Grey wolf optimizer combined with *k*-nn algorithm for clustering problem

Katarzyna Prokop

Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, Poland

Abstract

The clustering problem is an important task in machine learning. Clustering algorithms allow for the division of the set into individual clusters on the basis of a specific measure. Such an idea is used for undescribed data, where its label must be automatically assigned. One of the most popular algorithms is the k-nearest neighbors. In this paper, we propose a modification of this algorithm by combining it with heuristics, i.e. the grey wolf optimizer. The idea assumes that individuals in heuristic will be understood as a sample and unknown classes as victims in heuristic. Then the heuristic operation is used for analyzing the set. The proposition was described in terms of original algorithms and proposed hybridization of them. Then it was tested on Iris Flower Dataset and obtained results were discussed in terms of its advantages.

1. Introduction

Machine learning algorithms are known as data-hungry. It means, that many of them need a large number of samples to fit/train the model. However, in many cases, collected data are not labeled and cannot be used in the supervised training process. Therefore, the clustering method can be used to split them into some classes/clusters. An example of clustering visual features was presented in [1]. Another solution is modifying a k-means algorithm by introducing some dynamic changed conditions [2, 3]. Moreover, different approaches to clustering are modeled and it can be seen in the example of deep spectral clustering that uses an auto-encoder network [4].

Moreover, the optimization task is important in the area of machine learning. Therefore, many newer algorithms are modeled as an alternative and accurate approach [5]. Except for new models, the hybridization of them is introduced. One such example is a cooperative idea of many such algorithms [6]. The application of these algorithms shows that it is a promising approach and can help in different areas of artificial intelligence. For instance, meta-heuristic algorithms were used in the federated training process of convolutional neural networks [7, 8]. Also, it was combined to create a neuro-swarm heuristic for dynamics of covid19 [9]. Interesting solution was to use nature-inspired algorithms in image analysis [10, 11]. Heuristic algorithms were used for motion planning of aircraft [12], or others engineering problems. In most cases, engineering problems can be presented as an optimization task, where the best coefficients must be found to reach the best results [13]. Except using heuristics in hybridization and optimization, these algorithms are also used for feature extraction [14]. The extracted information can be used for describing an object and used in further classification.

In this paper, a hybridization of k-nn and selected heuristic algorithm was proposed. It is an alternative way that indicates that these two solutions can be combined and result in good accuracy. For the research the Grey Wolf Optimizer was chosen. This is a fairly young method [5], uses the hierarchy of units in the herd. This heuristic algorithm shows competitive results compared to other known metaheuristics for the function optimization problem. The Gray Wolf Optimizer can be successfully used, for example, in industry [15] or smart home solutions [16].

2. Methodology

The main idea is to combine the operation of k Nearest Neighbors classifier with Grey Wolf Optimizer and test the effectiveness of the method obtained as a result.

2.1. *k* Nearest Neighbors Algorithm

The k Nearest Neighbors classifier (k-nn) is an example of an algorithm that is used for finding the k most closely related items to the one that is considered, which makes classifying this object enabled. This algorithm is used for clustering, interpolation and even classification tasks [17, 18]. Therefore this algorithm determines the similarity between two objects using selected measures. Based on the results a group of the items with the least difference i s created. This set contains eponymous k elements. Objects reminding the considered item are called "neighbors". By the "voice of majority", they are responsible for assigning the tested object to the appropriate class. This means that obtained class is the most frequently appearing label among neighbors.

The algorithm is also used in a clustering problem. It is possible to create groups of elements with similar

IVUS 2022: 27th International Conference on Information Technology
ktn.prokop@gmail.com (K. Prokop)

features applying k-nn. There are many different ways of dividing the same dataset so various methods and their variable elements can be customized for this purpose.

Records in a given database and tested elements can be treated as vectors. Then the similarity between them may be calculated by a distance function. In this paper Euclidean metric and Manhattan metric were applied. Assuming a and b are two records being compared, where each of them consists of n attributes, the following vectors are obtained:

$$a = [a_1, ..., a_n],$$

$$b = [b_1, ..., b_n],$$

which means that attributes should take numerical values. Distance function between a and b for the Euclidean metric is defined as below:

$$d(a, b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}. \quad (1)$$

For the Manhattan metric, distance function can be described by:

$$d(a, b) = \sum_{i=1}^{n} a_i - b_i. \quad (2)$$

Let mark every i^{th} database's element with n attributes as $x^i = [x_1^i, ..., x_n^i]$, $i = 1, ..., m$. Thus m is the number of all records in the dataset. Obviously, the number k is less than or equal to m. Tested item can be presented as $y = [y_1, ..., y_n]$. By the pseudocode 1, with the selected distance function d, the k Nearest Neighbors Algorithm is shown.

Algorithm 1: *k* Nearest Neighbors Algorithm.

Input: dataset with m vectors x^i, unclassified vector y, neighbors number k
Output: y group

```
i := 1;
for i ≤ m do
    Calculate the distance from y to x^i using measure d;
    i++;
end
Sort the records in the database in ascending order relative to the calculated distances;
j := 1;
for j ≤ k do
    Make a note of the assigned group for x^j;
    j++;
end
Select the most popular group;
return y group;
```

2.2. Grey Wolf Optimizer

The method proposed in this paper besides the classifier also uses an optimizer. In detail, Grey Wolf Optimizer was applied. This algorithm is an example of a heuristic method of optimization which means that its aim is to find an approximate solution to a given problem but there is no guarantee of its correctness. Heuristics are useful in case of high resource cost or high computational complexity of classic methods.

Grey Wolf Optimizer was developed in 2014 [5] based on the behavior of a pack of wolves. Wolves are predators and live in herds where a hierarchy occurs. Every pack is led by a leader, the so-called male wolf. This individual is responsible for launching attacks. The male wolf is also the strongest wolf in the pack and initiates all pack's actions. It is selected from the herd by victories in direct battles with other wolves. An important role in the pack is also played by the second strongest wolf. With the male wolf, they complement each other. This individual takes command of the herd when the leader is indisposed. Further two groups of wolves are distinguished: the third level in the hierarchy are individuals who are doing fairly well and the last group consists of old and sick wolves. The tasks undertaken by the pack include mainly searching for food, i.e. hunting mammals. In nature, wolves hunt in various configurations: alone, in pairs, or as a whole pack.

Grey Wolf Optimizer uses a group hunting strategy. The different levels of the wolf hierarchy can be represented by the symbols: α – the male wolf, β – the second strongest wolf, δ – the third level of the hierarchy, and ω – old and sick wolves. In particular, the first three levels have an impact on the operation of the algorithm. Firstly, the pack consisting of a fixed number of wolves is initiated. A wolf is treated like a vector x:

$$x = [x_1, ..., x_n],$$

the values of which determine the wolf's location. The number n defines the dimension of a given problem that is a number of variables of the function $f(\cdot)$ wanted to optimize. In the initial pack, coordinates $x_1, ..., x_n$ are drawn from a given interval. Next, the three strongest individuals are selected: α, β, δ (the best wolf in the third level of hierarchy). This operation takes place by comparing the values of the function $f(\cdot)$ for all individuals in the herd. When a hierarchy is established, wolves move around in relation to the victim they are hunting. Since α is always the leader in the hunt, followed by β and δ, the position of the other wolves depends on the movements of the strongest individuals (because they are closest to the victim).

Assuming being in the j^{th} time step, the location of the wolf x in the next moment can be defined by equation 3:

$$x^{j+1} = \frac{X_A + X_B + X_D}{3}, \tag{3}$$

where

$$X_A = x_\alpha - A^j \cdot D_\alpha, \tag{4}$$

$$X_B = x_\beta - A^j \cdot D_\beta, \tag{5}$$

$$X_D = x_\delta - A^j \cdot D_\delta. \tag{6}$$

X_A, X_B, X_D are the coefficients depending on the positions of the best wolves at the moment (denoted as x_α, x_β, x_δ),
A^j is a parameter that updates in each iteration j:

$$A^j = 2 \cdot a^j \cdot p, \tag{7}$$

and p is a random value from the range $[0, 1]$. The value a^j depends on the interval $[a_{min}, a_{max}]$ set in the beginning and is calculated for each moment $j = 0, 1, \ldots, j_{max}$ according to the formula:

$$a^j = a_{max} - \frac{a_{max} - a_{min}}{j_{max}} \cdot j. \tag{8}$$

Usually, it is assumed that the a^j value decreases from 2 to 0 [15]. Then $a_{min} = 0$ and $a_{max} = 2$. The values D_α, D_β, D_δ evaluate the distance from the given individual x to the best adapted wolves:

$$D_\alpha = C^j \cdot x_\alpha - x, \tag{9}$$

$$D_\beta = C^j \cdot x_\beta - x, \tag{10}$$

$$D_\delta = C^j \cdot x_\delta - x, \tag{11}$$

where

$$C^j = 2 \cdot r, \tag{12}$$

which is recalculated for each iteration j and r, like p, is a random value in the range $[0, 1]$.

The above operations are performed a certain number of times (j_{max}) to finally select the best-adapted wolf (α) that is closest to the victim, i.e. the wanted solution. The scheme of the algorithm is presented in the pseudocode 2.

Algorithm 2: Grey Wolf Optimizer.

Input: wolves number w, iterations number j_{max}, range of the arguments, range a
Output: individual α

```
Generate initial pack with w individuals;
j := 0;
for j < j_max do
    Calculate a^j according to the equation (8);
    Calculate A^j according to the equation (7);
    Calculate C^j according to the equation (12);
    Find the best individuals α, β, δ;
    h := 0;
    for wolves do
        Calculate distances from the best individuals D_α, D_β, D_δ according to the equations (9), (10), (11);
        Calculate coefficients X_A, X_B, X_D according to the equations (4), (5), (6);
        Update wolf's location according to the equation (3);
        h++;
    end
    j++;
end
Find the best individuals α, β, δ;
return α;
```

2.3. Hybridization

Let $y = [y_1, ..., y_n]$ be an $n-$dimensional vector of unknown class as in the k Nearest Neighbors classifier model. Then y is identified with the victim that wolves hunt, while the pack consists of records from the database with n attributes, the values that are stored by wolves. Therefore, the population consists of units of the following form:

$$x^i = [x_1^i, ..., x_n^i], \tag{13}$$

where each i^{th} unit (wolf) stores information about the i^{th} n-dimensional record from the specified database. Naturally, the pack is the same size of m as the considered dataset. Next the strongest individuals α, β, δ have to be selected. Due to the necessity of hierarchy establishment, the values of the function $f(\cdot)$ for individual wolves have to be compared. The function $f(\cdot)$ corresponds to the Euclidean distance function (1) or distance for Manhattan metric (2). The strongest units are the wolves closest to the victim at the moment. When a hierarchy in the herd is established, the positions of the wolves are updated. Being in the j^{th} time step, the location of appropriate wolf in the next moment is described by the formula (3), which uses coefficients X_A, X_B, X_D represented by the equations (4), (5), (6). Each of these coefficients is a difference between location from one of the best wolves (α, β or δ) and product of the parameter A^j defined by the formula (7) and corresponding D coefficient (D_α, D_β or D_δ described by the equations (9), (10), (11)), respectively. The parameter A^j is modified in each iteration due to the variability of the parameter a^j (described by the equation (8)). Additionally, the value of the D coefficient is influenced by the value of the changing C^j parameter defined by the formula (12).

The above steps are repeated j_{max} times. Then, the

best wolf from the pack is selected (α). It is the record that after j_{max} iterations is at the shortest distance from the considered vector y out of the entire pack. Searching for such an individual takes place in total k times in order to identify k closest neighbors of the k Nearest Neighbors algorithm. Finally, according to the concept of the k Nearest Neighbors algorithm, the appropriate class for the y vector is selected based on the occurrence of individual classes among neighboring records.

The pseudocode 3 shows the structure of solving the classification problem using the Grey Wolf Optimizer.

Algorithm 3: Combination of k-nn with Grey Wolf Optimizer.

Input: dataset of m records (x^i), unclassified vector y, nearest neighbors number k, iterations number j_{max}, range of a
Output: y class

```
Generate initial pack consists m records;
Create array classes of length k;
i := 0;
for i < k do
    j := 0;
    for j < j_max do
        Calculate a^j according to the formula (8);
        Calculate A^j according to the formula (7);
        Calculate C^j according to the formula (12);
        Find the best individuals α, β, δ;
        h := 0;
        for wolves do
            Calculate distances from the best individuals D_α, D_β, D_δ according to the equations (9), (10), (11);
            Calculate coefficients X_A, X_B, X_D according to the equations (4), (5), (6);
            Update wolf's location according to the equation (3);
            h++;
        end
        j++;
    end
    Find the best individuals α, β, δ;
    Assign to classes[i] the class of individual α;
    i++;
end
Choose the most frequently repeated class in the array classes;
return y class;
```

3. Experiments

The hybrid method using Grey Wolf Optimizer and k-nn was tested in a process of matching the class to the object from a database. The database of iris flowers was used for experiments. The author of this dataset, the British biologist and statistician Ronald Fisher [19], shared data in 1936.

The iris flowers dataset consists of 150 records describing the appearance of these plants. Every record stores information about 5 attributes: the length of the plot of the flower cup, the width of the plot, the length of the petal and its width, and also the name of the species. Thus the first four characteristics are expressed by numerical value and the fifth one constitutes a class label presented in a text form. Three species of iris are included in the collection: *Iris setosa*, *Iris virginica*, *Iris versicolor*.

In order to test the designed method, a program was implemented. 20% of all records were checked whether the appropriate class was matched. For calculations, the program retrieved the first four attributes of each record omitting the class labels. However, labels were stored for later comparison to obtaining results with the actual state. As it was earlier described, every record was treated like a vector to use the created method. To determine effectiveness of the method, the following coefficient acc was defined:

$$acc = \frac{n_c}{n_t} \cdot 100\%, \tag{14}$$

where n_c is the number of correct matches and n_t is equal to the number of all tested records. The final result was rounded to two decimal places.

The program was tested for four variants of iterations number. A range of a in all cases was assumed to $[0; 2]$. For each option effectiveness of the method was checked for Euclidean distance function 1 and also for distance in Manhattan metric 2 by launching the program five times in both cases and calculating the arithmetic mean of the obtained results. These operations were performed for 15 different values of k parameter: $k = 1, ..., 15$.

The first variant of iterations number was $j_{max} = 100$. For the Euclidean metric, the highest arithmetic means of acc value was obtained for $k = 2$ and reached 71.33%. This value did not fall below 34.00%. In the case of Manhattan distance, the arithmetic means of the acc coefficient assumed values between 26.00% and 44.00%. Detailed results are placed in Table 1.

In the second test, the number of iterations was modified to 25. The obtained results are presented in the table 2. It turned out that for Euclidean distance significant improvement of effectiveness was achieved regardless of the value of k. In this case, the arithmetic mean of acc was greater than or equal to 90.00%. It means that nearly all of the tested records were correctly classified. For the Manhattan distance, the results were similar to the first

Table 1
Arithmetic mean of the acc coefficient for $j_{max} = 100$, $a \in [0; 2]$.

k	Euclidean distance	Manhattan distance
1	55.33%	31.33%
2	71.33%	33.33%
3	62.00%	43.33%
4	44.00%	34.67%
5	34.00%	37.33%
6	41.33%	28.67%
7	64.67%	38.67%
8	70.67%	28.67%
9	47.33%	34.66%
10	53.33%	35.33%
11	65.33%	27.33%
12	53.33%	41.33%
13	58.00%	44.00%
14	58.67%	30.67%
15	48.00%	26.00%

Table 2
Arithmetic mean of the acc coefficient for $j_{max} = 25$, $a \in [0; 2]$.

k	Euclidean distance	Manhattan distance
1	96.67%	37.33%
2	93.33%	42.67%
3	91.33%	37.33%
4	94.67%	33.33%
5	92.67%	38.66%
6	91.33%	32.67%
7	94.67%	36.67%
8	90.00%	27.33%
9	92.00%	42.67%
10	92.00%	37.33%
11	94.67%	37.33%
12	94.67%	37.33%
13	92.00%	40.00%
14	97.33%	35.33%
15	93.33%	38.00%

test. The arithmetic mean between 27.33% and 42.67% was obtained.

Another test was performed for 50 iterations. As in the previous step, results for the Manhattan metric did not improve noticeably. The highest arithmetic mean 38.67% can be observed for $k = 4$. 30.67% is the lowest obtained value. For Euclidean metric results are not as good as in the previous test but there were much more correctly classified records than for 100 iterations, for example when $k = 4$ it was 72.00% in this variant ($j_{max} = 50$) and 44.00% when $j_{max} = 100$. Other values are presented in table 3.

The last test assumed $j_{max} = 1000$ so the number of iterations significantly increased. Table 4 presents the arithmetic mean of the acc coefficient in this variant. Again, the distance function for Manhattan brought results similar to other tests. None of the values of parameter k causes results markedly better than others. If the Euclidean metric is applied, results depend on k value. For example, when $k = 7$, the arithmetic mean of acc is equal to 30.67% and it is the lowest result. Simultaneously, for $k = 11$ it is 71.33%. Thus the range of these results is quite big – almost 40%.

Table 3
Arithmetic mean of the acc coefficient for $j_{max} = 50$, $a \in [0; 2]$.

k	Euclidean distance	Manhattan distance
1	65.33%	35.33%
2	61.33%	34.67%
3	70.67%	30.67%
4	72.00%	38.67%
5	59.33%	36.00%
6	71.33%	36.00%
7	62.66%	34.00%
8	67.33%	36.66%
9	70.00%	31.33%
10	64.00%	30.67%
11	66.67%	34.67%
12	67.33%	36.00%
13	62.67%	36.00%
14	72.00%	34.00%
15	60.00%	30.67%

Table 4
Arithmetic mean of the acc coefficient for $j_{max} = 1000$, $a \in [0; 2]$.

k	Euclidean distance	Manhattan distance
1	36.67%	38.67%
2	32.00%	37.33%
3	38.00%	30.67%
4	42.67%	35.33%
5	42.00%	32.00%
6	47.33%	33.33%
7	30.67%	39.33%
8	34.00%	26.67%
9	57.33%	29.34%
10	44.67%	31.33%
11	71.33%	30.00%
12	61.33%	33.33%
13	60.67%	33.33%
14	56.00%	34.00%
15	40.66%	33.33%

4. Conclusions

The effectiveness of the method obtained from a combination of k-nn with Grey Wolf Optimizer depends on established features. Using this method and appropriate input parameters can bring satisfactory results – for example as in the test for $j_{max} = 25$ and the Euclidean metric. Comparing other values, it can be observed that with the problem described in this paper, the distance function for the Manhattan metric is not very effective. The correctness of matches using this metric is low. On the other hand, in some cases, the value of k parameter also has an influence on results. The fourth performed test proves it. In connection with the above, this hybrid method can be useful with appropriate assumptions. However, it requires more experimentation for specific cases.

References

[1] M. Caron, P. Bojanowski, A. Joulin, M. Douze, Deep clustering for unsupervised learning of visual features, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 132–149.

[2] M. Z. Hossain, M. N. Akhtar, R. B. Ahmad, M. Rahman, A dynamic k-means clustering for data mining, Indonesian Journal of Electrical engineering and computer science 13 (2019) 521–526.

[3] K. P. Sinaga, M.-S. Yang, Unsupervised k-means clustering algorithm, IEEE access 8 (2020) 80716–80727.

[4] X. Yang, C. Deng, F. Zheng, J. Yan, W. Liu, Deep spectral clustering using dual autoencoder network, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 4066–4075.

[5] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, Advances in engineering software 69 (2014) 46–61.

[6] M. Abd Elaziz, A. A. Ewees, N. Neggaz, R. A. Ibrahim, M. A. Al-qaness, S. Lu, Cooperative meta-heuristic algorithms for global optimization problems, Expert Systems with Applications 176 (2021) 114788.

[7] D. Połap, M. Woźniak, A hybridization of distributed policy and heuristic augmentation for improving federated learning approach, Neural Networks 146 (2022) 130–140.

[8] D. Połap, M. Woźniak, Meta-heuristic as manager in federated learning approaches for image processing purposes, Applied Soft Computing 113 (2021) 107872.

[9] M. Umar, Z. Sabir, M. A. Z. Raja, F. Amin, T. Saeed, Y. Guerrero-Sanchez, Integrated neuro-swarm heuristic with interior-point for nonlinear sitr model for dynamics of novel covid-19, Alexandria Engineering Journal 60 (2021) 2811–2824.

[10] D. Połap, M. Woźniak, R. Damaševičius, R. Maskeliūnas, Bio-inspired voice evaluation mechanism, Applied Soft Computing 80 (2019) 342–357.

[11] D. Połap, N. Wawrzyniak, M. Włodarczyk-Sielicka, Side-scan sonar analysis using roi analysis and deep neural networks, IEEE Transactions on Geoscience and Remote Sensing (2022).

[12] Y. Wu, A survey on population-based meta-heuristic algorithms for motion planning of aircraft, Swarm and Evolutionary Computation 62 (2021) 100844.

[13] G. Dhiman, Ssc: A hybrid nature-inspired meta-heuristic optimization algorithm for engineering applications, Knowledge-Based Systems 222 (2021) 106926.

[14] M. Sharma, P. Kaur, A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem, Archives of Computational Methods in Engineering 28 (2021) 1103–1127.

[15] Ł. Knypiński, L. Nowak, Zastosowanie algorytmu szarych wilków do rozwiązania zadań optymalizacji urządzeń elektromagnetycznych, Poznan University of Technology Academic Journals. Electrical Engineering (2019).

[16] S. N. Makhadmeh, A. T. Khader, M. A. Al-Betar, S. Naim, An optimal power scheduling for smart home appliances with smart battery using grey wolf optimizer, in: 2018 8th IEEE international conference on control system, computing and engineering (ICCSCE), IEEE, 2018, pp. 76–81.

[17] M. Włodarczyk-Sielicka, N. Wawrzyniak, Problem of bathymetric big data interpolation for inland mobile navigation system, in: International Conference on Information and Software Technologies, Springer, 2017, pp. 611–621.

[18] D. Zhao, X. Hu, S. Xiong, J. Tian, J. Xiang, J. Zhou, H. Li, K-means clustering and knn classification based on negative databases, Applied Soft Computing 110 (2021) 107732.

[19] R. A. Fisher, The use of multiple measurements in taxonomic problems, Annals of eugenics 7 (1936) 179–188.