Computer
& Robotics

A Novel Classification Method: A Hybrid Approach Based on Large Margin Nearest Neighbor Classifier

Alieh Ashoorzadeh[a], Abbas Toloie Eshlaghy[b,*], Mohammad Ali Afshar Kazemi[c]

[a]*Department of Information Technology Management, Science and Research Branch, Islamic Azad University, Tehran, Iran.*
[b]*Department of Industrial Management, Science and Research Branch, Islamic Azad University, Tehran, Iran.*
[c]*Department of Industrial Management, Central Tehran Branch, Islamic Azad University, Tehran, Iran*

Received 22 July 2023; Accepted 18 August 2023

Abstract

Classification is the operation of dividing various data into multiple classes where they share quantitative and qualitative similarities. Classification has many use cases in engineering fields such as cloud computing, power distribution, and remote sensing. The accuracy of many classification techniques such as *k*-nearest neighbor (*k*-NN) is highly dependent on the method used in the calculation of distances between samples. It is assumed that samples close to each other belong to the same class while samples that belong to different classes have a large distance between them. One of the popular distance calculation methods is the Mahalanobis distance. Many methods, including large margin nearest neighbor (LMNN), have been proposed to improve the performance of *k*-NN in recent years. Our proposed method aims to introduce a cost function to calculate data similarities while solving the local optimum pitfall of LMNN and optimizing the cost function determining distances between instances. Although *k*-NN is an efficient classification technique that is simple to comprehend and use, it is costly to compute for large datasets and sensitive to outlier data. Another difficult feature of *k*-NN is that it can only measure distance in Euclidean space. The distance metric should ideally be modified to fit the specific needs of the application. Due to the disadvantages in *k*-NN and LMNN methods, to optimize the objective function to calculate distances for the test data and to improve classification accuracy, we initially use the genetic algorithm to reduce the range of the solution space and then by using the gradient descent the optimal values of parameters in the cost function is obtained. Our method is carried out on different benchmark datasets with varying numbers of attributes and the results are compared to *k*-NN and LMNN methods. Misclassification rate, precision, f1 score, and kappa score are calculated for different values of *k*, mutation rate, and crossover rate. Overall, our proposed method shows superior performance with an average accuracy rate of 87.81% which is the highest among all methods. The average precision, f1 score, and kappa score of our method are 0.8453, 0.8513, and 0.6976 respectively.

Keywords: classification; large margin nearest neighbor; genetic algorithm; optimization.

1.Introduction

Research to develop improved optimization algorithms has been prompted by the intricacy of many real-world optimization situations. To find the best answers to complex optimization problems that cannot be addressed using conventional techniques, an effective meta-heuristic search is needed. Meta-heuristic global optimization algorithms are now a more popular and useful option for resolving complex and ill-defined issues that would otherwise be challenging to resolve using conventional techniques. This is because of their nature, which entails that the search space is discontinuous, that objective functions are not differentiated, and that the first possible solutions are not differentiated [1].

Effectively evaluating the similarity or distance between data is crucial for machine learning algorithms, pattern recognition [2], segmentation [3], and data mining [4]. The distance algorithm used to calculate the relationship between the input data affects the accuracy of various methods, including *k*-NN [5]. Regardless of the data collection, general methods like Euclidean distance, cosine distance, etc. are used to calculate the degree of similarity or distance for both vectors. This demonstrates the importance of distance metric learning, whose goal is to determine the most appropriate similarity or distance approach based on the data. After applying metric learning, the data that are conceptually similar to each other will be closer to each other, and

[*]Corresponding Author. Email: toloie@srbiau.ac.ir

the data that are not conceptually similar will move away from each other. One of the most important distance metric learning is Mahalanobis distance learning [6]. Many methods, including large margin nearest neighbor, have been proposed to improve the performance of *k*-NN in recent years.

One of the metric learning methods is the learning method with a large margin nearest neighbor (LMNN) [7]. The LMNN metric learning method increases the efficiency of the *k*-NN classification. In this learning method, *k* neighbors of each data have the same label as that data and dissimilar data at different classes are separated by a large margin compared to the data at the same class. In the LMNN method, for the training data x_i, the number of *k* nearest neighbors with the same label is considered as the target data. To perform a successful classification, *k*-NN requires that there is no similarly labeled data among the *k* nearest neighbors of the data x_i. Therefore, in the LMNN method, an area is considered for the data x_i, which contains the target data along with a safety margin. With this definition, the data that are placed in this safety margin are considered imposter data and pose a problem to classification. Shrinking these zones creates a linear transformation of the input space, which increases the number of samples whose *k* nearest neighbors are similarly labeled. Euclidean distance in the transformed space is the Mahalanobis distance in the original space. With this learning method, the distance calculation method is optimized.

Gradient descent methods are generally considered for local search improvements. However, classical methods cannot escape from getting trapped in locally optimal solutions due to the initial conditions of the non-convex problems [8]. Unlike gradient methods, where an initial condition is critical for convergence to an optimal solution, meta-heuristics methods do not necessarily require a good initial guess. In the last few decades, genetic algorithms (GA) have been shown to be an effective approach to solving real-world optimization problems [9]. However, it is clear that in the presence of huge solution space and many local optima, GA cannot guarantee finding the global optimum.

Motivated by its issues, several researchers have demonstrated that *k*-NN classification can be greatly improved by learning an appropriate distance metric from labeled samples.

Pawlovsky and Matsuhashi [10] applied genetic algorithm (GA) for component selection to improve the accuracy of the *k*-NN method for breast cancer prognosis. The GA uses the best chromosome (member) to generate a new generation. The crossover and mutation rates do not have fixed values, instead, they depend on the evaluation of the chromosomes involved in the production of new chromosomes. GA determines the best attributes that should be used in the prognosis with the *k*-NN method. The percentage of operators in this method is dynamic. The crossover percentage depends on the evaluation of the best member and the member to be replaced in the new generation. The mutation rate depends on the evaluation of the current member to be replaced and the average evaluation of all members of the current generation.

Sarkate and Deorankar [11] proposed a hybrid *k*-NN and genetic algorithm to classify drug data. A hybrid method that can solve the problem of classification and optimization and can have a more favorable result in the drug data set. Genetic search has been used to prune redundant and irrelevant features.

Poorheravi et al. [12] suggested that solving semi-definite programming (SDP) problems requires the interior point method, which is repetitive and slow, especially for large data. This can be improved by selecting more important data for embedding. For this purpose, triple mining technique is used, which combines the original data. They make a triple anchor with positive and negative values. In this method, in addition to proposing triple extraction techniques, a hierarchical approach has been proposed for optimization. Accelerated metric learning involves the iterative selection of data subsets with hierarchical stratified sampling to train the embedded subspace. The triplets are selected by stratified sampling in hierarchical hyperspheres. Not only does this approach speed up the SDP optimization approach by reducing time, but it also improves performance in some cases.

Xu [13] proposed a distance metric based on a new deep learning method, called Deep Large Margin Nearest Neighbor (DLMNN), which is an improved LMNN. A detailed learning framework and training algorithm for DLMNN is presented. Their method trains a convolutional neural network to represent features on a metric subspace, where the intra-class samples are as close as possible, while the inter-class samples are separated by a large margin. The evaluation is carried out in terms of different scenarios of walking data.

Zhao and Zhou [14] used LMNN to quickly push samples toward the center of the class to obtain a large class margin instead of between classes and further improve the classification performance. Along with linear discriminant analysis (LDA), a new linear feature extraction method called LDA-LMNN is proposed. This method can overcome the limitations of LDA. In addition, random elimination is used to learn the linear transformation matrix to improve the ability to scale.

He and Xu [15] proposed a method in which depth features are included in the learning process by learning the distance metric. Specifically, two distance metrics are simultaneously learned in two feature spaces by averaging the new cost function. During the test phase, the distance between a pair of RGB images is measured using a distance metric that is learned from visual images. In their approach, the LMNN algorithm is developed to take advantage of the specific information provided in the form of LMNN+. The slack variable that exists in the LMNN constraints is replaced by a slack function that is defined in the corresponding depth feature space, so the distance between samples using the visual feature is corrected with the corresponding inter-sample distance using the depth feature where the intra-class difference is minimized while the inter-class difference is maximized. The distances in the depth feature space can be used to guide the training process in the visual feature space.

In summary, although *k*-NN is an effective classification method that is easy to understand and implement, it is however sensitive to outlier data and computationally expensive for large datasets. Being limited to Euclidean distance is another challenging aspect of *k*-NN. The distance metric should ideally be adjusted to the particular application at hand. Consider the scenario where age and gender are determined for images of faces using *k*-NN. Even though distances are calculated between the same sets of extracted features (such as pixels and color histograms) for both tasks, using the same distance metric for age and gender classification cannot be considered optimal.

1.1.Motivation

The effectiveness of classification is fundamentally dependent on how distances are calculated between various examples due to the decision rule's very nature. The majority of *k*-NN implementations compute straightforward Euclidean distances when no prior knowledge is present (assuming the examples are represented as vector inputs). Euclidean distances unfortunately disregard any statistical regularities that could be inferred from a sizable training set of labeled examples. These problems have led a number of researchers to show that by learning a suitable distance metric from labeled examples, classification can be greatly enhanced. This is the problem of distance learning metrics. Recent research has demonstrated that even a linear transformation of the input features can significantly enhance classification. Our work is built by a novel way on the effectiveness of these earlier strategies. Introducing an approach with no dependency on the number of classes is another motivation for our work.

1.2.Contribution

Our proposed method aims to introduce a cost function to calculate data similarities while solving the local optimum pitfall of LMNN and optimizing the cost function determining distances between instances. Due to the shortcomings raised in *k*-NN and LMNN methods, to optimize the objective function in our method to obtain the distance for the test data and more accurate classification, we initially use the genetic algorithm to reduce the range of the solution space and then by using the gradient descent the optimal value of the parameter in the cost function is obtained.

The main contributions of this paper are as follows:

- The general framework for distance learning method with the mathematical formulation.
- Introducing an all-encompassing cost function to calculate distances between different data points.
- Utilizing a metaheuristic algorithm (i.e., genetic algorithm) to limit the solution space boundaries.
- Finding optimal values of parameters of cost function using gradient descent.
- Evaluating the exploration and exploitation aspects of our method

The rest of this paper is structured as follows: materials and the proposed method are discussed in Section 2. Simulation results are shown and discussed in Section 3. Finally, the conclusion is presented in Section 4.

2.Materials and Methods

In this section, we present the proposed algorithm of classification by first discussing k-NN and distance learning methods, and later in the section genetic algorithm and gradient descent are presented to improve those algorithms. Finally, our algorithm is explained in full detail.

2.1.k-Nearest Neighbor Classification

The k-NN classification algorithm is known as one of the widely used classification techniques. The k-NN algorithm uses one of the simplest classification ideas [16]. This technique is one of the oldest methods for general and non-parametric classification and is based on supervised learning. The goal of this method is to find the nearest *k* data available from the training data. First, the distance of the new instance with the set of training instances is calculated. Then, by considering *k* members from the nearest neighbors of the new instance, the corresponding class of this instance is predicted. In general, the performance of the k-NN algorithm depends on three factors: the sample size, the selection of the distance metric and, the value of *k*. The selection of the appropriate distance metric strongly affects the accuracy of the algorithm. Measuring the distance between two data points is one of the main requirements for the k-NN algorithm. In distance metric learning, the goal is to obtain the distance function (similarity) from the data so that the logically similar data are close to each other, and the data that are not logically similar to each other move away from each other. Many learning algorithms need a metric to determine the distance or similarity between objects. Many distance metrics have been proposed to calculate the distance of objects, such as Euclidean distance, Manhattan distance, and Cosine distance. However, these general metrics are not suitable for many applications and using the training data can obtain a better metric [17]. This has led to the emergence of metric learning methods. We want to find the distance function using training data. Training data that are similar in terms of meaning and concept to each other. Dissimilar data are separated from each other. Metric learning methods are usually used as pre-processing for machine learning and pattern recognition algorithms, such as classification by k-NN or *k*-means clustering.

2.2.Distance Learning Method and LMNN

In the past few decades, researchers have proposed various methods to obtain distance metric learning [18]. Choosing a suitable metric greatly improves the classification accuracy compared to the Euclidian distance which makes no distinction between different features of the data. Methods for distance metric learning can be divided into linear methods and non-linear methods. Naturally, linear methods can’t be suitable for classification data where there is a non-linear correlation in features.

Assume a set of $X = \{x_i\} \complement R^n$ of data points, the general Mahalanobis distance is as follows:

$$D(\vec{x}_i, \vec{x}_j) = (\vec{x}_i - \vec{x}_j)^T \mathrm{M} (\vec{x}_i - \vec{x}_j) \tag{1}$$

where M can be any positive matrix that is found by optimization.

LMNN metric learning method is presented to learn Mahalanobis distance in the k-NN classifier [19]. In this learning method, *k* neighbors of each data have the same label as that data and dissimilar data are separated by a large margin. In the LMNN method, for the training data x_i, a number of *k* nearest neighbors with the same label is considered as the target data. To perform a successful k-NN classification, it is necessary that among the *k* nearest neighbors of the data x_i, all labels are not the same. Therefore, in the LMNN method, a zone is considered for the data x_i, which contains the target data along with a safety margin. With this definition, the similar data placed in this zone are difficult to classify and considered imposter data. In the LMNN learning method, the number of imposter data should be minimized after learning.

2.3.Genetic Algorithm

Genetic algorithms are evolutionary algorithms that search and find optimal solutions. It looks for the best solution in a multi-dimensional search space [20]. Instead of finding one solution, genetic algorithms produce a set that includes different solutions. By evaluating multiple points, the likelihood of reaching a suitable solution is increased. Each one is a vector on the multi-dimensional space vector. Genetic algorithms simulate an evolutionary continuum in a computer environment to solve problems. They do not develop only one structure for a solution like some other optimization methods but develop a set that is formed with these structures.

Every element in an individual is called a gene. Individuals in the population are determined by genetic algorithm processes on an evolutionary continuum. The process of a genetic algorithm usually begins with a randomly selected population of chromosomes. These chromosomes are representations of the solutions to the problem. During the evaluation, two basic operators, crossover and mutation, are used to simulate the natural reproduction and mutation of species. The selection of chromosomes for survival and combination is biased towards the fittest chromosomes.

Two fundamental techniques used by genetic algorithms to efficiently search the solution space are exploration and exploitation. Exploitation refers to the search of the immediate vicinity of a promising region, whereas exploration refers to the search of the unexplored area of the feasible region. The degree to which these two search behaviors are balanced is crucial to the effectiveness of these algorithms. Exploitation in genetic algorithm is done through the selection process, while exploration is done by crossover and mutation. These two requirements are contradictory, and a good search algorithm must find a compromise between the two. A purely random search is good for exploration but not for exploitation. Combinations of these two strategies can be very effective.

Members of the population will converge to some point in the solution space even in the absence of any selection pressure. This is caused by the accumulation of stochastic errors. If a gene becomes prevalent in the population by chance, it is just as likely to become more prevalent in the next generation as it is to become less prevalent. If a gene's dominance increases over several generations and the population is finite, a gene can spread to all members of the population. Once a gene has converged in this manner, crossover can no longer introduce new gene values. This causes a ratchet effect, with each gene eventually becoming fixed as generations pass. As a result, the rate of genetic drift provides a lower bound on the rate at which a GA can converge to the correct solution. That is, if the GA is to use gradient information in the fitness function, the fitness function must have a large enough slope to compensate for any genetic drift. By increasing the mutation rate, the rate of genetic drift can be reduced. If the mutation rate is too high, the search becomes effectively random, and the gradient information in the fitness function is once again ignored.

2.4.Gradient Descent

Gradient descent is a general algorithm that is usually used to find the optimal solution in an unconstrained multivariate differentiable function. Gradient descent is not only used in linear regression but can be used in all machine learning topics. In general, this algorithm is applicable to infinite parameters [21].

We know that if we start from a point in the function, the fastest way to reach the optimal point is to move along the path with the greatest slope. The gradient of the function, which the partial derivatives of the function with respect to the variables Θ_1, Θ_2, …, Θ_n indicates the greatest slope. Therefore, the formulation of the problem is as follows:

We have a cost function $J(\theta_0, \theta_1, \dots, \theta_n)$ and we want to minimize Θ_0, Θ_1, …,Θ_n. The algorithm starts with initial Θ_0, Θ_1, …,Θ_n. The value of Θ_1, Θ_2, …, Θ_n is changed towards better results. The change of Θ_0, Θ_1, …,Θ_n is proportional to the partial derivatives of the cost function $J(\theta_0, \theta_1, \dots, \theta_n)$.

Changing the value of Θ_1, Θ_2, …, Θ_n continues until $J(\theta_0, \theta_1, \dots, \theta_n)\left\{\theta_i := \theta_i - \propto * \frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1, \dots, \theta_n)\right\}$ reaches the lowest point possible. The final solution may be the local optimum point instead of the global optimum point.

In each iteration of the algorithm, the values of Θ_0, Θ_1, …,Θ_n are updated simultaneously according to the partial derivatives of the cost function with respect to the parameters.

α is called the learning rate and controls the size of steps that the algorithm takes in each of its iterations. Usually, its value is between 0 and 1. If α is chosen very small, the convergence happens later because the gradient descent moves towards the minimum point with smaller steps. If α is chosen large, the value of $J(\Theta)$ may not decrease with each iteration or it may not reach convergence. Often learning rate is set at 0.1.

2.5.Proposed Method Design

According to the discussed topics, the purpose of the our method is to design an algorithm to find quantitative and qualitative similarities between samples. The basis of classification algorithm is based on the *k*-NN algorithm and uses a distance learning method similar to LMNN. In this method, the distance of the test data to the *k* nearest neighbors is determined by using the Mahalanobis distance. Then, for the training data x_i, the number of *k* nearest neighbors is determined. The neighbor with the same label is considered the target data. To perform a successful classification, there must be no dissimilar labeled data among the *k* closest neighbors of the data x_i. Therefore, an area is considered for the data x_i. It is possible that it contains the target data along with a margin of safety. With this definition, similar data that are placed in this area which cause problems for classification are considered imposter data.

Assuming a set of points $x_1, x_2, x_3,\ldots,x_n$ which labels are y_i (i=1,2,…,n). The goal is to learn a linear transformation L leading to the following transformed distance [22]:

$$D(\vec{x}_i,\vec{x}_j) = \left\|\mathrm{L}(\vec{x}_i-\vec{x}_j)\right\|^2 = (\vec{x}_i-\vec{x}_j)^T \mathrm{L}^T\mathrm{L}(\vec{x}_i-\vec{x}_j) \quad (2)$$

where L is a $d\times d$ matrix and d is the dimension of the input vector. For each input x_i, k target neighbors are selected which are k inputs with the same label as x_i. The target neighbors can be identified as the k nearest neighbor with the same label as x_i.

Our objective function for the distance metric has two terms. The first term intends to reduce the distance of any given data with its own neighbors while the second term intends to increase the distances of each data with all other data that are not in the same class.

These two terms have competitive effects because the first part is reduced by reducing the distance between samples, while the second part is reduced by increasing them. The first part of the cost function penalizes the large distance between each input and its target neighbors. In terms of a linear transformation of the input space, the sum of squares of this distance is as follows:

$$\varepsilon_1(\mathrm{L}) = \textstyle\sum_{j\to i}\left\|\mathrm{L}\left|\vec{x}_i-\vec{x}_j\right|\right\|^2 \quad (3)$$

where L is a $d\times d$ matrix and d is the dimension of the input vector, and x_i and x_j are inputs.

This expression creates a pulling force that attracts the neighbors of the target in the linear transformation of the input space. The above expression only penalizes the large distance between the inputs and their target neighbors and not the large distance between all the data with the same label. The second case is intentionally penalized. Our approach thus differs from many previous distance metric approaches only by penalizing large distances between neighbors.

The second part of the cost function penalizes the short distance between data with different labels:

$$\varepsilon_2(\mathrm{L}) = \sum_{i,j\to i}\sum_{l}(1-y_{il})\left[1+\left\|\mathrm{L}(\vec{x}_i-\vec{x}_j)\right\|^2 - \|\mathrm{L}(\vec{x}_i-\vec{x}_l)\|^2\right]_+ \quad (4)$$

where $[z]_+ = \max(0,z)$ is the hinge loss. L is a $d\times d$ matrix, and x_i and x_j are inputs. If y_i=y_l then y_{il}=1, otherwise y_{il}=0;

The goal is to optimize the cost function to find better neighbors and determine the data class. Our cost function becomes [22]:

$$\begin{aligned} cost(\mathrm{L},X) &= (1-\mu)\varepsilon_1(\mathrm{L}) + \mu\varepsilon_2(\mathrm{L}) \\ &= (1-\mu)\sum_{j\to i}\left\|\mathrm{L}\left|\vec{x}_i-\vec{x}_j\right|\right\|^2 \\ &\quad + \mu\sum_{i,j\to i}\sum_{l}(1-y_{il})\Big[1 + \left\|\mathrm{L}(\vec{x}_i-\vec{x}_j)\right\|^2 - \|\mathrm{L}(\vec{x}_i-\vec{x}_l)\|^2\Big]_+ \end{aligned} \quad (5)$$

where the positive constant μ changes the importance of those two terms.

LMNN uses semi-definite programming (SDP) to transform the distance metric learning problem into a convex problem [23]. The SDP is:

$$\text{Minimize} \quad \textstyle\sum_{ij}\eta_{ij}\,(\vec{x}_i-\vec{x}_j)^T \mathrm{M}(\vec{x}_i-\vec{x}_j) + c\sum_{ij}\eta_{ij}\,(1-\tilde{y}_{il})\varepsilon_{ijl} \quad (6)$$

where M is the semi-definite matrix of Mahalanobis metric, c is the control variable and $\varepsilon_{ijl}$is the slack variable for hinge loss.

With conditions

$$(\vec{x}_i-\vec{x}_l)^T\mathrm{M}(\vec{x}_i-\vec{x}_l) - (\vec{x}_i-\vec{x}_j)^T\mathrm{M}(\vec{x}_i-\vec{x}_j) \geq 1-\varepsilon_{ijl}$$

$$\varepsilon_{ijl} \geq 0$$

$M \succeq 0$

The cost function in equation (5) in terms of L is not convex. To minimize this function, the gradient descent approach is used in elements of L. However, such an approach is prone to getting trapped in local minima. The results of this form of gradient descent generally depend on initial estimates for L. Therefore, they may not be used in many problems and programs. To overcome this issue and optimize the objective function for more accurate classification, first we use genetic algorithm to find the global optimal range of the objective function and then by using the gradient descent method, it is precisely determined by obtaining the optimal point of the parameter L in the cost function.The suggestion to increase the efficiency of the global optimization methods is to combine the local optimization methods with the global one, which has the advantage of increasing the speed along with dodging local optima traps. After L is obtained, the distance between the test data and the neighboring points is calculated to determine the similarity, and it is calculated based on the type of neighbors of the data set.Our method can be summarized as: first, we choose the appropriate *k* and divide the dataset into training and test data. The cost function is defined using equation (5). Then using genetic algorithm, L_g is optimized. The first step of genetic algorithm is to find an initial population and the rate of mutation and crossover. Until the end criterion is met, parents are chosen, crossover and mutation are carried out and a new generation of offspring is created and their fitness value is calculated. These steps are repeated until the semi-optimum L_g is found. The optimum L is obtained by gradient descent. Initially, learning rate α is defined and L_0=L_g is set. Steps of gradient descent are repeated until optimum L is found. Then the distance between the test and all training data is calculated using equation (2). Afterwards, these distances are sorted in ascending order and *k* nearest neighbors are selected. In the end, class labels are assigned to each instance in test data based on the classes of those *k* nearest neighbors. The flowchart of the entire process is shown in Fig. 1.

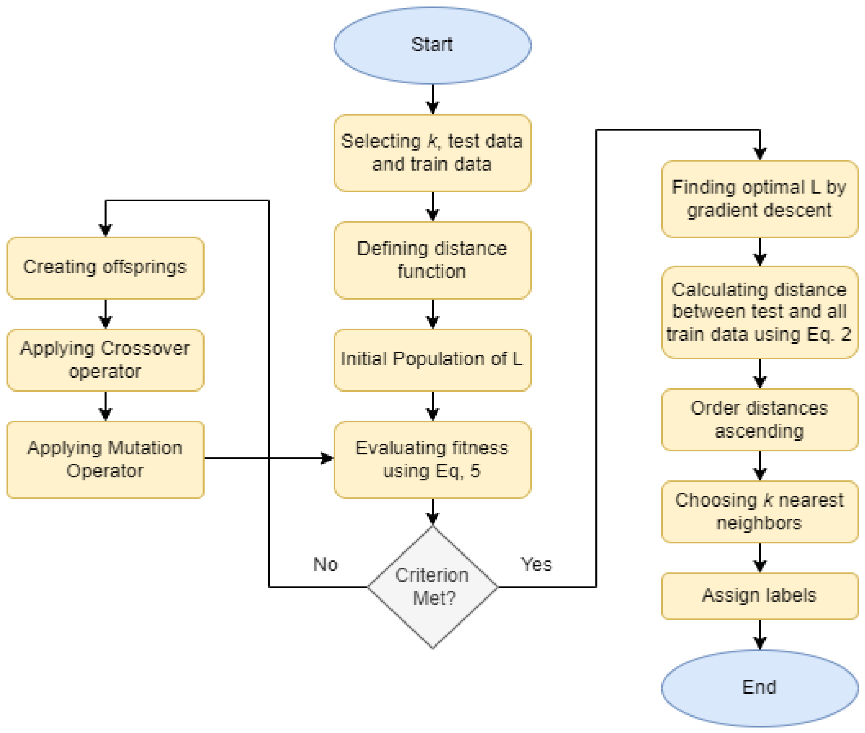
Start
Selecting k, test data and train data
Defining distance function
Initial Population of L
Evaluating fitness using Eq, 5
Criterion Met?
No
Yes
Creating offsprings
Applying Crossover operator
Applying Mutation Operator
Finding optimal L by gradient descent
Calculating distance between test and all train data using Eq. 2
Order distances ascending
Choosing k nearest neighbors
Assign labels
End

Fig. 1. Flowchart of the proposed algorithm

3.Simulation Results

To analyze our method, simulations are conducted using MATLAB. To evaluate the exploration and exploitation aspects of our method, 23 benchmark problems presented in Table 1 are used. These benchmark tests have different types of complexity and modality. *Dim* is the dimension of functions; *Range* is the input boundaries of each function and *min* is the global minimum. To evaluate the effectiveness of our method, it is compared to Particle Swarm Optimization (PSO) [24], Gravitational Search Algorithm (GSA) [25], and Whale Optimization Algorithm (WOA) [26]. The objective is to minimize these functions.

The exploitation ability of meta-heuristic algorithms is evaluated by F1-F7 unimodal functions since they only have one global minimum. To evaluate the exploration capability of algorithms, multimodal functions F8-F23 are selected which unlike unimodal functions include multiple local minima. The results are reported in Table 2. These results show that our algorithm gives the best answer for F1, F3, F4, F8, F10, F14, F16, and F17-F22 while giving the second-best answer in some cases.

For the genetic algorithm, the population is set to 30 chromosomes and the number of iterations is set to 250. The learning rate of gradient descent α is set to 0.1. To evaluate the proposed method, misclassification rate, precision, F1 score, and kappa score are calculated and those rates are compared to k-NN and LMNN methods. Different values for μ from equation (5), k (number of nearest neighbors), mutation rate, and crossover rate are examined. μ has a range between 40 and 70. crossover rate is between 35 and 65, mutation rate is set between 5 and 15 and k is between 3 and 7.

The benchmark UCI datasets [27] are as follows:

Balance data set: This data set has been extracted from psychological experimental results which have 625 instances. The instances are classified into three classes: tip to the right, tip to the left, or balanced. The 4 attributes are the left weight, the left distance, the right weight, and the right distance. From these instances, 438 are used for training and 187 for testing.

Cancer-Int data set: This data set consists of two classes of benign and malignant cancer diagnoses which originates from the Wisconsin Breast Cancer data set. This dataset contains 699 instances and 10 attributes.

Diabetes data set: This data set is used for the diagnosis of diabetes disease based on glucose measurement of patients throughout the day. This data set has 768 elements. The first 538 elements are used as the training set and the remainder 230 as the test set.

E. coli data set: This data set classifies E. coli bacteria into different localization sites. This data set has 336 members each with 8 attributes that are classified into 8 classes from which we picked 235 members for training and 101 members for testing.

Horse Colic data set: This data set is used to predict whether a horse died, survived, or was euthanized. The data set has 368 instances and 27 attributes.

Iris data set: This data set is the perhaps most famous data set in the UCI repository and is used for the classification of Iris flowers into one of the three classes of Iris Setosa, Iris Versicolor, and Iris Virginica. The data set has 150 elements and 4 attributes.

Wine data set: This data set is used for the classification of wines into 3 different classes based on their chemical composition. This data set has 178 elements. And each instance has 13 attributes.

The misclassification rates are the average of 50 independent runs where 70% of the instances are used for training and the remaining 30% are for testing. Tables 3-6. show the misclassification rates for different datasets with varying values for μ, crossover rate, and mutation rate while keeping k=3.

Table 1
Benchmark tests for evaluation of exploration and exploitation

Function	*Dim*	*Range*	*min*
$f_1(x)=\sum_{i=1}^{n} x_i^2$	30	[-100,100]	0
$f_2(x)=\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10,10]	0
$f_3(x)=\sum_{i=1}^{n}(\sum_{j-1}^{i} x_j)^2$	30	[-100,100]	0
$f_4(x)=max_i\{\lvert x_i\rvert, 1\le i \le n\}$	30	[-100,100]	0
$f_5(x)=\sum_{i=n}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	30	[-30,30]	0
$f_6(x)=\sum_{i=1}^{n}([x_i+0.5])^2$	30	[-100,100]	0
$f_7(x)=\sum_{i=1}^{n} i x_i^4 + random[0,1)$	30	[-1.28,1.28]	0
$f_8(x)=\sum_{i=1}^{n} -x_i sin(\sqrt{\lvert x_i \rvert})$	30	[-500,500]	-418.9829×5
$f_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$	30	[-5.12,5.12]	0
$f_{10}(x)=-20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})-exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i))+20+e$	30	[-32,32]	0
$f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})+1$	30	[-600,600]	0
$f_{12}(x)=\frac{\pi}{n}\{10\sin(\pi y_i)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n} u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4}$ $u(x_i,a,k,m)=\begin{cases} k(x_i-a)^m & x_i>a \\ 0 & -a<x_i<a \\ k(-x_i-a)^m & x_i<-a \end{cases}$	30	[-50,50]	0
$f_{13}(x)=0.1\{sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+sin^2(3\pi x_i+1)]$ $+(x_n-1)^2[1+sin^2(2\pi x_n)]\}+\sum_{i=1}^{n} u(x_i,5,100,4)$	30	[-50,50]	0
$f_{14}(x)=(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6})^{-1}$	2	[-65,65]	1
$f_{15}(x)=\sum_{i=1}^{11}[a_i-\frac{x_i(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}]^2$	4	[-5,5]	0.00030
$f_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	[-5,5]	-1.0316

$f_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_i + 10$	2	[-5,5]	0.398
$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ $\times[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[-2,2]	3
$f_{19}(x) = -\sum_{i=1}^{4} c_i exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij}{}^2)$	3	[1,3]	-3.86
$f_{20}(x) = -\sum_{i=1}^{4} c_i exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij}{}^2)$	6	[0,1]	-3.32
$f_{21}(x) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.1532
$f_{22}(x) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.4028
$f_{23}(x) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.5363

Table 2

Results of benchmark tests F_1 to F_{23}

Benchmark Test	Proposed Method	PSO	GSA	WOA
F1	8.2439E-32	0.00013	2.53E-16	1.41E-30
F2	0.0039	0.04214	0.05565	1.06E-21
F3	4.5873E-08	70.1256	896.534	5.39E-07
F4	0.0674	1.08648	7.35487	0.072581
F5	28.8943	96.7183	67.5430	27.86558
F6	2.9658	0.00010	2.5E-16	3.116266
F7	0.0192	0.12285	0.089441	0.001425
F8	-5214.24	-4841.29	-2821.07	-5080.76
F9	28.3712	46.70423	25.96841	0
F10	6.7749E-8	0.276015	0.062087	7.4043
F11	0.08237	0.009215	27.70154	0.000289
F12	0.54963	0.006917	1.799617	0.339676
F13	4.87629	0.006675	8.899084	1.889015
F14	2.01947	3.627168	5.859838	2.111973
F15	0.00134	0.000577	0.003673	0.000572
F16	-1.03163	-1.03163	-1.03163	-1.03163
F17	0.398	0.397887	0.397887	0.397914
F18	3	3	3	3
F19	-3.5657	-3.86278	-3.86278	-3.85616
F20	-3.2663	-3.26634	-3.31778	-2.98105
F21	-7.7028	-6.8651	-5.95512	-7.04918
F22	-10.1484	-8.45653	-9.68447	-8.18178
F23	-10.1749	-9.95291	-10.5363	-9.34238

Table 3
Misclassification rates for Balance and Cancer dataset

				Balance			Cancer	
μ	Crossover	Mutation	Proposed	LMNN	*k*-NN	Proposed	LMNN	*k*-NN
70	35	5	0.1712	0.2041	0.2112	0.0700	0.0832	0.0870
70	35	10	0.1866	0.2227	0.2280	0.0735	0.0876	0.0897
70	35	15	0.2031	0.2319	0.2376	0.0778	0.0881	0.0918
70	50	5	0.1914	0.2173	0.2214	0.0757	0.0857	0.0887
70	50	10	0.1964	0.2224	0.2344	0.0770	0.0878	0.0905
70	50	15	0.2122	0.2302	0.2388	0.0712	0.0769	0.0787
70	65	5	0.1726	0.2160	0.2271	0.0722	0.0910	0.0937
70	65	10	0.1789	0.2241	0.2322	0.0669	0.0836	0.0876
70	65	15	0.1931	0.2309	0.2409	0.0724	0.0857	0.0891
50	35	5	0.2287	0.2264	0.2365	0.0921	0.0918	0.0954
50	35	10	0.2247	0.2223	0.2261	0.0852	0.0852	0.0883
50	35	15	0.2359	0.2214	0.2323	0.0884	0.0830	0.0889
50	50	5	0.2433	0.2313	0.2433	0.0870	0.0828	0.0873
50	50	10	0.2423	0.2284	0.2360	0.0890	0.0845	0.0861
50	50	15	0.2385	0.2157	0.2237	0.0923	0.0836	0.0874
50	65	5	0.2131	0.2219	0.2333	0.0867	0.0905	0.0962
50	65	10	0.2253	0.2353	0.2488	0.0812	0.0847	0.0872
50	65	15	0.2395	0.2378	0.2507	0.0872	0.0867	0.0903
40	35	5	0.2065	0.2232	0.2328	0.0750	0.0809	0.0837
40	35	10	0.2298	0.2479	0.2622	0.0838	0.0900	0.0931
40	35	15	0.2116	0.2177	0.2301	0.0812	0.0836	0.0864
40	50	5	0.2243	0.2334	0.2441	0.0793	0.0819	0.0853
40	50	10	0.2153	0.2214	0.2340	0.0786	0.0817	0.0848
40	50	15	0.2298	0.2261	0.2330	0.0830	0.0825	0.0863
40	65	5	0.1931	0.2198	0.2300	0.0762	0.0864	0.0889
40	65	10	0.1888	0.2152	0.2236	0.0745	0.0847	0.0898
40	65	15	0.2165	0.2348	0.2369	0.0836	0.0907	0.0949

For the Balance dataset, best results are obtained when μ, crossover rate, and mutation rate are 70%, 35%, and 5% or 70%, 65%, and 5% respectively. However, in the Cancer dataset when μ, crossover rate, and mutation rate are set to 70%, 65%, and 10% the best results are obtained for the proposed method, while the best results for LMNN and *k*-NN are given when μ, crossover rate, and mutation rate are set to 70%, 50%, and 15%.

Table 4
Misclassification rates for Diabetes and Ecoli Dataset

			Diabetes			Ecoli		
μ	Crossover	Mutation	Proposed	LMNN	*k*-NN	Proposed	LMNN	*k*-NN
70	35	5	0.2232	0.2658	0.2774	0.1459	0.1745	0.1820
70	35	10	0.2103	0.2499	0.2644	0.1514	0.1784	0.1853
70	35	15	0.2015	0.2288	0.2331	0.1694	0.1914	0.2000
70	50	5	0.2146	0.2423	0.2529	0.1735	0.1965	0.2047
70	50	10	0.2088	0.2376	0.2449	0.1741	0.1978	0.2078
70	50	15	0.2328	0.2544	0.2631	0.1684	0.1829	0.1896
70	65	5	0.1944	0.2435	0.2497	0.1588	0.1984	0.2100
70	65	10	0.1840	0.2299	0.2395	0.1473	0.1837	0.1902
70	65	15	0.1958	0.2335	0.2427	0.1733	0.2055	0.2161
50	35	5	0.2371	0.2362	0.2421	0.1910	0.1902	0.2008
50	35	10	0.2358	0.2354	0.2446	0.1933	0.1906	0.2041
50	35	15	0.2554	0.2419	0.2499	0.1945	0.1829	0.1896
50	50	5	0.2479	0.2340	0.2463	0.1951	0.1845	0.1927
50	50	10	0.2471	0.2329	0.2423	0.1924	0.1818	0.1965
50	50	15	0.2653	0.2387	0.2537	0.2098	0.1884	0.1963
50	65	5	0.2308	0.2400	0.2486	0.1935	0.2029	0.2098
50	65	10	0.2372	0.2478	0.2533	0.1876	0.1957	0.2037
50	65	15	0.2210	0.2196	0.2278	0.1884	0.1869	0.1949
40	35	5	0.2117	0.2295	0.2396	0.1708	0.1859	0.1929
40	35	10	0.2224	0.2409	0.2510	0.1749	0.1902	0.2008
40	35	15	0.2262	0.2341	0.2433	0.1822	0.1884	0.1955
40	50	5	0.2169	0.2240	0.2304	0.1833	0.1898	0.1959
40	50	10	0.2341	0.2422	0.2524	0.1833	0.1886	0.1986
40	50	15	0.2410	0.2362	0.2454	0.1982	0.1953	0.2057
40	65	5	0.2106	0.2399	0.2494	0.1612	0.1833	0.1894
40	65	10	0.2077	0.2349	0.2441	0.1578	0.1794	0.1865
40	65	15	0.2344	0.2537	0.2571	0.1806	0.1963	0.2039

When μ, crossover rate, and mutation rate are set to 70%, 65%, and 10% or 70%, 65%, and 5%, the proposed method performs best for Diabetes dataset. LMNN performs better when μ, crossover rate, and mutation rate are set to 50%, 65%, and 15%. The same value for *k*-NN also results in a relatively low misclassification rate.

For the Ecoli dataset, the lowest classification rates for all methods are obtained when μ, crossover rate, and mutation rate are set to 70%, 35%, and 5%.

Table 5
Misclassification rates for Horse and Iris dataset

			Horse			Iris		
μ	Crossover	Mutation	Proposed	LMNN	*k*-NN	Proposed	LMNN	*k*-NN
70	35	5	0.3272	0.3917	0.4117	0.0338	0.0431	0.0453
70	35	10	0.2960	0.3516	0.3655	0.0369	0.0453	0.0484
70	35	15	0.3228	0.3659	0.3699	0.0396	0.0449	0.0458
70	50	5	0.3240	0.3699	0.3739	0.0382	0.0427	0.0431
70	50	10	0.3336	0.3793	0.4028	0.0360	0.0440	0.0440
70	50	15	0.3163	0.3435	0.3558	0.0391	0.0440	0.0462
70	65	5	0.2870	0.3578	0.3697	0.0364	0.0484	0.0489
70	65	10	0.2976	0.3723	0.3906	0.0360	0.0453	0.0480
70	65	15	0.3044	0.3629	0.3787	0.0364	0.0418	0.0449
50	35	5	0.3607	0.3574	0.3721	0.0467	0.0462	0.0484
50	35	10	0.3769	0.3732	0.3844	0.0418	0.0418	0.0444
50	35	15	0.3811	0.3606	0.3763	0.0493	0.0440	0.0444
50	50	5	0.3872	0.3661	0.3809	0.0444	0.0431	0.0453
50	50	10	0.3776	0.3565	0.3752	0.0480	0.0458	0.0471
50	50	15	0.3857	0.3484	0.3699	0.0476	0.0449	0.0444
50	65	5	0.3407	0.3545	0.3629	0.0391	0.0431	0.0436
50	65	10	0.3683	0.3828	0.3987	0.0427	0.0436	0.0453
50	65	15	0.3684	0.3672	0.3787	0.0409	0.0404	0.0409
40	35	5	0.3123	0.3369	0.3497	0.0400	0.0418	0.0444
40	35	10	0.3472	0.3769	0.3958	0.0404	0.0436	0.0453
40	35	15	0.3363	0.3450	0.3550	0.0413	0.0427	0.0427
40	50	5	0.3534	0.3672	0.3840	0.0413	0.0427	0.0444
40	50	10	0.3442	0.3547	0.3692	0.0458	0.0467	0.0489
40	50	15	0.3837	0.3778	0.3923	0.0458	0.0436	0.0458
40	65	5	0.3345	0.3809	0.4006	0.0356	0.0440	0.0444
40	65	10	0.3261	0.3708	0.3879	0.0404	0.0453	0.0480
40	65	15	0.3283	0.3556	0.3815	0.0382	0.0409	0.0431

For the proposed method when μ, crossover rate, and mutation rate are set to 70%, 65%, and 5% misclassification rate is the lowest for the Horse dataset. However, for *k*-NN and LMNN methods best results are obtained when μ, crossover rate, and mutation rate are set to 40%, 35%, and 5%

For the Iris data set the best result for the proposed method is obtained when μ, crossover rate, and mutation rate are set to 70%, 35%, and 5% and the second-best result is obtained when μ, crossover rate, and mutation rate are set to 70%, 65%, and 10%. LMNN and *k*-NN also perform relatively well with those two sets of parameters.

Table 6
Misclassification rates for Wine dataset

			Wine		
μ	Crossover	Mutation	Proposed	LMNN	*k*-NN
70	35	5	0.0517	0.0634	0.0626
70	35	10	0.0525	0.0630	0.0638
70	35	15	0.0540	0.0623	0.0619
70	50	5	0.0540	0.0615	0.0642
70	50	10	0.0528	0.0592	0.0604
70	50	15	0.0543	0.0596	0.0615
70	65	5	0.0494	0.0619	0.0634
70	65	10	0.0479	0.0619	0.0630
70	65	15	0.0562	0.0653	0.0679
50	35	5	0.0592	0.0589	0.0596
50	35	10	0.0592	0.0592	0.0600
50	35	15	0.0615	0.0577	0.0589
50	50	5	0.0660	0.0638	0.0660
50	50	10	0.0675	0.0642	0.0675
50	50	15	0.0702	0.0630	0.0657
50	65	5	0.0600	0.0615	0.0634
50	65	10	0.0626	0.0664	0.0664
50	65	15	0.0615	0.0611	0.0608
40	35	5	0.0581	0.0626	0.0660
40	35	10	0.0577	0.0611	0.0657
40	35	15	0.0570	0.0600	0.0608
40	50	5	0.0581	0.0600	0.0608
40	50	10	0.0581	0.0608	0.0619
40	50	15	0.0630	0.0623	0.0645
40	65	5	0.0562	0.0649	0.0664
40	65	10	0.0517	0.0577	0.0585
40	65	15	0.0562	0.0600	0.0642

For the Wine data set the best result for the proposed method is obtained when μ, crossover rate, and mutation rate are set to 70%, 65%, and 10% and the second best result is obtained when μ, crossover rate, and mutation rate are set to 70%, 65%, and 5%. The best result for LMNN is obtained when μ, crossover rate, and mutation rate are set to 40%, 65%, and 10% and the second best result is obtained when μ, crossover rate, and mutation rate are set to 50%, 35%, and 15%.

These results illustrated in Tables 3-6 show that overall, the best setting for μ, crossover, and mutation parameters are 70%, 65%, and 10%. Our simulations were carried out on rates above and below these ranges, however, these changes lead to noticeable degradation in performance among all methods. Now the results for various values for *k* are calculated while keeping the best setting for other parameters. Figs. 2-6 show that in all cases when the value *k* is 3, all three algorithms show their lowest misclassification rate. Increasing the number of neighbors has a negative effect on performance.

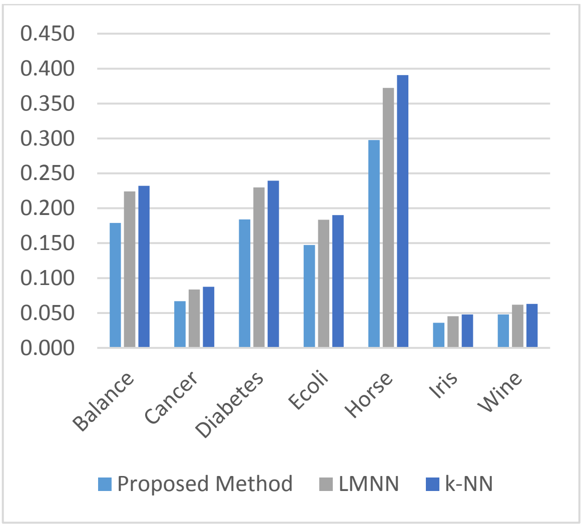

Fig. 2. Misclassification rate for *k*=3

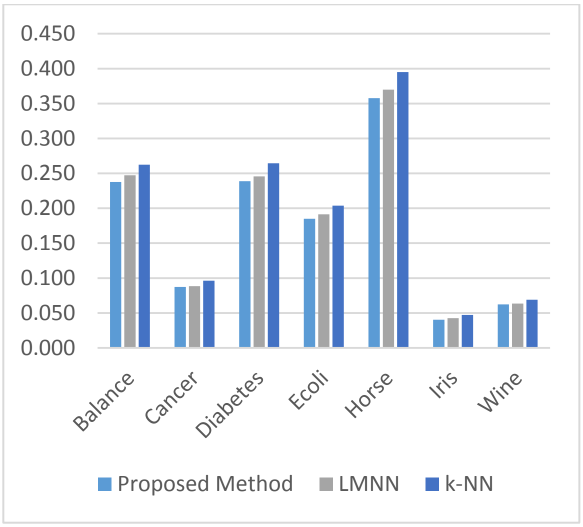

Fig. 3. Misclassification rate for *k*=4

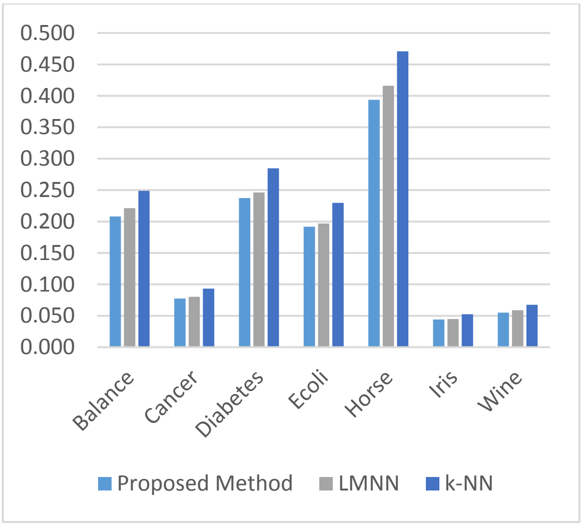

Fig. 4. Misclassification rate for *k*=5

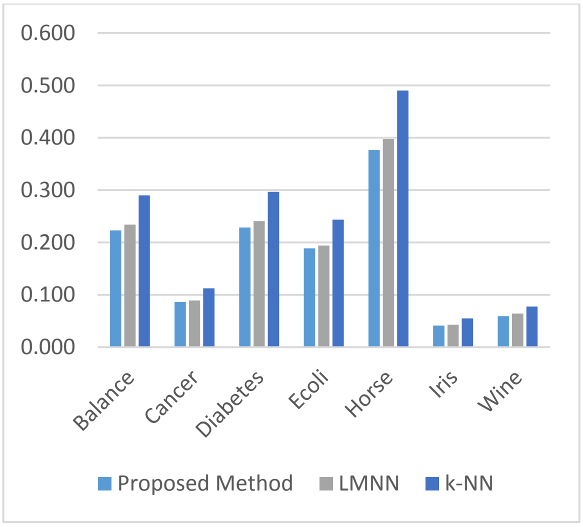

Fig. 5. Misclassification rate for *k*=6

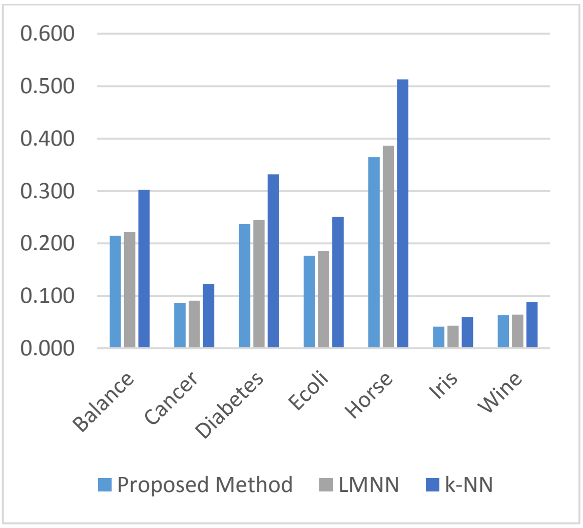

Fig. 6. Misclassification rate for *k*=7

To further analyze and compare different methods, we use a variety of validation metrics. The accuracy is the number of true results (both true positives (TP) and true negatives (TN)) divided by the total number of tests, in other words, accuracy is 1-(misclassification rate). Methods with the lowest misclassification rate will have the highest accuracy. Precision is the ratio of true positives to the sum of true and false positives. Sensitivity also known as recall is the number of true positives divided by both true positives and false positives. A measure that combines precision and sensitivity is the harmonic mean of precision and sensitivity or the F1 score. Equations (7-10) are the formulas to calculate these metrics [28].

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

$$\text{precision} = \frac{TP}{TP + FP} \tag{8}$$

$$\text{sensitivity} = \frac{TP}{TP + FN} \tag{9}$$

$$\text{F1 score} = 2 * \frac{precisiion * sensivity}{precision + sensitivity} \tag{10}$$

The kappa score compares an observed accuracy to an expected accuracy and is calculated as:

$$\text{Kappa score} = \frac{P_0 - P_e}{1 - P_e} \tag{11}$$

where P_0 is the overall accuracy of the model and P_e is the measure of the agreement between the predictions and the actual class values. To calculate precision, F1 score, and kappa score, we have set μ, crossover, and mutation parameters at 70%, 65%, and 10% and kept k at 3. The precision of the proposed method is compared to LMNN and k-NN algorithms in Table 7. Table 8 shows the F1 score among all methods for different datasets. Methods with kappa scores higher than 80% are considered to be great classifiers. Table 9 Illustrates the kappa score for all three methods for different datasets using the most favorable parameters obtained so far. In all cases, our proposed method outperforms LMNN and k-NN. Overall, the results for the Horse dataset are worse than other datasets which can be assumed due to the high number of attributes in this dataset.

Table 7
Comparison of Precision among all methods

Dataset	Proposed Method	LMNN	*k*-NN
Balance	0.7479	0.7027	0.6950
Cancer	0.9610	0.9504	0.9538
Diabetes	0.6975	0.6501	0.6341
Ecoli	0.7990	0.7088	0.7615
Horse	0.7820	0.7190	0.6945
Iris	0.9641	0.9569	0.9544
Wine	0.9656	0.9353	0.9430

Table 8
Comparison of F1 Score among all methods

Dataset	Proposed Method	LMNN	*k*-NN
Balance	0.7684	0.7202	0.7121
Cancer	0.9465	0.9319	0.9337
Diabetes	0.7544	0.7068	0.6915
Ecoli	0.8268	0.7303	0.7419
Horse	0.7375	0.6677	0.6454
Iris	0.9622	0.9541	0.9518
Wine	0.9639	0.9318	0.9399

Table 9
Comparison of kappa score among all methods

Dataset	Proposed Method	LMNN	*k*-NN
Balance	0.5969	0.4987	0.4795
Cancer	0.8494	0.8096	0.8155
Diabetes	0.6062	0.5280	0.5024
Ecoli	0.6044	0.2088	0.2527
Horse	0.3934	0.2451	0.1970
Iris	0.9150	0.8970	0.8920
Wine	0.9183	0.8467	0.8650

The proposed algorithm has been compared with several well-known algorithms [29]. These algorithms include Firefly Algorithm (FA), Artificial Bee Colony algorithm (ABC), Particle Swarm Optimization (PSO), Bat Algorithm (BA), and Cuckoo Seach Algorithm (CSA). Classification accuracy percentage has been measured for our method and the other 5 algorithms and reported in Table 10. It can be easily verified that the proposed algorithm gives the best solution for most problems. Table 11 shows the ranking of the algorithms by computing their average classification accuracy percentages. Our proposed algorithm is the best according to the ranking which has been shown in Table 11.

Table 10
Classification accuracy rate of the algorithms on each problem

Dataset	FA	ABC	PSO	BA	CSA	Proposed Method
Balance	84.62	84.62	74.53	84.62	75.8	84.79
Cancer	97.19	97.19	94.2	97.19	94.2	96.1
Diabetes	77.61	77.61	74.48	77.61	73.13	77.75
Ecoli	86.59	86.59	84.64	86.59	85.37	84.87
Horse	64.29	61.74	59.02	67.03	61.54	78.2
Iris	100	100	97.37	100	97.37	96.42
Wine	100	100	97.78	100	97.78	96.56

Table 11
Average classification accuracy rate and ranking of the algorithms on all problems

Dataset	FA	ABC	PSO	BA	CSA	Proposed Method
Avg accuracy	87.18	86.82	83.14	87.57	83.59	87.81
Ranking	3	4	6	2	5	1

4.Conclusion

Improving the existing methods and proposing new methods of classification are of great interest. Traditional methods of classification tend to be highly dependent on the method used in the calculation of distances between samples. We introduced a cost function to calculate data similarities while solving the problem of premature convergence of LMNN and optimizing the cost function determining distances between data. We used the genetic algorithm to reduce the range of the solution space and then by using the gradient descent the optimal parameter in the cost function was obtained. We tested our method on 7 different benchmark datasets (Balance, Cancer, Diabetes, Ecoli, Horse, Iris, and Wine) each consisting of a different number of attributes, and the results were compared to *k*-NN and LMNN methods. Misclassification rates for a variety of values for *k*, mutation rate, and crossover rate were calculated and the results were compared to Fa, ABC, PSO, BA, and CSA. Overall, our proposed method showed superior performance with an average accuracy rate of 87.81% which was the highest among all methods. The average precision, f1 score, and kappa score of our method were 0.8453, 0.8513 and 0.6976 respectively.

For future work, we recommend exploring the idea of fine-tuning the value of *k* algorithmically instead of our method of trial and error. Small values of *k* make the algorithm susceptible to noisy data while reducing accuracy in some cases. However, simply choosing a large value for *k* increases the computational cost. Using larger data sets to further investigate our method efficiency can be useful too. A classification method applicable to large data sets can be suitable for image processing applications.

References

[1] Jahwar, A.F. and Abdulazeez, A.M., 2020. Meta-heuristic algorithms for K-means clustering: A review. PalArch's Journal of Archaeology of Egypt/Egyptology, 17(7), pp.12002-12020.

[2] Abiodun, O.I., Jantan, A., Omolara, A.E., Dada, K.V., Umar, A.M., Linus, O.U., Arshad, H., Kazaure, A.A., Gana, U. and Kiru, M.U., 2019. Comprehensive review of artificial neural network applications to pattern recognition. IEEE Access, 7, pp.158820-158846.

[3] Haq, E.U., Jianjun, H., Huarong, X., Li, K. and Weng, L., 2022. A hybrid approach based on deep cnn and machine learning classifiers for the tumor segmentation and classification in brain MRI. Computational and Mathematical Methods in Medicine, 2022.

[4] Gupta, M.K. and Chandra, P., 2020. A comprehensive survey of data mining. International Journal of Information Technology, 12(4), pp.1243-1257.

[5] Taunk, K., De, S., Verma, S. and Swetapadma, A., 2019, May. A brief review of nearest neighbor algorithm for learning and classification. In 2019 International Conference on Intelligent Computing and Control Systems (ICCS) (pp. 1255-1260). IEEE.

[6] Siddappa, N.G. and Kampalappa, T., 2020. Imbalance data classification using local mahalanobis distance learning based on nearest neighbor. SN Computer Science, 1(2), pp.1-9.

[7] Gong, X., Yang, J., Yuan, D. and Bao, W., 2021. Generalized Large Margin k-NN for Partial Label Learning. IEEE Transactions on Multimedia, 24, pp.1055-1066.

[8] Mei, J., Xiao, C., Szepesvari, C. and Schuurmans, D., 2020, November. On the global convergence rates of softmax policy gradient methods. In International Conference on Machine Learning (pp. 6820-6829). PMLR.

[9] Lambora, A., Gupta, K. and Chopra, K., 2019, February. Genetic algorithm-A literature review. In 2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon) (pp. 380-384). IEEE.

[10] Pawlovsky, A.P. and Matsuhashi, H., 2017, March. The use of a novel genetic algorithm in component selection for a kNN method for breast cancer prognosis. In 2017 Global Medical Engineering Physics Exchanges/Pan American Health Care Exchanges (GMEPE/PAHCE) (pp. 1-5). IEEE.

[11] Sarkate, P.A. and Deorankar, A.V., 2018. Classification of chemical medicine or drug using K nearest neighbor (KNN) and genetic algorithm. International Research Journal of Engineering and Technology, 5(3), pp.833-834.

[12] Poorheravi, P.A., Ghojogh, B., Gaudet, V., Karray, F. and Crowley, M., 2020. Acceleration of large margin metric learning for nearest neighbor classification using triplet mining and stratified sampling. arXiv preprint arXiv:2009.14244.

[13] Xu, W., 2021. Deep Large Margin Nearest Neighbor for Gait Recognition. Journal of Intelligent Systems, 30(1), pp.604-619.

[14] Zhao, G. and Zhou, Z., 2019. Efficient linear feature extraction based on large margin nearest neighbor. IEEE Access, 7, pp.78616-78624.

[15] He, J. and Xu, D., 2019. Large margin nearest neighbor classification with privileged information for biometric applications. IEEE Transactions on Circuits and Systems for Video Technology, 30(12), pp.4567-4577.

[16] Xing, W. and Bei, Y., 2019. Medical health big data classification based on KNN classification algorithm. IEEE Access, 8, pp.28808-28819.

[17] Wang, L., Liu, X., Yi, J., Jiang, Y. and Hsieh, C.J., 2020. Provably robust metric learning. Advances in Neural Information Processing Systems, 33, pp.19302-19313.

[18] Ali, N., Neagu, D. and Trundle, P., 2019. Evaluation of k-nearest neighbor classifier performance for heterogeneous data sets. SN Applied Sciences, 1(12), pp.1-15.

[19] Wang, D. and Tan, X., 2017. Robust distance metric learning via Bayesian inference. IEEE Transactions on Image Processing, 27(3), pp.1542-1553.

[20] Vaishali, R., Sasikala, R., Ramasubbareddy, S., Remya, S. and Nalluri, S., 2017, October. Genetic algorithm based feature selection and MOE Fuzzy classification algorithm on Pima Indians Diabetes dataset. In 2017 international conference on computing networking and informatics (ICCNI) (pp. 1-5). IEEE.

[21] Lin, T., Jin, C. and Jordan, M., 2020, November. On gradient descent ascent for nonconvex-concave minimax problems. In International Conference on Machine Learning (pp. 6083-6093). PMLR.

[22] Khan, M.M.R., Arif, R.B., Siddique, M.A.B. and Oishe, M.R., 2018, September. Study and observation of the variation of accuracies of KNN, SVM, LMNN, ENN algorithms on eleven different datasets from UCI machine learning repository. In 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT) (pp. 124-129). IEEE.

[23] Taheri, M., Moslehi, Z., Mirzaei, A. and Safayani, M., 2019. A self-adaptive local metric learning method for classification. Pattern Recognition, 96, p.106994.

[24] Kennedy, J. and Eberhart, R., 1995, November. Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks (Vol. 4, pp. 1942-1948). IEEE.

[25] Rashedi, E., Nezamabadi-Pour, H. and Saryazdi, S., 2009. GSA: a gravitational search algorithm. Information sciences, 179(13), pp.2232-2248.

[26] Mirjalili, S. and Lewis, A., 2016. The whale optimization algorithm. Advances in engineering software, 95, pp.51-67.

[27] Dua, D. and Graff, C., 2019. UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA (2019).

[28] Malmir, Z. and Rezvani, M.H., 2019. A novel ensemble approach for anomaly detection in wireless sensor networks using time-overlapped sliding windows. Journal of Computer & Robotics, 12(1), pp.1-13.

[29] Dhal, K. G., Das, A., Ray, S., & Das, S., 2019. A clustering based classification approach based on modified cuckoo search algorithm. Pattern Recognition and Image Analysis, 29, 344-359.