BIJIT - BVICAM's International Journal of Information Technology
Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM), New Delhi (INDIA)

Optimization of KNN with Firefly Algorithm

Alka Lamba[1] and **Dharmender Kumar**[2]

Submitted in April, 2016; Accepted in July, 2016

Abstract – Data mining has turned out to be a milestone in information industry. The need of data mining tools can be evidenced in almost every field. Classification is one of the data mining techniques which are used for knowledge discovery. Out of the various alternatives to evolve a classification model, KNN is a very popular and apprehensible one. Although, KNN incorporates a number of limitations in it but these can be bumped-off by making some alterations to the standard KNN algorithm. Numerous variants of KNN have been proposed by many researchers in previously done studies and they have also outperformed the standard KNN. In present study, a modified version of KNN algorithm has been proposed which commingles firefly algorithm with standard KNN. The performance of this modified algorithm is examined with respect to the standard KNN and it is found that the proposed algorithm works well in case of large data sets.

Index Terms – classification; data mining; firefly; KNN; self-adaptive

1.0 INTRODUCTION

Today the data is increasing by leaps and bounds. The availability of abundant amount of data has increased the necessity of data mining tools. These tools help in exploring data in such a way that it results in obtaining some crucial information. These results of data mining can be utilized to make important decisions in various fields such as marketing, financial data analysis, medical science, intrusion detection, retail industry etc[1],[2],[3]. Data mining offers different data mining techniques which are used for mining knowledge from data i.e. clustering, classification, association rule, prediction, outlier detection. An introduction to these data mining techniques and their applications are given in [4]. Classification is one of the data mining techniques which are used frequently. In this data mining technique, a classification model is built which is called classifier. The model is capable of classifying a data tuple. Classification constructs this classifier using a class-labeled data set. This is the reason why classification is said to be an example of supervised learning. The process of classification starts with partitioning of the data set into two sets: training set and test set.

[1]Master of Technology (CSE), Guru Jambheshwar University of Science and Technology, Hisar, Haryana, India.
[2]Associate Professor (CSE), Guru Jambheshwar University of Science and Technology, Hisar, Haryana, India.
[1]alkalamba07@gmail.com and
[2]dharmindia24@gmail.com

After this, two steps are followed. The construction of a classifier using a pre-classified training data set is the first step and the assessment of the constructed classifier using test set is the second step. The second step testifies that how good is the built classifier in predicting class labels for unknown tuples. There are several classification algorithms to carry out the process of classification for example k-Nearest Neighbor (KNN), Bayesian Classifier, Decision Tree Induction, Support Vector Machine (SVM), Classification using Back-Propagation, Rule-Based Classification etc. [5]. A comparative study of these classification algorithms is projected in [6]. Out of these, KNN is the simplest and most comprehensible. KNN employs the nearest neighbor technique where the classification of an unseen tuple is done using similar data tuples to it. K-Nearest Neighbor classifies a data tuple on the basis of class-labels of the k nearest data tuples to it in the data set. The k is assumed to be a positive integer and passed as input to the KNN algorithm. KNN algorithm is a lazy learner with non-parametric nature [7]. Unlike parametric methods the non-parametric methods does not make any presumption about the shape of the classification model. The reason of categorizing KNN as a lazy learner and rest of the classification algorithm as eager learners is that KNN does not construct any classifier as done normally by other classification algorithms on getting training data tuples. All the calculations are done only at the time of classification of an unseen tuple. On account of this working principle of KNN, another name of it is instance-based learner.

Besides simplicity, there are many plus points of KNN. It is scalable. No prior knowledge is required by it regarding the data set. It gives quite good results when compared to the results given by other classification algorithms. Insensitivity towards the noisy data adds another element to its list of merits. Despite of a number of positives, many negatives are associated with KNN. k is the only parameter that KNN takes as input. But it a very challenging task to determine the appropriate value of k. It is so because taking the small value of k would increase variance of the obtained model and large value of k would increase bias of the resulting model [8]. And it is well- known that we have to get a trade-off between variance and bias to construct a good model. Another issue with KNN is that it requires a lot of memory to store the training tuples due to its lazy nature. This can be managed in case of small data sets but with large data sets it becomes very nasty. The large computational cost is another demerit of KNN. The cause of this increased computational cost is the manner in which KNN classifies a data tuple. In KNN, for classification of a novel tuple the k nearest tuples to it are needed to be determined. And this is accomplished by computing the distance of the novel tuple with all the training tuples. It raises the computational cost of the algorithm.

Owing to these infirmities of KNN, many researchers have proposed reformed versions of KNN. Different variants of KNN are discussed in [9]. These variants tried to alleviate the shortcomings of KNN algorithm. The results of KNN algorithm are influenced by numerous key factors such as the value of k, distance metric which is used for computing similarity between any two tuples, the weights of attributes in a data set. By manipulating these key factors the performance of KNN can be ameliorated. [10] chooses the value of k by consulting local neighborhood of the data tuple which is to be classified. In the primitive KNN all the attributes of data set give equal contribution in classification of unprecedented tuple. But not all the attributes are significant. This can contaminate the results of KNN. [11] proposes an algorithm which is called weight adjusted KNN. In this proposed algorithm the enumeration of weights for each of the feature in the data set is done. [11] exerted the proposed weight adjusted KNN for text categorization. [12] uncovered the aspect that the classes in the data set are not evenly dispersed. There can be more number of tuples embraced in one class than other. By dint of which the outcome can be biased towards the class which encompasses larger number of tuples. [12] suggested a solution for this by using different values of k for distinct classes according to the number of tuples contained by them. Small value of k is used for the class that has fewer tuples and large value of k is used for the class that has large number of tuples. To lessen the computational cost incurred in KNN, an amended variant of KNN called adaptive KNN is proffered in [13]. It uses a non-fixed value of k instead of a fixed value along with some early-break heuristics. [14] discusses various extensions for KNN which are density- based KNN classifier, variable KNN classifier, weighted KNN classifier, class-based KNN classifier and discernibility KNN. All of them use different methodology to uproot the flaws in standard KNN. Another variant of KNN is presented in [15] which performs classification of a data tuple using shared neighbors. To gauge similarity between any two tuples, it uses BM25 similarity measure. To confine the number of neighbors that can vote for classification of a novel tuple a threshold is set. An amalgam of clustering algorithm K-Means with KNN is proposed in [16]. This amalgamation tried to reduce the computational cost of KNN. To enhance the accuracy attained from standard KNN, distance metric plays a very vital role. Standard KNN generally employs Euclidean distance metric. A new distance metric is introduced in [17] which is called Mahalanobis distance metric. The advantage of using this distance metric in place of Euclidean is that the correlation between data tuples is also reckoned by it. One more distance metric, informativeness is introduced in [18]. The algorithm proposed in [18] takes two parameters as inputs which are k and I. Firstly, k nearest neighbors to the unseen data tuple is determined. After this, informativeness for each nearest neighbor is evaluated. Out of these, only I most informative data tuples are considered to vote in classifying any tuple. [19] proposed an algorithm which is combination of a number of KNN classifiers. Each classifier is trained on different part of data set. To find the class of a test tuple, the class of the test tuple is enumerated using all classifiers. The majority class will be the class of test tuple.

Soft computing which offers information processing when united with data mining in a creative way, then this formation can be used efficaciously for knowledge discovery in large databases. [20], [21], [22], [23] elaborates that how soft computing helps in carrying out a better data analysis. Many data mining tasks can be expressed as optimization problems such as feature selection, clustering, classification etc. And soft computing can be used to find approximate solutions for these optimization problems. The principal components of soft computing include Fuzzy Logic, Rough Sets, Neural Networks and Evolutionary Computing. Further evolutionary computing comprises of two kinds of algorithm: Evolutionary algorithms and Meta-heuristic algorithms [24]. Evolutionary algorithms include genetic algorithm and differential algorithm whereas meta-heuristic algorithms embrace cuckoo search, particle swarm optimization (PSO), firefly algorithm, ant colony optimization (ACO), artificial bee colony (ABC), Bayesian network etc. [25] discusses about all these nature-inspired meta-heuristic algorithms. There are various studies that have been carried out earlier which demonstrate that how well these meta-heuristic algorithms perform in case of classification. [26], [27], [28] have deployed Ant Colony Optimization algorithm for classification. [29],[30] have used Particle Swarm Optimization algorithm for classification. Artificial Bee Colony algorithm is used for image classification in [31]. In [32] a hybrid fuzzy firefly algorithm is used to derive classification rules. [33] proposed a fuzzy classification system. Some of these principal components of soft computing are infused with KNN in earlier done studies. [34] proposed a fuzzy version of KNN algorithm. In contrast to KNN, which gives crisp membership of the tuples, it gives fuzzy membership. A fuzzy-rough nearest algorithm is proposed in [35] which combines rough set theory and fuzzy set theory. The proposed algorithm employs Rough set theory to compute the lower and upper approximations of classes using nearest neighbors [36]. Test tuple is classified based on its membership in these approximations. Being a powerful optimization tool [37], genetic algorithm has also been implanted with KNN. A hybrid version of KNN with genetic algorithm is proposed in [38]. Instead of using any distance metric, it utilizes genetic algorithm for determining the k nearest data tuples. [39] has used genetic algorithm with KNN differently. It exercised genetic algorithm for extracting worthy features from a data set. [40] indulges ACO algorithm with KNN to pick out good features from a data set. PSO algorithm has been integrated with KNN divergently. In [41], PSO is used to find representatives of distinct classes in the data set. The representatives will now be the new training data tuples and KNN will be implemented on these new tuples for classification of a novel tuple. [42] made use of PSO for figuring out weights for features in the data set. ABC algorithm has also been used similar to that of PSO with KNN. [43] practiced ABC for extracting good features from a data set and [44] implemented ABC to find representatives of distinct classes as done in [40].

[43] put to use the combination of ABC and KNN for diagnosing coronary heart disease.

After studying the erstwhile studies and weaknesses of KNN, it can be concluded that standard KNN algorithm can be refined further in order to grab good accuracy. Taking inspiration from these, this study proffers another variant of KNN which would conglomerate firefly algorithm and KNN. Firefly algorithm is inspired from the flashing behavior of fireflies. [45], [46], [47], [48] scrutinizes the performance of firefly algorithm. Many variants of firefly algorithm can be seen in former studies. A randomization term is needed in firefly algorithm which comprises of two parameters α and ε. α is called randomization parameter and it decides the next place to search for solution in the search space or explicitly it defines the step size for a firefly. ε is a vector of random numbers which is originated from a probability distribution method. There are numerous methods to draw this vector such as uniform distribution, Gaussian distribution, levy flights distribution etc. [49] proposed a variant of firefly algorithm called Levy Flights Firefly algorithm. The algorithm draws the random number vector using Levy Flights distribution. The firefly algorithm makes an assumption that if all the fireflies have same brightness then they will move randomly. [50] suggested that rather than moving randomly they should move towards the global best. Also it employed Gaussian distribution for drawing ε. Another variant of firefly was proposed in [51] which is called self-adaptive step firefly algorithm. The self-adaptive step firefly algorithm computes step size for each firefly according to its fitness values in previous generations. In addition to flashing light of fireflies the algorithm propounded in [52] considers some other affecting parameters also i.e. gene exchange of firefly, its pheromone and the dispersion of the pheromone due to wind. In this research paper, self-adaptive step firefly algorithm is opted to infuse into standard KNN algorithm.

In the subsequent sections of present research paper we will learn about KNN, firefly algorithm and then the proposed modified KNN algorithm. The performances of the KNN and the propounded algorithm will be monitored on six data sets. The data sets are taken from UCI Repository and Keel.

2.0 K-NEAREST NEIGHBOR ALGORITHM

KNN algorithm is a very popular classification technique. It can be put into practice very easily. The prerequisites for the KNN algorithm are: a class-labeled data set and the input parameter k. The value of input parameter k would resolve that how many nearest neighbors are to be taken into account for classification of any tuple. The procedure of classifying any tuple using KNN is straightforward. Initially, the data set is bifurcated. The two subsets are called training set and test set. The part of both is same as they have in classification. Later on k nearest data tuples to unseen tuple from the training set are determined. The class which has majority in these unearthed k data tuples is assigned to the unseen tuple, which is to be classified. Test set will compute the accuracy of the KNN algorithm. Pseudo code for the standard KNN algorithm is given below:

ALGORITHM I

Input Parameters: Data set, k
Output: Classified test tuples
Step 1: Store all the training tuples.
Step 2: for each test tuple
A. Compute distance of it with all the training tuples using (1).
B. Find the k nearest training tuples to the test tuple.
C. The class which is most common in the k nearest training tuples to the test tuple is assigned to the test tuple.
End for

Each tuple in data set can be viewed as a data point in the n-dimensional space, where n is the number of attributes describing the data set. The distance between the data points is computed generally using Euclidean distance. Euclidean distance between two data tuples x and y is given below:

$$\sqrt{\sum_{1 \le i \le n}(x_i - y_i)^2} \qquad (1)$$

n = number of attributes in data set

x_i and y_i are values of attribute i in data tuples x and y respectively. Manhattan distance and Minkowski distance are some other distance metrics which can also be used.

The simplest case of k-nearest neighbor algorithm is when k is taken to be 1. This case is called nearest neighbor rule, where the class assigned to the unseen tuple is the class of most nearest tuple to it. Another property of KNN is that it can be employed not only for predicting a categorical attribute but also for predicting a continuous valued attribute. The later one is called regression. In regression, the value of class attribute of an unseen tuple will be the average of the class attribute values of the k nearest tuples to the unseen tuple.

3.0 FIREFLY ALGORITHM

Firefly algorithm is meta-heuristic in nature and is used to find an approximate solution for an optimization problem. Flashing behavior of the fireflies is inspiration of the firefly algorithm. There are three assumptions made in the firefly algorithm:

- Any firefly can be attracted towards any other firefly.
- The attractiveness is relative to brightness of the firefly. Brighter firefly would attract all other fireflies having less brightness than the brighter firefly.
- When all fireflies have same brightness then they will move randomly.

The attractiveness of a firefly is calculated using following function:

$$\beta(r) = \beta_0 . e^{-\gamma r^2} \qquad (2)$$

where β_0 is the attractiveness of the firefly when r = 0 and γ is light absorption coefficient. The firefly's movement totally depends on its attractiveness. Firefly i would move towards firefly j if and only the attractiveness of the firefly j is greater

than that of firefly i. In that case, the movement is shown by following formula:

$$x_{ik} = x_{ik} + \beta_0 . e^{-\gamma r_{ij}^2} . (x_{ik} - y_{jk}) + \alpha . S_k . (rand_{ik} - 0.5) \quad (3)$$

x_{ik} and y_{jk} are values of attribute k. k takes values from 1,2……n, where n is the dimension of the data set. $rand_{ik}$ is a random number between 0 and 1. α is called randomization parameter which will decide how much to move and takes value between 0 & 1. S_k is scaling parameter which is calculated for each attribute. S_k is calculated as

$$S_k = | u_k - l_k | \quad (4)$$

u_k and l_k are the upper bound and lower bound of the attribute k respectively. r_{ij} is the distance between the fireflies i and j which calculated from:

$$r_{ij} = \sqrt{\sum_{1 \le i \le n} (x_i - y_i)^2} \quad (5)$$

The value of attractiveness in optimization problems is calculated using an objective function. The algorithm for standard firefly algorithm is given below:

ALGORITHM II

Input: Objective function f(x) and algorithm parameters α_0, β_0 and γ
Output: Minimized function value position x_{min}
Step 1: Initialize firefly population p randomly.
Step 2: Initialize algorithm parameters α_0, β_0 and γ.
Step 3: Calculate fitness value using the objective function f(x) for each firefly.
Step 4: while (t <maxgeneration)
for i=1:p
for j=1:i
if (f(x_j) < f(x_i))
move firefly i towards j using (3)
calculate fitness value again of all fireflies
end if
end for
end for
end while
Step 5: Rank the fireflies to find the current best firefly.

4.0 KNN WITH FIREFLY ALGORITHM

As discussed before, there are many variants of firefly algorithm available in antecedent studies. Self-adaptive step algorithm is one of them. A comparison of the performances of self-adaptive step firefly algorithm and the standard firefly algorithm is demonstrated in [32]. The obtained results demonstrate that self-adaptive firefly algorithm is better than standard firefly algorithm in every aspect. The self-adaptive step firefly algorithm and standard firefly algorithm differs at randomization parameter α. In standard firefly algorithm the parameter α is either fixed all time or decreases exponentially. But in case of self-adaptive step firefly algorithm α is calculated for each firefly according to the fitness values that it has attained previously. The notion behind is that a firefly which is far from the global best solution should take larger steps and the firefly which is near to the global best solution should take smaller steps so that it can converge slowly to give best results.

$$h_i(t) = 1/\sqrt{(fit(t-1) - fit(t-2))^2 + 1} \quad (6)$$

$$\alpha(t+1) = 1 - 1/\sqrt{(fit_{best}(t) - fit(t))^2 + (h_i(t))^2 + 1} \quad (7)$$

Here, fit(t-1) = fitness of the firefly in $(t-1)^{th}$ generation.
fit(t-2) = fitness of the firefly in $(t-2)^{th}$ generation.
fit_{best} = fitness value of the best firefly in $(t-1)^{th}$ generation.
fit(t) = fitness of the firefly in t^{th} generation.

This conglomerate of KNN and self-adaptive step firefly algorithm works as follows: The foremost task is to reckon the representative of each distinct class in the data set using self-adaptive step firefly algorithm. After accomplishing this, the process of classifying any tuple becomes very easy. These obtained representatives would now be acting as new training data tuples. And when a job of classifying any unseen tuple is assigned we have to just calculate its distance from these new training data tuples only. The unknown tuple is categorized in that class, the representative of which has the least distance with that unknown tuple. The pseudo code for the proposed algorithm is given below:

ALGORITHM III

Step 1: Normalize the data set.
Step 2: Find representatives of each class in data set using Self-adaptive step firefly algorithm and in order to fulfill this, follow the subsequent steps.

a) Initialize algorithm parameters , and γ and input objective function f(x).
b) Divide the training data set according to the class attribute.
c) for each training data set grouped via class attribute. Let n be the number of fireflies in set.
while (t <maxgeneration)
for i=1:n
for j=1:i
if (f(x_j) < f(x_i))
move firefly i towards firefly j using equation (3). Calculate α using formulas in (6) and (7).
end if
end for
end for
end while
d) Find the current best firefly and choose it as representative of that class.
e) for each test tuple
calculate the distance of the test tuple from each of class representatives. Assign
that class to the test tuple from whose representative it has the least distance.

There are many advantages of this reoriented KNN algorithm. The first one is you don't need to pass the input parameter k

anymore as we have to do in standard KNN. Ascertaining the appropriate value of the parameter k is itself a challenging task. The second benefit is, it would reduce the cost complexity of KNN algorithm. This optimized KNN would sustain for longer period because once you have computed the representatives of each distinct class then the task of classifying a tuple would take only a fraction of seconds. On the other hand in case of KNN algorithm the cost complexity was high because for classification of a tuple, you have to compute its distance with all the training tuples every time.

5.0 EXPERIMENT AND RESULTS

The performance of the proposed modified KNN and the standard KNN is tested on six data sets of different sizes. The data sets are picked from UCI repository and Keel. Their performances are summarized in form of tables TABLE I and TABLE II. The TABLE I depicts the algorithmic parameters taken for self-adaptive step firefly algorithm, the number of generations for which the firefly algorithm is run. The proposed algorithm and the standard KNN algorithm both are implemented in MATLAB software. Hold out method has been used to split up the data set into training and test sets. 30% of the data set is upheld as test data set and rest of the data set is used to train the model. Three parameters are used to compare performances of both the algorithms which are accuracy, time taken for classifying all the test tuples and kappa statistics. The performances of both the algorithms in aspects of accuracy and time are shown with help of graphs. Fig. 1 depicts that the proposed algorithm gives accuracy comparable to that of standard KNN in case of large data sets. From Fig. 2 it can be seen that the proposed algorithm takes much less time in classifying the test tuples when compared to standard KNN algorithm and the difference enlarges when the size of data set is large.

6.0 CONCLUSION

In present research paper, a modified KNN algorithm is proposed which has used self-adaptive step firefly algorithm to find representatives of distinct classes in data set. This study demonstrates that the proposed algorithm optimize the results by taking much less time in comparison to standard KNN. Due to which the cost of computation also got reduced. On scrutinizing the results obtained, it can be concluded that the proposed algorithm performs well in case of large data sets.

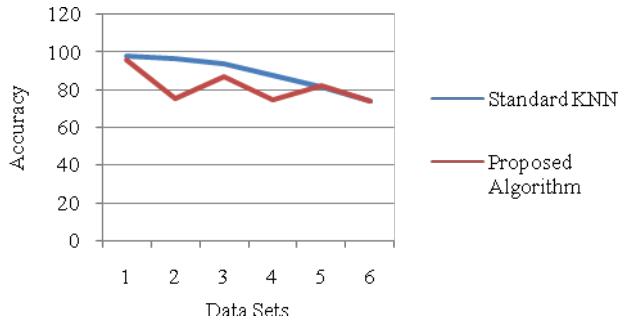

Figure 1: Graph depicting accuracies attained by both the algorithms

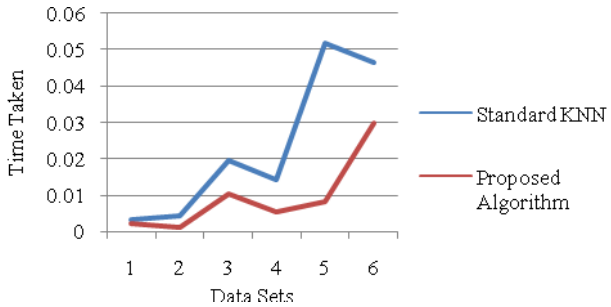

Figure 2: Graph depicting time taken by both the algorithms to classify test tuples

Table 1: Experiment Results for the Proposed Algorithm

Data sets	No. of instances in data set	Number of classes	Algorithmic parameters				Accuracy(%)	Time taken to classify test tuples(sec)	Time saved In comparison to Standard KNN	Kappa Statistics
			α_0	β_0	γ	Number of generations				
Iris	150	3	0.9	1	0.1	50	95.55	0.002110	39.57%	0.93
Wine	178	3	0.9	1	0.1	60	75.00	0.001160	74.15%	0.63
Wholesale	440	2	0.9	1	0.1	50	87.02	0.010322	47.34%	0.72
Data user modeling	404	4	0.9	1	0.1	70	74.79	0.005516	61.58%	0.66
Australian	690	2	0.9	1	0.1	50	82.04	0.008165	84.21%	0.63
pima	768	2	0.9	1	0.1	50	73.91	0.029670	35.87%	0.43

Table 2: Experiment Results For The Standard Algorithm

Data sets	No. of instances in data set	k	Number of classes	Accuracy(%)	Time taken to classify test tuples(sec)	Kappa Statistics
Iris	150	5	3	97.78	0.003492	0.97
Wine	178	5	3	96.15	0.004487	0.94
Wholesale	440	5	3	93.89	0.019602	0.86
Data user modeling	404	5	4	87.39	0.014356	0.83
Australian	690	5	2	81.55	0.051713	0.63
pima	768	5	2	73.91	0.046269	0.42

REFERENCES

[1]. "Data Mining Applications," ZenTut. .

[2]. S. Bagga and G. N. Singh, "Applications of Data Mining," Int. J. Sci. Emerg. Technol. Latest Trends, vol. 1, no. 1, pp. 19–23, 2012.

[3]. N. Padhy, P. Mishra, and R. Panigrahi, "The Survey of Data Mining Applications And Feature Scope," Int J ComputSciTechnol, vol. 2, no. 3, pp. 43–58, Jun. 2012.

[4]. B. M. Ramageri, "Data Mining Techniques and Applications," Int J ComputSciEng, vol. 1, no. 4, pp. 301–305.

[5]. R. Kumar and R. Verma, "Classification algorithms for data mining: A survey," Int. J. Innov. Eng. Technol. IJIET, vol. 1, no. 2, pp. 7–14, 2012.

[6]. S. S. Nikam, "A Comparative Study of Classification Techniques in Data Mining Algorithms," Orient. J. Comput. Sci. Technol., vol. 8, no. 1, pp. 13–19, Apr. 2015.

[7]. "A Detailed Introduction to K-Nearest Neighbor (KNN) Algorithm," God, Your Book Is Great !!, 18-May-2010. .

[8]. T. Larose and C. D. Larose, Data Mining and Predictive Analytics, 2nd ed. Wiley, 2015.

[9]. Lamba and D. Kumar, "Survey on KNN and Its Variants," IJARCCE, vol. 5, no. 5, pp. 430–435, May 2016.

[10]. Wettschereck and D. Thomas G., "Locally adaptive nearest neighbor algorithms," Adv. Neural Inf. Process. Syst., pp. 184–184, 1994.

[11] H. E.H.S, E.-H. Sam, G. Karypis, and V. Kumar, "Text categorization using weight adjusted k-nearest neighbor classification," in Text categorization using weight adjusted k-nearest neighbor classification, Springer Berlin Heidelberg, 2001, pp. 53–65.

[12] B. Li, S. Yu, and Q. Lu, "An Improved k-Nearest Neighbor Algorithm for Text Categorization," in Proceedings of the 20th International Conference on Computer Processing of Oriental Languages, Shenyang, China, 2003.

[13] S. Ougiaroglou, A. Nanopoulos, A. N. Papadopoulos, Y. Manolopoulos, and T. Welzer-Druzovec, "Adaptive k-Nearest-Neighbor Classification Using a Dynamic Number of Nearest Neighbors," in Advances in Databases and Information Systems, Y. Ioannidis, B. Novikov, and B. Rachev, Eds. Springer Berlin Heidelberg, 2007, pp. 66–82.

[14] Z. Voulgaris and G. D. Magoulas, "Extensions of the K Nearest Neighbour Methods for Classification Problems," in Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications, Anaheim, CA, USA, 2008, pp. 23–28.

[15] Y. Cai, D. Ji, and D. Cai, "A KNN Research Paper Classification Method Based on Shared Nearest Neighbor," in Proceedings of the 8th NTCIR Workshop Meeting, 2008.

[16] P. WiraBuana, S. Jannet D.R.M., and I. KetutGedeDarma Putra, "Combination of K-Nearest Neighbor and K-Means based on Term Re-weighting for Classify Indonesian News," Int. J. Comput. Appl., vol. 50, no. 11, pp. 37–42, Jul. 2012.

[17] K. Q. Weinberger and L. K. Saul, "Distance Metric Learning for Large Margin Nearest Neighbor Classification," J. Mach. Learn. Res., vol. 10, pp. 207–244, Dec. 2009.

[18] Y. Song, J. Huang, D. Zhou, H. Zha, and C. L. Giles, "Iknn: Informative k-nearest neighbor pattern classification," in Knowledge Discovery in Databases: PKDD 2007, 2007, pp. 248–264.

[19] A. K. Saxena, "On the Importance of Ensembles of Classifiers," BIJIT, vol. 5, no. 1, pp. 569–576, Jun. 2013.

[20] S. Mitra, S. K. Pal, and P. Mitra, "Data mining in soft computing framework:a survey," IEEE Trans. Neural Netw., vol. 13, no. 1, pp. 3–14, 2002.

[21] K. Lal, N. .Mahanti, and P. Bihar, "Role of soft computing as a tool in data mining," pp. 526–537, 2011.

[22] Y. K. Mathur and A. Nand, "Soft Computing Techniques and its impact in Data Mining," Int. J. Emerg. Technol. Adv. Eng., vol. 4, no. 8, Aug. 2014.

[23] R. Kruse, C. Borgelt, D. D. Nauck, N. J. Van Eck, and M. Steinbrecher, "The Role of Soft Computing in Intelligent Data Analysis," in Proceedings of the 16th IEEE International Conference on Fuzzy Systems, 2007, pp. 9–17.

[24] T. Gupta and D. Kumar, "Performance Optimization of Benchmark Functions Using VTS-ABC Algorithm," BIJIT, vol. 6, no. 2, Dec. 2014.
[25] Y. Kumar and D. Kumar, "Parametric Analysis of Nature Inspired Optimization Techniques," Int J ComputAppl Found ComputSci, vol. 32, no. 3, pp. 42–49, Oct. 2011.
[26] R. S. Parpinelli, H. S. Lopes, and A. Freitas, "Data mining with an ant colony optimization algorithm," IEEE Trans. Evol. Comput., vol. 6, no. 4, pp. 321–332, Aug. 2002.
[27] P. Jaganathan, K. Thangavel, A. Pethalakshmi, and M. Karnan, "Classification rule discovery with ant colony optimization and improved Quick Reduct algorithm," IAENG Int. J. Comput. Sci., vol. 33, no. 1, pp. 50–55, Mar. 2007.
[28] D. Martins, M. D. Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens, "Classification with ant colony optimization," IEEE Trans. Evol. Comput., vol. 11, no. 5, pp. 651–665, Oct. 2007.
[29] Y. Lui, Z. Qin, Z. Shi, and J. Chen, "Rule discovery with particle swarm optimization," in Content Computing, 2004, pp. 291–296.
[30] C. R. Hema, M. P. Paulraj, S. Yaacob, A. H. Adom, and R. Nagarajan, "Particle swarm optimization neural network based classification of mental tasks," in 4th Kuala Lumpur International Conference on Biomedical Engineering 2008, 2008, pp. 883–888.
[31] S. Banerjee, A. Bharadwaj, D. Gupta, and V. K. Panchal, "Remote sensing image classification using artificial bee colony algorithm," Int J ComputSciInf, vol. 2, no. 3, pp. 67–72, 2012.
[32] M. B. Pouyan, R. Yousefi, S. Ostadabbas, and M. Nourani, "A Hybrid Fuzzy-Firefly Approach for Rule-Based Classification," in Proceedings of the Twenty-Seventh International Florida Artificial Intelligence Research Society Conference, 2014, pp. 357–362.
[33] S. S. Jamsandekar and R. R. Mudholkar, "Fuzzy Classification System by Self Generated Membership Function Using Clustering Technique," BIJIT, vol. 6, no. 1, pp. 697–704, Jun. 2014.
[34] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy K-nearest neighbor algorithm," IEEE Trans. Syst. Man Cybern., vol. SMC-15, no. 4, pp. 580–585, Jul. 1985.
[35] R. Jensen and Cornelis, Chris, "Fuzzy-rough nearsest neighbor classification," in Transactions on rough sets XIII, Springer Berlin Heidelberg, 2011, pp. 56–72.
[36] N. Verma, N. Verma, and A. B. Patki, "Rough Set Techniques for 24 Hour Knowledge Factory," BIJIT, vol. 4, no. 1, pp. 421–426, Jun. 2012.
[37] S. V. Chande and M. Sinha, "Genetic Algorithm: A Versatile Optimization Tool," BIJIT, vol. 1, no. 1, pp. 7–12, Jun. 2009.
[38] N. Suguna and K. Thanushkodi, "An Improved k-Nearest Neighbor Classification Using Genetic Algorithm," Int J ComputSci Issues, vol. 7, no. 2, pp. 18–21, Jul. 2010.
[39] M. A. Jabbar, B. Deekshatulu, and P. Chandra, "Classification of heart disease using K-nearest neighbor and genetic algorithm," Procedia Technol., vol. 10, pp. 85–94, Dec. 2013.
[40] Shrivastava, Shailendra Kumar and P. Mewada, "ACO Based Feature Subset Selection for Multiple K-Nearest Neighbor Classifiers," Int. J. Comput. Sci. Eng., vol. 3, no. 5, pp. 1831–1838, May 2011.
[41] I. Babaoğlu, O. Findik, E. Ulker, and N. Aygul, "A novel hybrid classification method with particle swarm optimization and k-nearest neighbor algorithm for diagnosis of coronary artery disease using exercise stress test data," Int. J. Innov. Comput. Inf. Control, vol. 8, no. 5, 2012.
[42] M. CHEN, J. GUO, C. WANG, and Fenlin WU, "PSO-based Adaptively Normalized Weighted KNN Classifier," J. Comput. Inf. Syst., vol. 11, pp. 1407–1415, Apr. 2015.
[43] H. Yigit, "ABC-based distance-weighted kNN algorithm," J. Exp. Theor. Artif. Intell., vol. 27, no. 2, pp. 189–198, Mar. 2015.
[44] İ. Babaoğlu, "Diagnosis of Coronary Artery Disease Using Artificial Bee Colony and K-Nearest Neighbor Algorithms," Int. J. Comput. Commun. Eng., pp. 56–59, 2013.
[45] A. Hashmi, S. Goel, N. Goel, and D. Gupta, "Firefly Algorithm for Unconstrained optimization," IOSR J. Comput. Eng. IOSR-JCE, vol. 11, no. 1, pp. 75–78, Jun. 2013.
[46] F. Iztok, X.-S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," Swarm Evol. Comput., vol. 13, pp. 34–46, Dec. 2013.
[47] X.-S. Yang and X. He, "Firefly algorithm: recent advances and applications," Int J Swarm Intell, vol. 1, no. 1, pp. 36–50, Jan. 2013.
[48] N. Ali, M. A. Othman, M. N. Husain, and M. H. Misran, "A Review of Firefly Algorithm," ARPN J EngApplSci, vol. 9, no. 10, pp. 1732–1736, Oct. 2014.
[49] X.-S. Yang, "Firefly Algorithm, Lévy Flights and Global Optimization," in Research and development in intelligent systems XXVI, 2010, pp. 209–218.
[50] S. M. Farahani, A. A. Abshouri, B. Nasiri, and M. R. Meybodi, "A Gaussian Firefly Algorithm," Int. J. Mach. Learn. Comput., vol. 1, no. 5, pp. 448–453, Dec. 2011.
[51] S. Yu and S. Yang, "Self-Adaptive Step Firefly Algorithm," J. Appl. Math., Nov. 2013.
[52] A. Ritthipakdee, A. Thammano, and N. Premasathian, "An Improved Firefly Algorithm for Optimization Problems," in The 5th International Symposium on Advanced Control of Industrial Processes (ADCONIP 2014), 2014.