model area. The paper describes MM using linear regression [14]. The selected method is a basic algorithm that can model only linear dependency. It is not complex enough for a mathematical modeling in the entire domain of real world dataset. However, we have to remember that we use this algorithm only to model some subset of data from the whole data set. The algorithm proceeds as follows:

1. Define the boundaries of MM-domains using *k*-means clustering algorithm.
2. Check to which mini-model domain (*k*-means cluster) query point belongs.
3. Learn linear regression model only on data samples contained inside mini-model domain:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon \tag{7}$$

4. Calculate the error committed by the mini-model on learning samples, and answer of the MM to the query point.

Figure 1 presents example division of input 2D-space with use of k-means clustering algorithm. Each polygon is a MM-domain. Figure 2 presents example mini-model in 3D-space

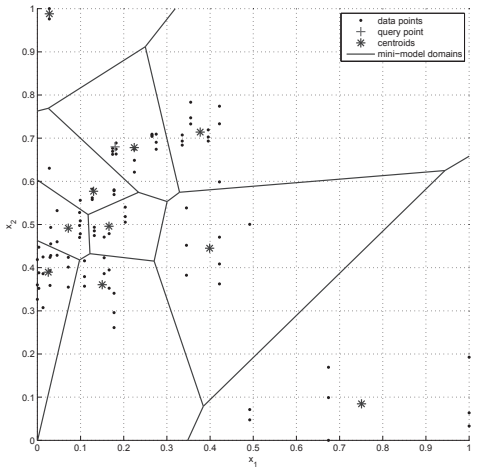

Fig. 1. Example of k-means division of input space.

Results of Experiments

At the beginning we have to notice that MM-method is sensitive to initial conditions due to properties of the *k*-means algorithm. The parameter *k* as well as initial location of centroids impact on the method results. Table 1 presets results of three different iteration of the algorithm. In every iteration authors compare impact of the *k* parameter on the MM accuracy. The table presents mean absolute error (MAE). We can observe that for the same *k* the results are slightly different in following iterations. In experiments authors use *k*-means++ algorithm to find initial centroid seed. It improves the running time of the algorithm and the quality of the final solution. Nev-

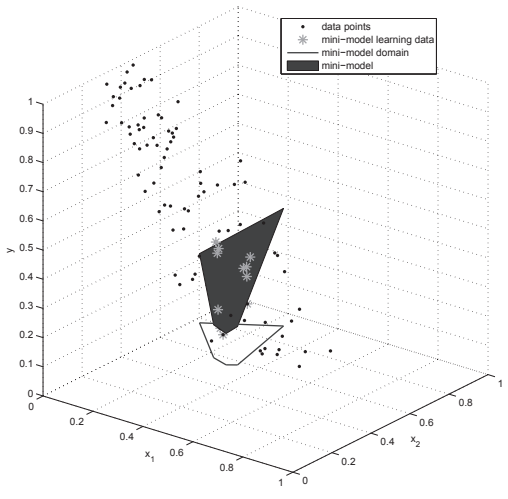

Fig. 2. Example of MM-method in the full space.

Table 1. Impact of the parameter *k* and initial random centroids location on the MM-method results.

k	1st trial	2nd trial	3rd trial
1	0.0818	0.0818	0.0818
2	0.0679	0.0679	0.0676
3	0.0642	0.0650	0.0648
4	0.0633	0.0624	0.0649
5	0.0626	0.0626	0.0645
6	0.0620	0.0609	0.0603
7	0.0601	0.0592	0.0587
8	0.0576	0.0606	0.0603

ertheless, the initial phase of the algorithm is heuristic and the final results depend on the initial centroids location. We have to remember that in previous version, the MM-method also check many possible MM-domains and choose the optimal one.

The proposed version of MM-method was compared to MM based on *n*-simplexes using the linear regression [3]. In the MM based on a simplex the dimension of the polytope was appropriate to the dimension of the input space. The method discarded the result for particular query point if it was unable to find a valid mini-model area. Results of experiments have shown that in this situation usually the error value was very high. The method was also compared with other instance-based learning algorithms: *k*-NN method, RBF network, GRNN network, local linear regression [15]. The MM-method works on points from local neighborhood and they were compared with use of leave-one-out cross validation method. The experiments were conducted on six datasets from UCI Machine Learning [16]:

- Boston Housing - (x_3 - proportion of non-retail business acres per town, x_5 - nitric oxides concentration (parts per 10 million), x_6 - average number of rooms per dwelling, x_7 - proportion of owner-occupied units built prior to 1940, x_{10} - full-value property-tax rate per $ 10,000, x_{11} - pupil-teacher ratio by town, x_{13} - % lower status of the population) (506 instances),
- Concrete Compressive Strength - (all available input attributes) (1029 instances),
- Auto MPG - (all available input attribute except: x_8 - origin, x_9 - car name) (391 instances),

Table 2. Comparison of effectiveness of tested methods.

		Housing	Concrete	Auto
MM with k-means using y-value				
sq. euclidean	error	0.0520	0.0722	0.0513
	k	16	20	6
cityblock	error	0.0545	0.0729	0.0525
	k	12	18	6
cosine	error	0.0539	0.0718	0.0516
	k	19	18	5
correlation	error	0.0589	0.737	0.0538
	k	9	18	5
MM with k-means not using y-value				
sq. euclidean	error	0.0530	0.0712	0.0531
	k	17	21	11
cityblock	error	0.0564	0.0724	0.0534
	k	16	21	6
cosine	error	0.0556	0.0702	0.0523
	k	15	26	4
correlation	error	0.0566	0.0702	0.0521
	k	13	32	5
Other methods				
MM based on simplex	error	0.0551	0.0483	0.0529
	samples	25 - 40	20 - 40	20 - 60
k-NN	error	0.0567	0.0722	0.0531
	k	4	1	3
Local linear regression	error	0.0534	0.0705	0.0498
	learn. error	0.08	0.06	0.09
GRNN	error	0.0545	0.0699	0.0502
	spread	0.1	0.03	0.08
RBF	error	0.0517	0.0511	0.0512
	spread	0.8	1.0	1.0
	learn. error	0.004	0.003	0.004

- Concrete Slump Test - (all available input attribute, output attribute: 28-day Compressive Strength) (102 instances),
- Yacht Hydrodynamics - (all available input attribute) (307 instances),
- Servo - (all available input attributes) (166 instances)

Results of experiments are presented in Table 2 and 3. Experiments were performed with the optimal values of all parameters for all tested methods. Mini-models, based on the *k*-means algorithm, were compared for four variants differing with the concepts of measures: squared euclidean, cityblock, cosine, correlation. In the data set “servo” some points have too small relative magnitudes and cosine measure cannot be used. Similar situation was with correlation measure because some points have too small relative standard deviations. The authors performed two different experiments. In the first experiment *k*-means algorithm uses input and output variables of data points i.e.: $x_1, x_2, \ldots, x_n, y$. However, in the part of assigning query point to specific cluster algorithm uses only input variables i.e.: $x_1, x_2, \ldots, x_n$. It is caused by the fact that query point does not have information about output vari-

Table 3. Comparison of effectiveness of tested methods.

		Slump	Yacht	Servo
MM with k-means using y-value				
sq. euclidean	error	0.0462	0.0427	0.0325
	k	3	20	45
cityblock	error	0.0373	0.473	0.0394
	k	3	94	127
cosine	error	0.0417	0.0414	0.0667
	k	3	94	12
correlation	error	0.0412	0.0424	0.1065
	k	5	80	6
MM with k-means not using y-value				
sq. euclidean	error	0.0460	0.0335	0.0365
	k	2	87	96
cityblock	error	0.0473	0.0332	0.0515
	k	3	126	99
cosine	error	0.0433	0.384	-
	k	4	112	-
correlation	error	0.0466	0.0363	-
	k	3	122	-
MM based on simplex	error	0.0500	0.0167	0.0493
	samples	8 - 20	12 - 25	10 - 25
k-NN	error	0.0600	0.0371	0.0437
	k	2	2	3
Local linear regression	error	0.0554	0.0296	0.0378
	learn. error	0.14	0.03	0.06
GRNN	error	0.0621	0.0385	0.0381
	spread	0.1	0.07	0.05
RBF	error	0.0257	0.0220	0.0528
	spread	2.0	2.9	0.7
	learn. error	0.0008	0.003	0.002

able y, what is our point of interest. In the second experiment *k*-means uses only input variables i.e.: $x_1, x_2, \ldots, x_n$. There is no rule which version is better, it depends on data set.

Conclusion

Mini-models have competitive accuracy with previously described version of mini-models and with other tested instance based-learning methods. It depends on data set which version of mini-models performs better. Using MM-method based on *k*-means gives an edge in the situation in which we have to calculate answer for several query points, because it frees us from the process of checking all possible MM-domains for all queries. It calculates local neighborhoods for each query points at once at the beginning of the learning process. Clustering algorithm makes the MM-method simpler and gives lower computational complexity. Although the proposed version is simpler, often it works more accurately than the previous one. Unfortunately MM-method is sensitive to initial conditions due to properties of *k*-means algorithm. The parameter *k* as well as initial location of centroids impact on the method results. We can observe that for the same *k* the results are slightly different in different iterations. However, we have to remember that in previous version of the MM-method also check many possible MM-domains and choose the optimal one. Authors in future research should move towards the use of mini-models with other clustering algorithms.

REFERENCES

[1] Piegat, A., Wasikowska, B., Korzeń, M.: Differences between the method of mini-models and the k-nearest neighbors an example of modeling unemployment rate in Poland. Information Systems in management IX-Business Intelligence and Knowledge Management. WULS Press, Warsaw, 34–43, (2011)

[2] Pietrzykowski, M.: Comparison between mini-models based on multidimensional polytopes and k-nearest neighbor method: case study of 4D and 5D problems. Series: Advances in Intelligent Systems and Computing, vol. 342, pp. 107–118, Springer Verlag, (2015)

[3] Pietrzykowski, M., Piegat, A.: Geometric Approach in Local Modeling: Learning of Mini-Models Based on n-Dimensional Simplex Artificial Intelligence and Soft Computing, Lecture Notes in Computer Science Volume 9120, pp. 460-470 (2015)

[4] Pluciński, M.: Mini-models - Local Regression Models for the Function Approximation Learning. In Rutkowski L. et. al. Proceedings of ICAISC 2012, Part II, LNCS 7268, Springer-Verlag Berlin Heidelberg, pp. 160–167, (2012)

[5] Pluciński, M.: Evaluation of the mini-models robustness to data uncertainty with the application of the information-gap theory. Artificial intelligence and soft computing : 12th International Conference, ICAISC 2013, Springer, 2013 pp. 230–241, (2013)

[6] Pluciński, M.: Application of Mini-Models to the Interval Information Granules Processing. Series: Advances in Intelligent Systems and Computing, vol. 342, pp. 37–48, Springer Verlag, (2015)

[7] Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, second edition, Prentice Hall. ISBN 0-13-080302-2, (2003)

[8] Samworth, R.J.: Optimal Weighted Nearest Neighbour Classifiers, Annals of Statistics, vol. 40(5), pp. 2733–2763, (2012)

[9] Nose, K., Lotufo, A.D.P., Minussi, C.R.: Short-Term Multinodal Load Forecasting Using a Modified General Regression Neural Network. IEEE Transactions on Power Delivery, vol. 26 (4), pp. 2862–2869, (2011)

[10] Lin, C.L., Wang, J.F., Chen, C.Y., Chen, C.W., Yen, C.W.: Improving the generalization performance of RBF neural networks using a linear regression technique. Expert Systems with Applications, vol. 36 (10), pp. 12049–12053, (2009)

[11] Jain, A. K. Data clustering: 50 years beyond K-means. IEEE Pattern Recognition Letters, vol. 31 (8), pp. 651-666, (2010)

[12] Celebi, M. E., Kingravi, H. A., Vela, P. A.: A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert systems with applications, vol. 40 (1), pp. 200–201, (2013)

[13] David, A., Vassilvitskii, S.: K-means++: The Advantages of Careful Seeding. SODA '07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1027–1035, (2007)

[14] Kutner, M. H., Nachtsheim, C. J., Neter, J., et al.: Applied Linear Statistical Models. McGraw-Hill/Irwin, ISBN 9780073108742, (2005)

[15] Ruppert D., Wand M.P.: Multivariate locally weighted least-squares regression. Annals of Statistics, vol. 22 (3), pp. 1346–1370, (1994)

[16] UCI MACHINE LEARNING REPOSITORY, *http://archive.ics.uci.edu/ml/*

***Authors**: Ph.D. Marcin Pluciński, M. Sc. Marcin Pietrzykowski, Faculty of Computer Science and Information Technology, West Pomeranian University of Technology, Żołnierska 49, 71-210 Szczecin, Poland, email: mpluciński@wi.zut.edu.pl, mpietrzykowski@wi.zut.edu.pl*