pattern recognition, nearest neighbour rule,
reference set condensation, reference set reduction,
bubble algorithms

Artur SIERSZEŃ[1]

BUBBLE ALGORITHM FOR THE REDUCTION OF REFERENCE

A vast majority of algorithms for the condensation of the reference set requires a great number of computations in case of processing a very large set, one that contains several dozens of objects. This fact formed the grounds for the presented attempt to develop a completely new classifier, an algorithm which would not only maintain the quality of classification similar to one obtained with the primary reference set but also allow to accelerate computations considerably. The proposed solution consists in covering the primary reference set with disjoint hyperspheres; however, these hyperspheres may contain objects from one class only. Classification is completed when it is determined that the classified point belongs to one of the mentioned spheres. If an object does not belong to any hypersphere, it is counted among the objects of the same class, to which the objects from the nearest hypersphere belong (the distance to the centre of the sphere minus the radius). As was indicated by the tests, this algorithm proved to be very efficient with very large sets.

1. INTRODUCTION

The most desired property of a classifier is a low percentage of incorrect decisions (the quality of classification). Apart from the quality of classification, the speed of classification is very important in some applications, e.g. when a classifier is used to analyse optic images.

In case of classifying very large sets, it is worth paying attention, among many solutions (including neural networks or decision trees), to classifiers which use distance functions. This group of solutions includes classifiers which do not have too strict assumptions and offer the quality of classification similar to the quality of the Bayes classifier, i.e. the classifier which is theoretically the best one [1]. This type of classifiers includes e.g. the *k* nearest neighbours rule (*k*-NN) [2] and its numerous modifications, e.g.:

- the concept of the surrounding neighbourhood and the *k*-NCN decision rule *(k - Nearest Centroid Neighbours*) resulting from it [7],
- the *k*-DNN rule (*k-Diplomatic Nearest Neighbours*) [10],
- local metrics [8, 9],
- the concept of the *k* -ENN rule (*k-Edited Nearest Neighbour*) [6],
- the *k* -NN rule with a pre-classifier of 1-NN type [4],
- the fuzzy *k* -NN rule [5, 3].

The *k*-NN rule assigns the classified object to the class which is represented by the most of its *k* nearest neighbours in the reference set. The reference set is conventionally the whole learning set.

2. SPEED OF 1-NN CLASSIFICATION ALGORITHM

In practical applications a significant role is played by the classifier operating according to the rule of the *nearest neighbour* type, one which is a particular case of the *k*-NN rule. It is the fastest version of the *k*-NN rule; moreover, a classifier of this type can be used to approximate other versions of the *k*-NN rule. To this purpose, re-classification of the reference set must be performed with an approximated classifier and then the 1-NN rule should be applied with the modified reference set. So far, many algorithms for the reduction of the reference set have been proposed in order to increase the speed of the 1-NN rule. New algorithms are being developed all the time. Another research direction aiming at the acceleration of the nearest neighbour rule is the condensation of the reference set, i.e. the creation of a set of artificial objects on the basis of the primary learning set. Only few methods of reference set

[1] Technical University of Łódź, Computer Engineering Department, Stefanowskiego 18/22, 90-924, Łódź, e-mail: asiersz@kis.p.lodz.pl.

condensation have been developed. Compared to the algorithms for the reduction of the reference set, they are able to steer the compromise between the size of the obtained condensed set, in other words the speed of classification, and the quality of classification measured with the percentage of correct decisions.

3. SPEED OF 1-NN CLASSIFICATION ALGORITHM

The presented algorithm for the condensation (reduction) of the reference set requires the modification of the decision rule. Therefore, the condensation process itself may be considered as the learning of a classifier. Since it is necessary to change the classification rule with this classifier, it is not only the learning phase, consisting in the condensation of the reference set, that requires detailed description; the classification phase requires it as well. The classification phase is not going to use the standard version of the 1-NN rule.

In the learning phase, the separated part of the set of objects with the known class affiliation, i.e. the learning set, is used to determine a set of hyperspheres. Each of the hyperspheres contains objects from the learning set, ones which belong to one class only. Such hyperspheres are described as homogenous. An homogenous hypersphere is described with the following parameters:

- the base point – a point from the learning set which is the centre of an homogenous hypersphere;
- the radius – a distance from the base point to the nearest point belonging to another class or to the edge of another homogenous area;
- the number of contained points – which determines how many points belonging to the same class as the base point (excluding the base point) is contained in an homogenous area.

The process of constructing homogenous hyperspheres is presented in the Figures 1 and 2. Its graphic interpretation gave the author an idea of the name for the developed method.

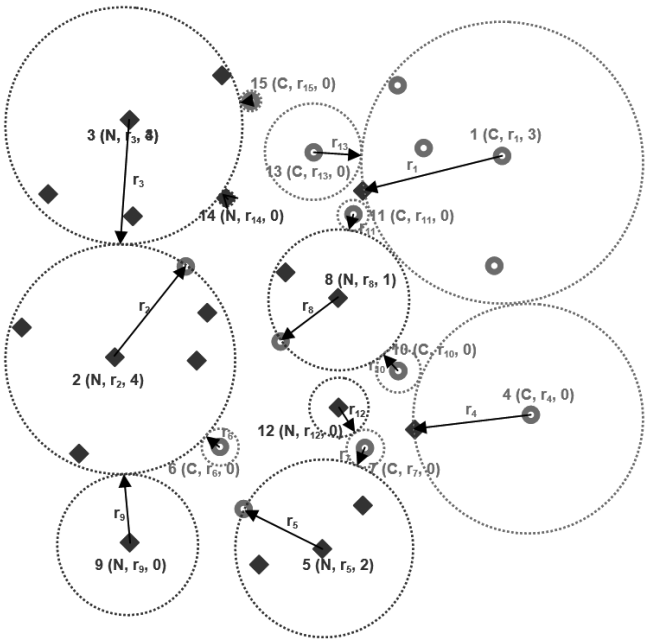

Fig. 1. The learning phase of the bubble classifier – the construction of homogenous hyperspheres (a construction diagram).

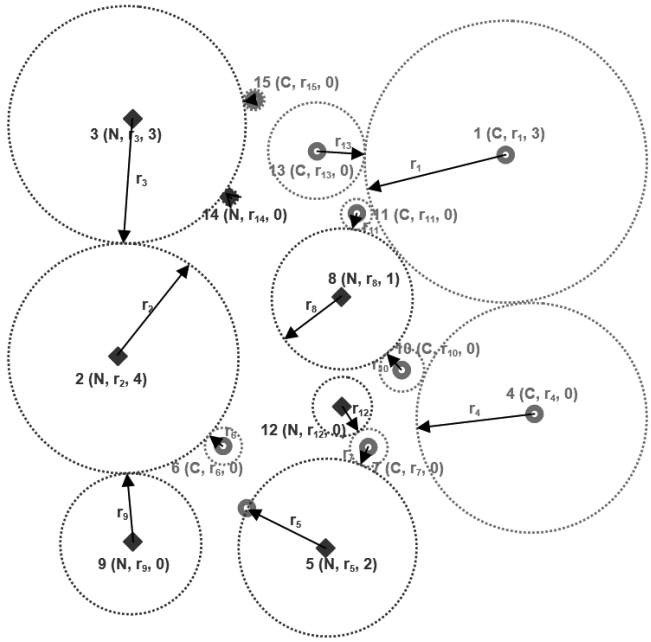

Fig. 2. The learning phase of the bubble classifier – the construction of homogenous hyperspheres (after the internal objects have been removed from hyperspheres).

The algorithm of the bubble classifier's learning phase chooses a random point from the learning subset (at the beginning from the whole learning set) and marks it as the base point. Then, the distance is determined from this base point to the nearest point belonging to another class or to the edge of another homogenous area. This distance is used to determine the range of a homogenous hypersphere. Points lying inside that area are sought, their number are remembered, and then these points are removed. The point from another class which was used to determine the range of the area is removed as well. The learning set is covered with hyperspheres. This algorithm is presented below in the form of the pseudocode.

00. START; $i = 2$; $T^1 = T$;
01. Choose a random point t_1 from T^1 set, the distance r_1 to the nearest point y_1 from another class and the number n_1 of points from T set located in the hypersphere $K^1=(t_1,r_1,n_1)$;
02. $T^i = T^{i-1} – Z^{i-1} – \{y_{i-1}\}$, if y_{i-1} does not exist, assume that $\{y_{i-1}\}$ is a empty set;
03. If T^i is empty, go to 06;
04. Choose a random point t_i from T^i set and determine $K^i=(t_i,r_i,n_i)$:
- distance d_1 from t_i to the nearest point y_i from another class,
- distance d_2 to the nearest, already existing, hypersphere K^j, $j=1,2,..,i$ (it is computed as a distance to the centre of the sphere minus its radius);
- mark the shorter of these two distances as r_i;
- determine a number of points n_i from T^i set located in the hypersphere with t_i centre and r_i radius;
05. $i=i+1$; go to 02;
06. END

where:

T – the learning set containing m objects;
T^i – sets reduced in individual iterations;
$K^i=(t_i,r_i,n_i)$ – a combination: a centre, a radius, a number of points inside a hypersphere without the base point;
Z^i – a set of points from T set located inside K^i sphere;

As a result of the learning phase of the bubble classifier, the set of homogenous hyperspheres is obtained. The next step consists in sorting them. The first choice criterion is the length of the radius

which determined the area (from the longest one to the shortest one). In case when several areas have the same radius, it is the areas that contain more reduced points that are chosen first. The sorting is supposed to accelerate the next phase of the bubble classifier's operation, to be more precise the classification phase.

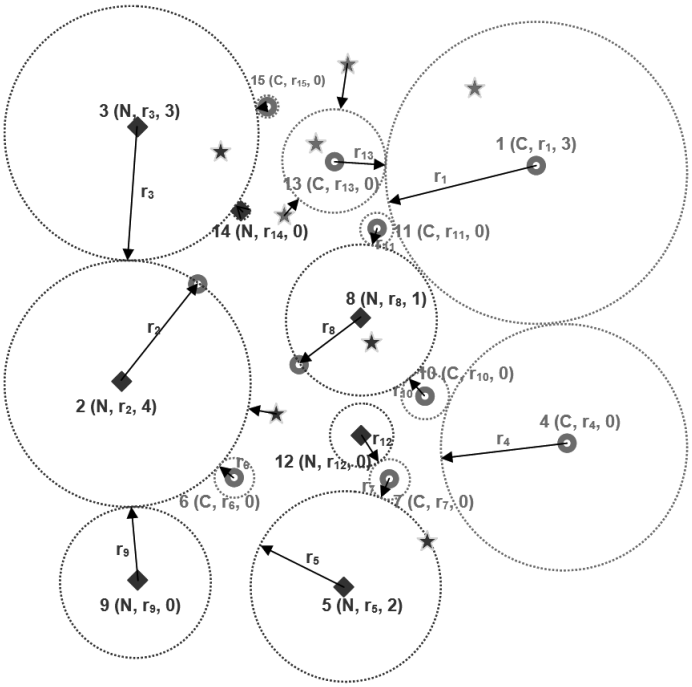

Fig. 3. The classification phase of the bubble algorithm.

Points are loaded from the set of objects for classification or from the testing set (in the Figure 3 these are represented by stars); then their distribution towards the determined homogenous hyperspheres is examined. Three cases are possible:

- a point lies inside an homogenous area – this point is assigned to the class to which this area is assigned;
- a point lies on the edge of an homogenous area – a point is assigned to the class to which this hypersphere is assigned;
- a point lies outside each homogenous hypersphere – a point is assigned to the class to which the nearest hypersphere is assigned (a fact that is determined with the distance to its edge).

4. EXPERIMENTAL RESULTS

The bubble classifier was implemented in C++ in Microsoft Visual Studio .NET 2003 environment and it was tested in Windows environment with the use of a PC computer equipped with Intel Pentium 4 HT 3GHz and 512MB of operating memory.

During the experiments and tests of the algorithm and during the verification of its operation the author focused mainly on the LIVER set, which is very large. It is a set of data describing ultrasound images of a liver. The most useful information is contained in the greyness level distribution of a pixel's examined neighbourhood. The set contains two classes of pixels (class 1 represents the area of cancer cells; class 2 represents the background, that is the areas of liver free from cancer cells). A general description of the set:

- a number of classes: 2,
- a number of properties: 13,
- a number of samples: 81968 (40000 – the first class, 41968 – the second class).

Table 1. Operation results (the average value, median, standard deviation, and the minimum and maximum value) of the bubble classifier algorithm obtained for the Liver set.

The division into the learning set and the testing set			Measured value	Average	Median	Standard deviation	Minimum value	Maximum value
1:9	Testing set [%]:	10	Measurement time [ms]	3542553	3540127	16410	3511195	3565009
	Learning set [%]:	90	Error comp. time [ms]	171721	171719	18	171696	171750
	Testing set [pcs]:	8196	Correct	8040	8047	22	8001	8069
	Learning set [pcs]:	73772	Incorrect	156	150	22	127	195
2:8	Testing set [%]:	20	Measurement time [ms]	2814996	2814975	20270	2814545	2815414
	Learning set [%]:	80	Error comp. time [ms]	315266	315093	383	314808	315969
	Testing set [pcs]:	16393	Correct	16054	16053	45	15993	16115
	Learning set [pcs]:	65575	Incorrect	339	340	45	268	400
3:7	Testing set [%]:	30	Measurement time [ms]	2413276	2406325	22267	2385875	2451989
	Learning set [%]:	70	Error comp. time [ms]	460658	459912	2715	456464	465264
	Testing set [pcs]:	24590	Correct	24097	24141	129	23889	24275
	Learning set [pcs]:	57378	Incorrect	493	450	129	315	701
4:6	Testing set [%]:	40	Measurement time [ms]	1848918	1853515	45283	1785811	1912294
	Learning set [%]:	60	Error comp. time [ms]	531744	531698	210	531382	531992
	Testing set [pcs]:	32787	Correct	32171	32208	115	31989	32339
	Learning set [pcs]:	49181	Incorrect	616	580	115	448	798
5:5	Testing set [%]:	50	Measurement time [ms]	2021178	2011238	60962	1923718	2127437
	Learning set [%]:	50	Error comp. time [ms]	692070	692194	7216	670985	704032
	Testing set [pcs]:	40984	Correct	39205	39204	53	39114	39286
	Learning set [pcs]:	40984	Incorrect	1779	1781	53	1698	1870
6:4	Testing set [%]:	60	Measurement time [ms]	1353924	1353609	7295	1340812	1364484
	Learning set [%]:	40	Error comp. time [ms]	681420	682124	5910	669453	690281
	Testing set [pcs]:	49180	Correct	45913	45877	213	45594	46284
	Learning set [pcs]:	32788	Incorrect	3267	3304	213	2896	3509
7:3	Testing set [%]:	70	Measurement time [ms]	740321	740488	1525	737937	742252
	Learning set [%]:	30	Error comp. time [ms]	569720	569691	2356	564750	572582
	Testing set [pcs]:	57377	Correct	52176	52260	208	51628	52498
	Learning set [pcs]:	24591	Incorrect	5201	5118	208	4879	5439
8:2	Testing set [%]:	80	Measurement time [ms]	323717	323724	567	322234	324453
	Learning set [%]:	20	Error comp. time [ms]	443449	444215	9496	427359	458344
	Testing set [pcs]:	65574	Correct	50992	51051	292	49788	51121
	Learning set [pcs]:	16394	Incorrect	14582	14524	292	14351	15786
9:1	Testing set [%]:	90	Measurement time [ms]	78794	78828	181	78515	79079
	Learning set [%]:	10	Error comp. time [ms]	266675	266330	6148	258266	275887
	Testing set [pcs]:	73771	Correct	48259	48519	1575	45460	49989
	Learning set [pcs]:	8197	Incorrect	25512	25252	1575	23247	28311

All tests were repeated 20 times. Each series was repeated for a different proportion of division into the testing set and the learning set (the following proportions were considered: 1 to 9; 2 to 8; 3 to 7; 4 to 6; 5 to 5; 6 to 4; 7 to 3; 8 to 2; 9 to 1). This way it was possible to thoroughly examine the influence of the learning and the testing sets' size on the classification quality and, above all, on the speed of algorithm's operation and classification error computation.

The Figure 4 presents the comparison of the average values obtained for the specific size of the learning set and the testing set. Additionally, the classification error obtained with the 1-NN method was shown, a factor which makes it easier to compare the quality of performed classification. In the figure, the x axis determines the proportion of the testing set to the learning set.

The table 1 presents the results (the average value, median, standard deviation, and the minimum and maximum value) of the bubble algorithm classifier for 20 measuring series repeated for nine different divisions of the *LIVER* set into the learning set and the testing set.

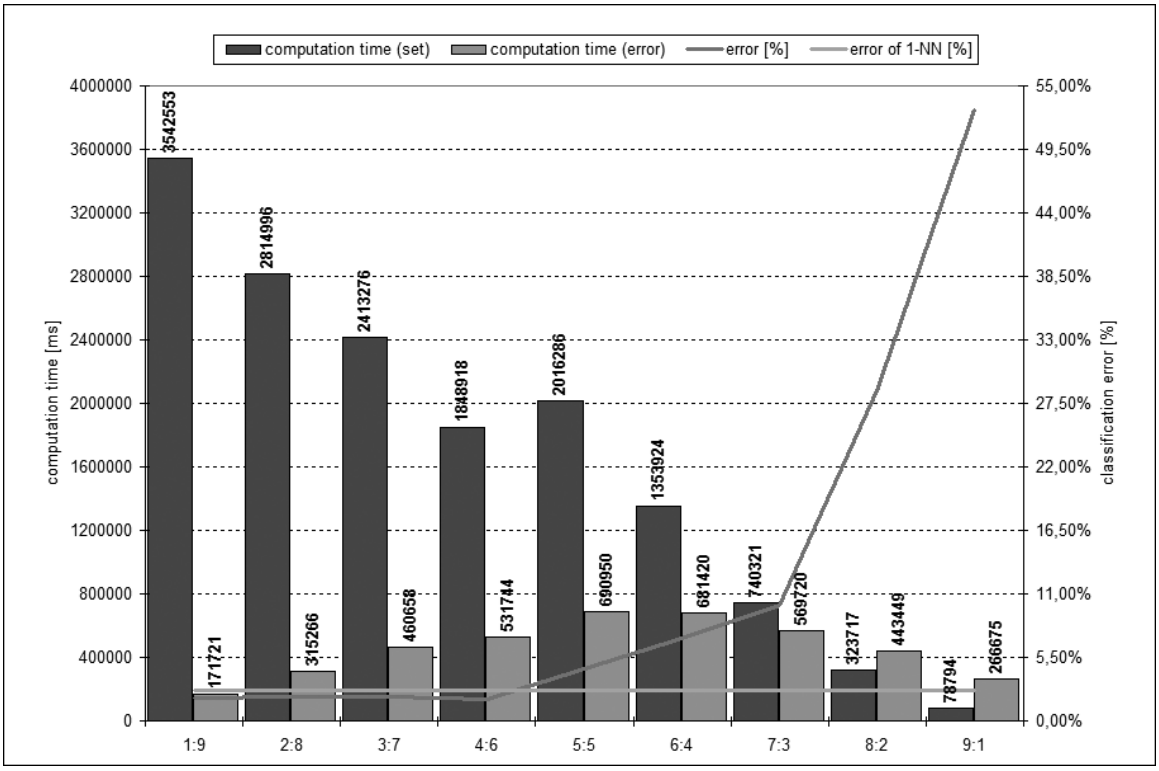

Fig. 4. The presentation of the average values of the algorithm's operation time, error calculation time and classification error for the LIVER set.

5. DISCUSSION

As can be seen in the Figure 4, in the first four test series for this set the algorithm obtained a better result of classification than in the case of the classification with the 1-NN method. The 1-NN method performs classification with the error amounting to 2.68%; the time required to obtain results with this method of classification is 4816.8 s (almost 80.5 min) with the use of the testing computer.

During the first measurement series (the testing set contained 8196 elements and the testing set contained 73772 elements), the bubble algorithm obtained the classification result amounting to 1.94% (the difference of 0.74% – over 1/4 better). The average computation time in the learning phase of this test series is 3542.6s (59 min) and for the classification phase it amounts to 171.8s (2.9 min) – in total over 1102.5s (18.4 min) better that the 1-NN method (this is an acceleration of above 22%).

The second series (the testing set contained 16393 elements and the learning set contained 65575 elements) gave a little worse classification quality, that is to say 2.11% (it was better than 1-NN anyway; the difference of 0.57 – over 1/5). The time required to obtain this result amounted to the average of 2815s (46.9 min) for the learning phase and 315.2s (5.3 min) for the classification phase. In total, these computations took the author 1686.5s (28 min) less compared to the 1-NN rule. This is an acceleration of 35%.

The third series (the testing set contained 24590 elements and the learning set contained 57378 elements) gave better classification quality than the former series. This result amounted to 2.05% (better than 1-NN; the difference of 0.63 – over 1/5). The learning phase computation time was 2413.2s (40.2 min) and the classification phase computation time was 460.7s (7.7 min). In total, these computations took the author 1942.8s (32.4 min) less that in case of the 1-NN rule. This is an acceleration of 40%.

The fourth measurement series (the testing set contained 32787 elements and the learning set contained 49181 elements) turned out to be the best among all discussed series. The obtained classification results was 1.92% (the difference of 0.76% – over 1/4 better). The computation time was the best as well. The learning phase for this series lasted for the average of 1489s (30.8 min) and the classification phase lasted for the average 531.7s (8.9 min). In total, both phases took the author 2436.1s

(40.6 min) less than computations in the 1-NN method, a difference which gave a considerable acceleration of over 50%.

The next measurement series gave deteriorated classification results. In the practical application this increase in incorrect decisions disqualifies the method; however, a good classification level and the considerable acceleration of computations in the first few series suggest that maintaining correct proportions between the testing set and the learning set may be beneficial. Tests performed with the use of the remaining sets revealed that the number of objects is also important apart from the proportions (in each case the first few series were the best concerning the classification quality, although none of them obtained the classification quality of the 1-NN method).

6. SUMMARY

In the article a new method was presented for the condensation of the reference set, i.e. the bubble classification algorithm. Its construction was inspired by a clear lack of algorithms which could manage a large number of computations performed during the processing of a very large set. This algorithm obtained a considerable acceleration of computations (50%) connected with the classification of the large LIVER set, simultaneously giving minimally better classification results.

Efforts should be taken to improve the presented solution. Further works are going to be focused on the optimum division into the learning part and the testing part so as to obtain the highest possible acceleration of computations with unchanged or even better classification quality.

BIBLIOGRAPHY

[1] DUDA R.O., HART P.E., STORK D.G.: Pattern Classification – Second Edition, John Wiley & Sons, Inc, 2001.
[2] FIX E., HODGES J.L. Jr., Discriminatory Analysis - Nonparametric Discrimination: Consistency Properties, Project 21-49-004, Report No. 4, US AF School of Aviation Medicine, Randolph Field, Tex., 1951, pp. 261–279.
[3] JÓŹWIK A., A learning scheme for a fuzzy k-NN rule, Pattern Recognition Letters, Vol. 1, No. 5–6, 1983, pp. 287–289.
[4] JÓŹWIK A., CHMIELEWSKI L., CUDNY W., SKŁODOWSKI M., A 1-NN preclassifier for fuzzy k-NN rule, 13th International Conference on Pattern Recognition, Vienna, Austria, Vol. IV, track D, 1996, pp. 234–238.
[5] KELLER J.M., GRAY M.R., GIVENS JR., J.A., A Fuzzy k-Nearest Neighbor Algorithm, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-15, No. 4, 1985, pp. 580–585.
[6] KUNCHEVA L.I., Reducing the Computational Demand of the Nearest Neighbor Classifier, School of Informatics, Symposium on Computing 2001, Aberystwyth, UK, 2001, pp. 61–64.
[7] SÁNCHEZ J.S., PLA F., FERRI F.J., On the use of neighbourhood-based non-parametric classifiers, Pattern Recognition Letters, Vol. 18, No. 11–13, 1997, pp. 1179–1186.
[8] HORT R.D., FUKUNAGA K., A new nearest neighbour distance measure, 5th IEEE International Conference on Pattern Recognition, Miami Beach, Florida, USA, 1980, pp. 81–86.
[9] SHORT R.D., FUKUNAGA K., The optimal distance measure for nearest neighbour classification, IEEE Transactions on Information Theory, Vol. IT-27, 1981, pp. 622–627.
[10] SIERRA B., LARRAÑAGA P., INZA I., K Diplomatic Nearest Neighbour: giving equal chance to all existing classes, Journal of Artificial Intelligence Research, 2000.

AKNOWLEDGEMENT

The author is a scholarship holder of project entitled "Innowacyjna dydaktyka bez ograniczeń – zintegrowany rozwój Politechniki Łódzkiej - zarządzanie uczelnią, nowoczesna oferta edukacyjna i wzmacnianie zdolności do zatrudniania, także osób niepełnosprawnych" supported by European Social Fund.

The author is a scholarship holder of project entitled "Innovative teaching without restrictions - integrated development of the Technical University of Lodz - university management, modern education and strengthening the ability to hire people, including the disabled" supported by European Social Fund.