Journal of Computer Science
https://dergipark.org.tr/en/pub/bbd
Anatolian Science

ISSN,e-ISSN: 2548-1304
Volume:8, Issue:1, pp:36-49/2023
https://doi.org/10.53070/bbd.1259377

Research Paper

The Analysis of a Linear Classifier Developed through Particle Swarm Optimization

Fatih Aydın*[1]

[1]Department of Computer Engineering, Balıkesir University, Balıkesir, Türkiye

(fatih.aydin @balikesir.edu.tr)

Received:Mar.02,2023 Accepted:Apr.15,2023 Published:Jun.08,2023

Abstract— Meta-heuristics are high-level approaches developed to discover a heuristic that provides a reasonable solution to many varieties of optimization problems. The classification problems contain a sort of optimization problem. Simply, the objective herein is to reduce the number of misclassified instances. In this paper, the question of whether meta-heuristic methods can be used to construct linear models or not is answered. To this end, Particle Swarm Optimization (PSO) has been engaged to address linear classification problems. The Particle Swarm Classifier (PSC) with a certain objective function has been compared with Support Vector Machine (SVM), Perceptron Learning Rule (PLR), and Logistic Regression (LR) applied to fifteen data sets. The experimental results point out that PSC can compete with the other classifiers, and it turns out to be superior to other classifiers for some binary classification problems. Furthermore, the average classification accuracies of PSC, SVM, LR, and PLR are 80.8%, 80.6%, 77.4%, and 57.7%, respectively. In order to enhance the classification performance of PSC, more advanced objective functions can be developed. Further, the classification accuracy can be boosted more by constructing tighter constraints via another meta-heuristic. The source code of the proposed algorithm and the data sets are available at https://github.com/fatihaydin1/PSC for computational reproducibility.

Keywords: *Machine learning, Artificial intelligence, Particle swarm optimization, Meta-heuristic algorithms, Supervised learning.*

1. Introduction

In this section, we present general information regarding artificial intelligence, machine learning, supervised learning, linear classifiers, and meta-heuristics. Moreover, we mention the motivation of the work and the organization of the paper, as well.

1.1. Preliminary

Lots of definitions of intelligence are in question. One of these definitions is that "intelligence is the ability to quickly discover, in a search space, a satisfactory solution that appears a priori to observers" (Lenat & Feigenbaum, 1991). Accordingly, we can define Artificial Intelligence (AI) as a discipline, which aims to make automata attain intelligent behaviors by representing intelligence via computational models. There exist four approaches to AI: (i) the Turing test approach, (ii) the cognitive modeling approach, (iii) the “laws of thought” approach, and (iv) the rational agent approach. In terms of the Turing test approach, learning is thought of as acclimating to new situations and discovering and inferring patterns (Aziz & Memon, 2023; Russell & Norvig, 2010). In this respect, Machine Learning (ML) is a research domain, in which machines obtain the learning ability to uncover patterns in real-world data. One of the ML tasks is supervised learning, which is extensively used in many problems. Given a data set $D = \{(x_1, y_1), \dots, (x_m, y_m)\} \in \mathbb{R}^{m \times n}$, which includes m instances and is defined by n features and c classes, from input data $X = \{x_i\}_{i=1}^{m} \Rightarrow x_i = \left(x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(n)}\right) \in \mathbb{R}^n, i = 1, \dots, m$ and labels $y_j \in Y, j = 1, \dots, c$. A supervised learner searches for a function $h: X \to Y$ that is a member of the hypothesis space H (Aydin, 2023). When utilizing supervised learning works, batch learning is largely favored in the industry since it is quick and straightforward to apply as compared to online learning, which accepts a real-time stream of data.

The models used to separate classes are divided into two: linear and non-linear. In the ML area, the objective of classification is to employ the features of data points to identify which class they belong to. A linear classifier performs the classification task by making a classification decision in accordance with a linear combination of the

features. Linear classifiers obtain satisfactory results on linear data sets (Sivrikaya et al., 2021; Yuan et al., 2012). Perceptron Learning Rule (PLR) (Karakurt et al., 2022; Rosenblatt, 1958), non-kernelized Support Vector Machine (SVM) (Bumin & Ozcalici, 2023; Cortes & Vapnik, 1995), and Logistic Regression (LR) (Ay et al., 2023; Berkson, 1944) are some well-known linear classifiers. The common property of these linear classifiers is that the models constructed by them are a linear combination of features. In other words, such classifiers have inductive bias. The goal is to directly minimize the cost (i.e., error rate) of the classification process by building a linear classifier. A way to minimize the cost function is to utilize meta-heuristic algorithms. There are lots of works that Meta-heuristics have been used for solving classification problems (Athira Lekshmi et al., 2018; De Falco et al., 2007; Lawrence et al., 2021; M. G. Omran et al., 2002; M. G. H. Omran et al., 2006; Sousa et al., 2004; Telikani et al., 2022).

1.2. Meta-heuristics

The meta-heuristic approaches are high-level systems devised to discover or adjust a heuristic that can deliver an adequately satisfactory solution to an optimization problem, particularly with deficient or insufficient information or limited computing capacity. Compared to the other optimization algorithms, meta-heuristic methods do not guarantee that some problems always approach a global solution. A lot of meta-heuristic methods apply a sort of stochastic optimization. In combinatorial optimization, meta-heuristics investigating a broad set of possible solutions can frequently find better solutions with less computing than the other approaches. Hence, meta-heuristic methods are beneficial techniques for optimization problems. Meta-heuristic methods mostly rely on empirical results. However, there are some theoretical results on convergence and finding the global optimum, as well (Balamurugan et al., 2015; Kotary & Nanda, 2020). Meta-heuristic algorithms are categorized by a variety of their properties. In terms of search strategy, meta-heuristics are separated into two: local search and global search. In terms of search size, meta-heuristics are separated into two: single-solution and population-based meta-heuristics. Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995), which is the method used in this study, is known as a global search and population-based meta-heuristic algorithm (Ahmad et al., 2023).

In this paper, our motivation is to show that a linear model generated by PSO can usually reach a global solution and compete with the well-known linear classifiers. Moreover, we would like to show it can be successfully and efficiently applied to real-world problems as well as fulfilling the aforementioned qualities. Briefly, the main contribution of the offered approach is that it is straightforward and fast.

This manuscript is organized in that way: Section 2 provides a short overview of PSO. Section 3 then introduces the proposed approach. Section 4 explains the experimental procedure. Section 5 includes the results and discussion. Eventually, Section 6 explains the conclusions of the work.

2. PSO Revisited

PSO holds a stochastic optimization strategy with population-based and is developed by being socially inspired by swarm behavior among the population. PSO explores a global minimum $a \in \mathbb{R}^n$ (i.e., the solution with the proper fitness value) by iteratively modifying the velocity and position of each particle in the search space with respect to a certain objective function $f: \mathbb{R}^n \to \mathbb{R}$. PSO utilizes both the best position in the current neighborhood g and the particle's best position p at iteration t (Baygin et al., 2019; Khennak et al., 2023). There exist various extensions proposed for the algorithm (Asif et al., 2022; Mezura-Montes & Coello Coello, 2011; Pedersen, 2010). In accordance with these modifications, the updated values of each particle's velocity and position are calculated as seen in Equations (1) and (2).

$$v_i^{(t+1)} = \mathrm{W} v_i^{(t)} + y_1 u_1 \left(p_i - x_i^{(t)}\right) + y_2 u_2 \left(g_i - x_i^{(t)}\right) \tag{1}$$

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \tag{2}$$

where v denotes velocity. $x_i^{(t)}$ denotes the current position of the particle i at iteration t. W denotes a vector with the same sign values in growing order, delivering the lower and upper bound of the adaptive inertia. u_1 and u_2 are uniformly distributed random vectors. y_1 denotes the weighting of each particle's best position when updating velocity. y_2 denotes the weighting of the global best position when updating velocity.

With the initialization of PSO, the initial particles are created, and initial velocities are assigned to them. The cost function for each particle's position is then assessed, and the most proper fitness value and the best position are planned. The new velocities and positions are respectively and iteratively updated dependent on the current velocity, the particles' best positions, and the best positions of their neighbors. Subsequently, the iterations flow until the PSO arrives at a stopping criterion. In PSO, the particles are foremost positioned at random locations in the search space and advance in arbitrary directions. The direction of particles is steadily adjusted to shift in the

direction of the best-encountered positions of themself and their counterparts, probing their surroundings and potentially discovering better positions [3]. The topology of the society specifies the subset of individuals with which each particle can transfer information regarding the solution and many topologies to be used in PSO have been proposed (Almasi & Khooban, 2018; Bratton & Kennedy, 2007; Elshamy et al., 2007; Lai & Sato, 2021; Oliveira et al., 2016; Yin et al., 2011). Further, the choice of PSO input arguments possesses a considerable influence on optimization performance (Mason et al., 2018; Taherkhani & Safabakhsh, 2016; Yilmaz & Yolcu, 2022).

Even if PSO is partly straightforward to use and its convergence to global optima is high, it can undoubtedly fall into local optima and early convergence, as well (Hu et al., 2022). In the past decades, lots of enhanced PSOs have been offered to modify the searchability of PSOs and decrease the probability of falling into a local optimum (Chamkalani et al., 2014; Eshtay et al., 2021; Ma et al., 2019; Tan et al., 2019). The inertia weight W, which is an influential argument of PSO, can significantly increase the capacity of PSO. A right W can allow PSO to establish a proper trade-off between global and local probing. A larger W is generally more suitable for a global probing, and a smaller W is preferable for a local probing.

3. The Proposed Approach

PSO aims to minimize a given objective function $f: \mathbb{R}^n \to \mathbb{R}$. To this end, it is first necessary to define the objective function. Given a data set $D = \{(x_1, y_1), \dots, (x_m, y_m)\} \in \mathbb{R}^{m \times n}$ and a classifier with the function $h: X \to Y$ that is a member of the hypothesis space H, the objective function (i.e., cost function) is as seen in Equation 3. The hypothesis function is seen in Equation (4). The transfer function is seen in Equation (5).

$$f(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}_i(h_\theta(x_i) \neq y_i) \tag{3}$$

$$h_\theta = u(X\theta) \tag{4}$$

$$u(a) = \begin{cases} -1, & a \leq 0 \\ 1, & a > 0 \end{cases} \tag{5}$$

where $\theta = (\theta_0, \cdots, \theta_n)$ is the weight vector and θ_0 is bias, which is in the search space for unconstrained problems. θ^* be the global minimum such that $f(\theta^*) \leq f(\theta)$ if $\theta^*: f(\theta^*) = 0$. Accordingly, the best position in the particle's best position p at the next iteration $t + 1$ is computed as seen in Equation (6).

$$p_i = \begin{cases} p_i, & \mathrm{f}\left(\theta_i^{(t+1)}\right) > p_i \\ \left(\theta_i^{(t+1)}\right), & \mathrm{f}\left(\theta_i^{(t+1)}\right) \leq p_i \end{cases} \tag{6}$$

The best position in the current neighborhood g at the next iteration $t + 1$ is calculated as $g_i = \min(p_i)$. Concisely, p_i is the best position that the individual particle i has visited. On the other hand, g_i is the most proper position discovered by any particle in the whole swarm.

4. Experimental Procedure

In this part, we clarify the experimental procedure, containing benchmark data sets, the other linear classifiers (i.e., PLR, SVM, and LR), implementations, and evaluation metrics. We compare the algorithms used in the experiments to quantify their performance over 11 real-world data sets and 1 synthetic binary data set from UCI Machine Learning Repository[1] and OpenML[2]. The descriptive information about the data sets is seen in Table 1.

Table 1. The binary data sets that are employed in the empirical analysis.

#	Data set	Points	Features	Imbalance rate
1	Banknote Authentication[3]	1372	4	1.27
2	Blood Transfusion[4]	748	4	3.20
3	Breast Cancer[5]	699	9	1.90

[1] http://archive.ics.uci.edu/ml
[2] https://www.openml.org/
[3] https://archive.ics.uci.edu/ml/datasets/banknote+authentication
[4] https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center
[5] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29

4	Climate Model[6]	540	18	10.73
5	Connectionist Bench[7]	208	60	1.14
6	Diabetic Retinopathy[8]	1151	19	1.13
7	EEG Eye State[9]	14980	14	1.23
8	Haberman[10]	306	3	2.78
9	HTRU2[11]	17898	8	9.92
10	Ionosphere[12]	351	34	1.78
11	Madelon[13]	2000	500	1.00
12	Mozilla4[14]	15545	5	2.04
13	Parkinson Speech[15]	1040	26	1.00
14	QSAR Biodegradation[16]	1055	41	1.96
15	Vertebral Column[17]	310	6	2.10

Table 2 shows the linear classifiers employed in the experiments. The classifiers have been used by default parameter values. The parameter values of the Particle Swarm Classifier (PSC) are as seen in Table 2 unless otherwise stated.

Table 2. The linear classifiers used in the experiments.

#	Algorithm	Parameter (by default values)
1	Particle Swarm Classifier (PSC)	SwarmSize=10, ObjectiveLimit=0
2	Support Vector Machine (SVM)	Standardize=true, KernelFunction=linear
3	Logistic Regression (LR)	Model=nominal
4	Perceptron Learning Rule (PLR)	divideFcn=dividerand, divideMode=sample, trainRatio=90, valRatio=0, testRatio=10, epochs=100

We employ the classification accuracy rate (ACC) given by Equation (7) to evaluate the performance of the algorithms.

$$ACC = \frac{1}{m}\sum_{i=1}^{m}\delta(a_i, p_i) \tag{7}$$

$$\delta(x, y) = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases} \tag{8}$$

where a denotes the true class labels and p denotes the predictions.

We carry out all the experiments on a CPU at 1.6 GHz with 8 GB of RAM. In addition, we use MATLAB R2021a as a programming and numeric computing platform. All the experiments have been performed under tenfold cross-validation.

5. Results and Discussion

[6] https://archive.ics.uci.edu/ml/datasets/climate+model+simulation+crashes
[7] http://archive.ics.uci.edu/ml/datasets/connectionist+bench+(sonar,+mines+vs.+rocks)
[8] https://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set
[9] https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State
[10] https://archive.ics.uci.edu/ml/datasets/haberman%27s+survival
[11] https://archive.ics.uci.edu/ml/datasets/HTRU2
[12] https://archive.ics.uci.edu/ml/datasets/ionosphere
[13] https://archive.ics.uci.edu/ml/datasets/madelon
[14] https://www.openml.org/search?type=data&sort=runs&id=1046&status=active
[15] https://archive.ics.uci.edu/ml/datasets/Parkinson+Speech+Dataset+with++Multiple+Types+of+Sound+Recordings
[16] https://archive.ics.uci.edu/ml/datasets/QSAR+biodegradation
[17] http://archive.ics.uci.edu/ml/datasets/vertebral+column

In this section, we compare the algorithms with each other with respect to classification accuracy and running time. Besides, we analyze the performance of PSC by tuning its hyperparameters.

5.1. The Comparison of the Algorithms

First, we compare the algorithms with each other concerning classification accuracy and execution time. The comparisons of accuracy rates of the classifiers on the data sets are seen in Figure 1. All the algorithms are run by their default values. From a statistical viewpoint, the average accuracy rates of the methods point out comparative performance well. Accordingly, PSC, SVM, LR, and PLR correspond to 77.8%, 80.6%, 77.4%, and 57.7%, respectively. In respect of the number of the highest accuracy rate on the data sets, SVM ranks first in 7 data sets. LR ranks second by 5 data sets and PSC ranks third by 3 data sets. PLR gives a much worse result than the other algorithms on only the Ionosphere data set. PSC, LR and SVM have a good performance on average on all the data sets. PLR has low performance on 7 data sets in comparison to the other algorithms. While somewhat increasing the value of the epochs hyperparameter of PLR, its performance first rises and then levels off. But even though PLR catches up with the others' performance, a large value of the epochs hyperparameter causes high execution time. Figure 2 shows the comparisons of execution times of the classifiers on the data sets. Statistically, the average execution times of the methods point out comparative performance well. Accordingly, PSC, SVM, LR, and PLR correspond to 0.08, 3.53, 3.15, and 241.40, respectively. Accordingly, PSC is the fastest algorithm on all the data sets. The performance of SVM and LR in the context of running time are near each other. The running time of PLR is the highest over all the data sets. Consequently, SVM, PLR, and LR exhibit low performance on massive sample sizes and high-dimensional data sets.

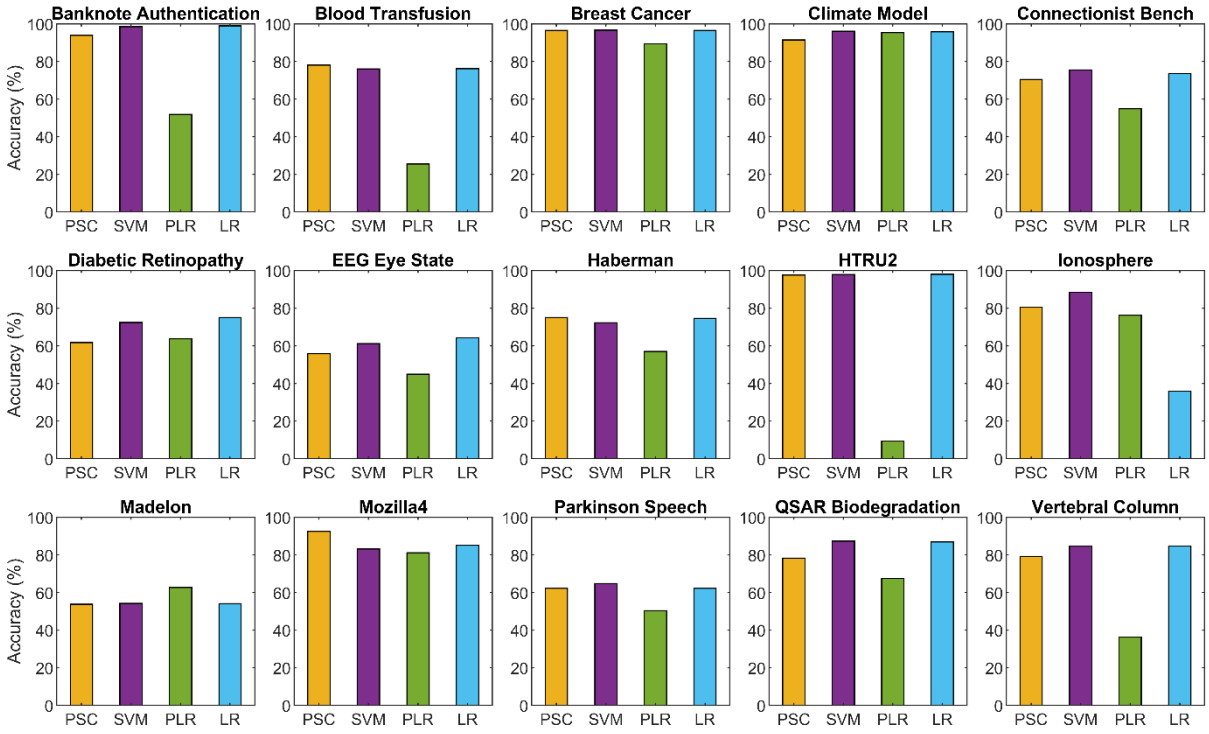

Figure 1. The comparative results of the algorithms relating to the classification accuracy.

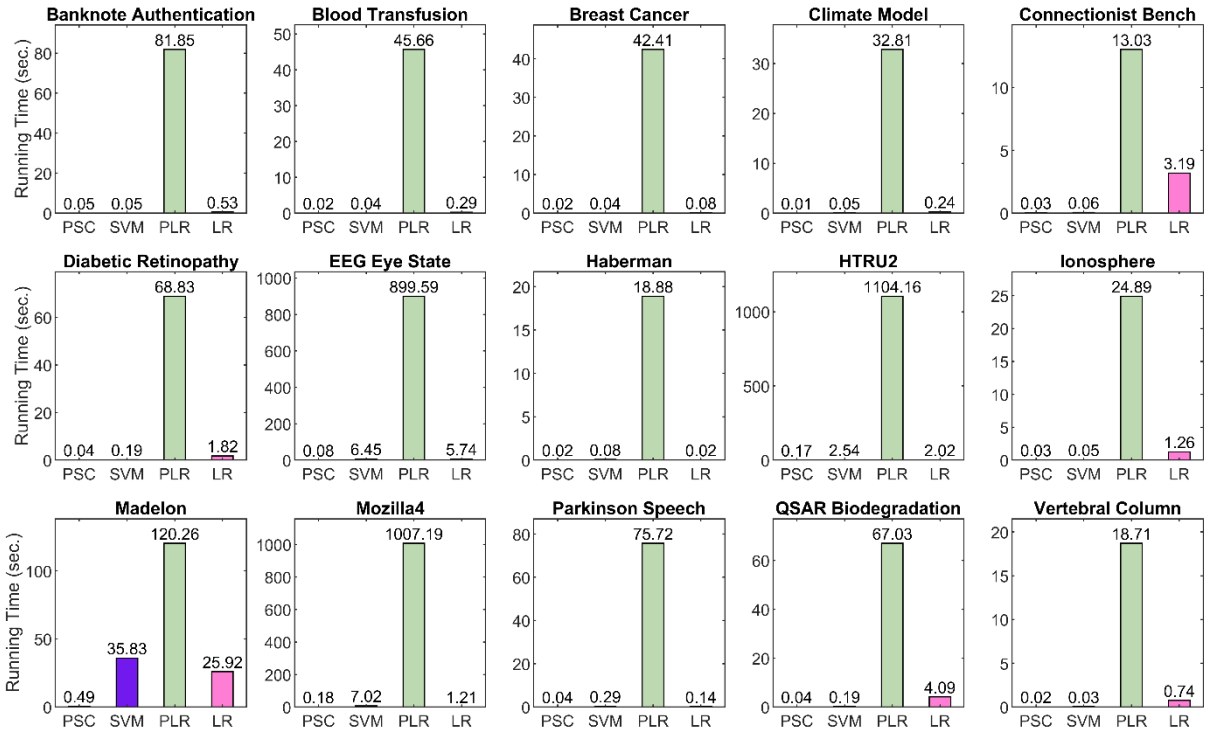

Figure 2. The comparative results of the classifiers with respect to the execution time

5.2. Adjusting the Hyperparameters of PSC

We first investigate the effects of the hyperparameters that can affect the performance of PSC. The hyperparameters that we detect are the swarm size, self-adjustment weight, social adjustment weight, and minimum neighbors' fraction. Figure 3 shows the change in the accuracy rate in terms of the swarm size. We first examine the influence of the swarm size hyperparameter on all the data sets. The lowest accuracy rate on the data set Banknote Authentication is approximately 93.9% with a swarm size of 10. The highest classification accuracy is approximately 99.0% by 70 particles. Accordingly, there is an important effect of the hyperparameter swarm size on the data set Banknote Authentication. Besides, the classification accuracy also rises as the swarm size increases. The lowest accuracy rate on the data set Blood Transfusion is approximately 76.5% with a swarm size of 80. The highest classification accuracy is approximately 78.2% by 40 particles. Accordingly, we can state that there exists a bit effect of the swarm size hyperparameter on the data set Blood Transfusion. Moreover, there is no proportional relationship between classification accuracy and swarm size. In other words, the classification accuracy does not rise as the swarm size increases. The lowest accuracy rate on the data set Breast Cancer is approximately 96.0% by swarm size of 20 and 100. The highest classification accuracy is approximately 97.6% by 90 particles. Accordingly, there exists little effect of the swarm size hyperparameter on the data set Breast Cancer. Further, there is no proportional relationship between classification accuracy and swarm size. The lowest classification accuracy on the data set Climate Model is approximately 90.6% with a swarm size of 50. The highest classification accuracy is approximately 93.7% by 70 particles. Accordingly, there exists the influence of the swarm size hyperparameter on the data set Climate Model. Furthermore, while swarm size increases, the classification accuracy first increases and then decreases after reaching the highest point. The lowest accuracy rate on the data set Connectionist Bench is approximately 69.7% with a swarm size of 20. The highest classification accuracy is approximately 79.8% by 90 particles. Accordingly, there is a significant effect of the hyperparameter swarm size on the data set Connectionist Bench. Furthermore, while swarm size increases, the classification accuracy generally increases. The lowest accuracy rate on the Diabetic Retinopathy data set is approximately 61.7% with a swarm size of 10. The highest classification accuracy is approximately 68.7% by 90 particles. As a result, there is a remarkable effect of the swarm size hyperparameter on the data set Diabetic Retinopathy. Furthermore, while swarm size increases, the classification accuracy mostly increases. The lowest classification accuracy on the data set EEG Eye State is approximately 55.7% with a swarm size of 10. The highest classification accuracy is approximately 58.2% by 90 particles. Consequently, there exists a bit effect of the swarm size hyperparameter on the data set EEG Eye State. Additionally, while swarm size increases, the classification accuracy slightly increases. The lowest accuracy rate on the data set Haberman is approximately 74.0% with a

swarm size of 30. The highest classification accuracy is approximately 76.1% by 80 particles. Consequently, there is an effect of the swarm size hyperparameter on the data set Haberman. In addition, while swarm size increases, the classification accuracy slightly increases. The lowest accuracy rate on the data set HTRU2 is approximately 97.6% with a swarm size of 10. The highest classification accuracy is approximately 97.9% by 100 particles. In consequence, there exists no effect of the hyperparameter swarm size on the data set HTRU2. In addition, while swarm size increases, there is no apparent change in the classification accuracy. The lowest accuracy rate on the data set Ionosphere is approximately 80.5% by swarm size of 10. The highest classification accuracy is approximately 87.6% by 100 particles. Accordingly, there is an apparent effect of the swarm size hyperparameter on the data set Ionosphere. Besides, while swarm size increases, the classification accuracy increases. The lowest accuracy rate on the data set Madelon is approximately 53.8% with a swarm size of 10. The highest classification accuracy is approximately 58.2% by 80 particles. Accordingly, there exists an effect of the hyperparameter swarm size on the data set Madelon. Besides, while swarm size increases, the classification accuracy first increases and then slightly levels off. The lowest accuracy rate on the data set Mozilla4 is approximately 92.5% with a swarm

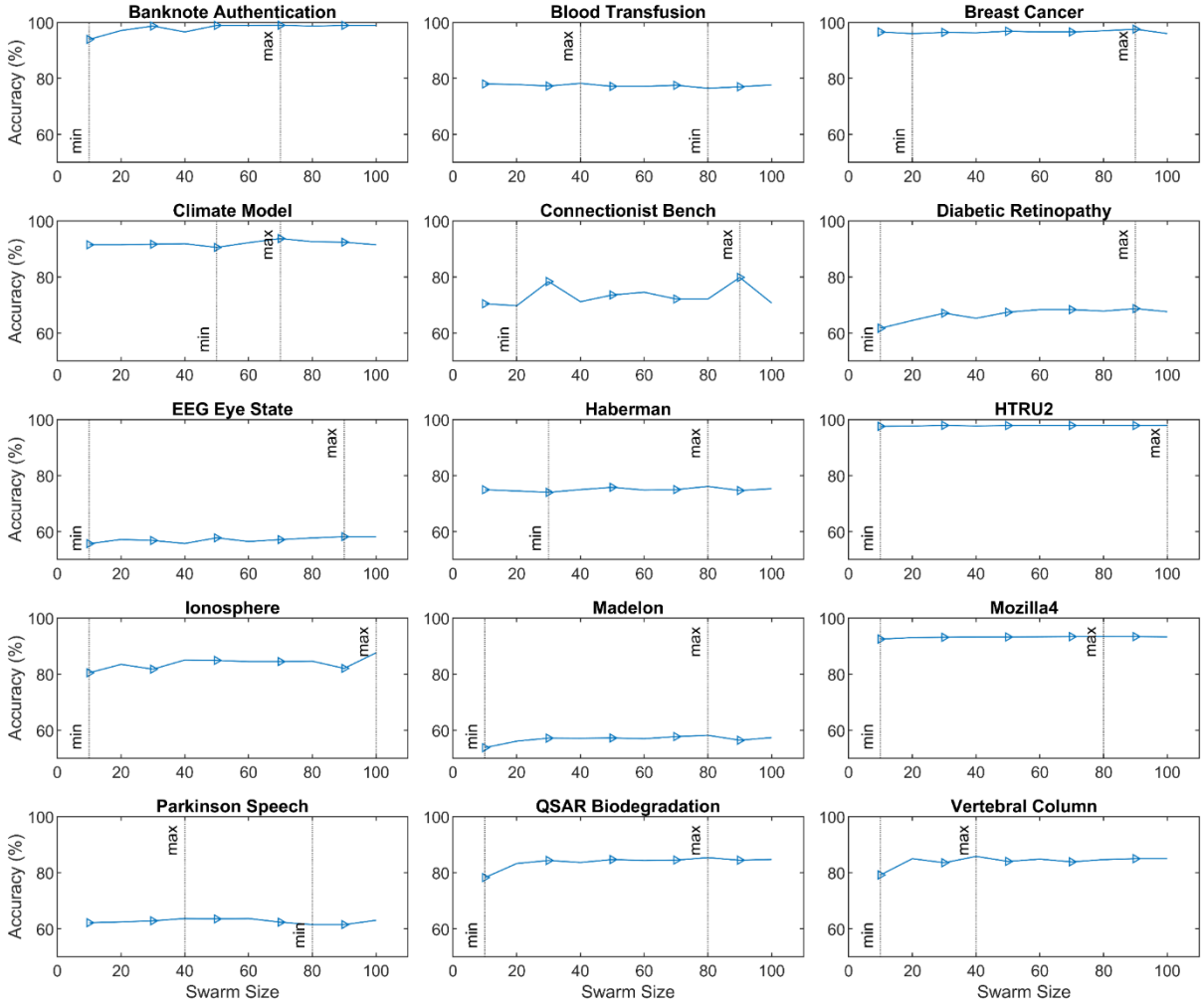

Figure 3. The change in the accuracy rate according to the swarm size

size of 10. The highest classification accuracy is approximately 93.4% by 80 particles. Accordingly, there is a low effect of the swarm size hyperparameter on the data set Mozilla4. Further, while swarm size increases, the classification accuracy first increases and then almost levels out. The lowest accuracy rate on the data set Parkinson Speech is approximately 61.5% by swarm size of 80 and 90. The highest classification accuracy is approximately 63.7% by 40 and 60 particles. Accordingly, there exists little effect of the swarm size hyperparameter on the data set Parkinson Speech. Besides, while swarm size increases, the change in the classification accuracy is not statistically significant. The lowest accuracy rate on the data set QSAR Bio Degradation is approximately 78.2% with a swarm size of 10. The highest classification accuracy is approximately 85.4% by 80 particles. Accordingly, there is a certain effect of the swarm size hyperparameter on the data set QSAR Bio Degradation. Besides, while

swarm size increases, the classification accuracy instantly increases after 10 particles and then levels out. The lowest accuracy rate on the data set Vertebral Column is approximately 79.2% by swarm size of 10. The highest classification accuracy is approximately 85.8% by 40 particles. Accordingly, there is a clear effect of the swarm size hyperparameter on the data set Vertebral Column. Besides, while swarm size increases, the classification accuracy instantly increases after 10 particles and then almost levels out. To sum up, there is an effect of the swarm size hyperparameter on some data sets. The swarm size effect on these data sets ranges from small to substantial. Overall speaking, the classification accuracy generally increases along with the many numbers of particles.

Figure 4 shows the change in the running time with respect to the swarm size. According to the results, it is obvious that the execution time increases as the swarm size grows unless considering a few ups and downs due to the experimental deviation (software and hardware-originated).

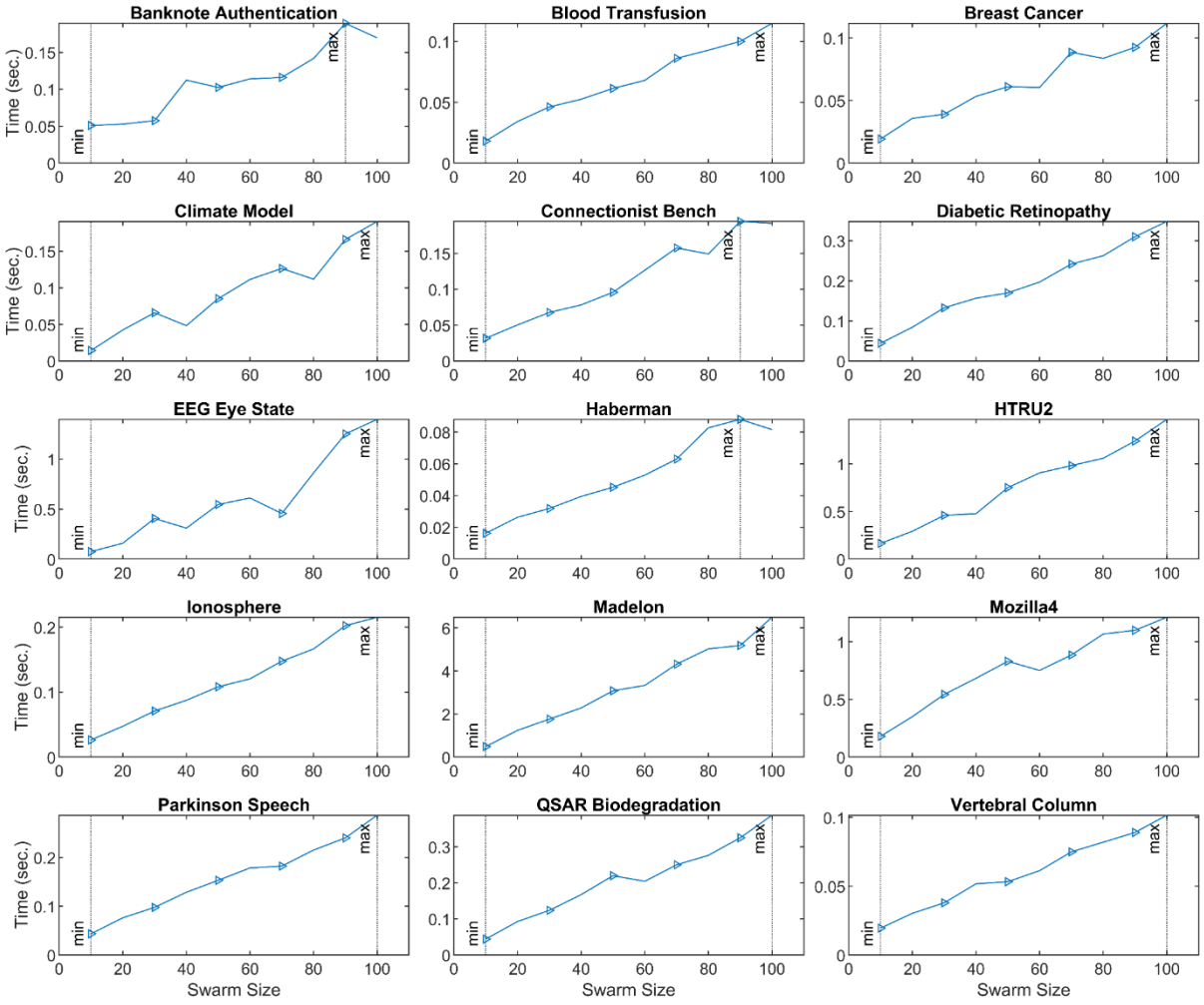

Figure 4. The change in the execution time according to the swarm size

Figure 5 shows the change in the classification accuracy according to the adjustment weight. We have obtained the results by changing the self-adjustment weight and social adjustment weight hyperparameters in the range 0.1 and 3 independently of each other. From a statistical point of view, the accuracy rates of the self-adjustment weight and social adjustment weight hyperparameters can better indicate comparative performance. Accordingly, the highest classification accuracy on the Banknote Authentication data set is approximately 97.0% when the value of the self-adjustment weight hyperparameter is 2. The highest classification accuracy is approximately 97.5% when the value of the social adjustment weight hyperparameter is 1.4. The highest classification accuracy on the Blood Transfusion data set is approximately 78.3% when the values of the self-adjustment weight hyperparameter are 0.2 and 1.7. The highest classification accuracy is approximately 78.5% when the values of the social adjustment weight hyperparameter are 1.9 and 2.1. The highest classification accuracy on the Breast Cancer data set is approximately 97.1% when the value of the self-adjustment weight hyperparameter is 1.5. The highest classification accuracy is approximately 97.1% when the value of the social adjustment weight hyperparameter is

2.8. The highest classification accuracy on the Climate Model data set is approximately 91.6% when the values of the self-adjustment weight hyperparameter are 1.5 and 2.1. The highest classification accuracy is approximately 91.6% when the value of the social adjustment weight hyperparameter is 1.6. The highest classification accuracy on the Connectionist Bench data set is approximately 73.1% when the value of the self-adjustment weight hyperparameter is 0.2. The highest classification accuracy is approximately 74.5% when the value of the social adjustment weight hyperparameter is 1.8. The highest classification accuracy on the Diabetic Retinopathy data set is approximately 64.0% when the value of the self-adjustment weight hyperparameter is 2.4. The highest classification accuracy is approximately 64.7% when the value of the social adjustment weight hyperparameter is 2.3. The highest classification accuracy on the EEG Eye State data set is approximately 57.6% when the value of the self-adjustment weight hyperparameter is 1.6. The highest classification accuracy is approximately 57.4% when the value of the social adjustment weight hyperparameter is 1.7. In terms of the self-adjustment weight hyperparameter, the accuracy rates and corresponding parameter values of the Haberman, HTRU2, Ionosphere, Madelon, Mozilla4, Parkinson Speech, QSAR Biodegradation, Vertebral Column data sets are approximately 76.0%, 97.8%, 81.9%, 56.1%, 93.4%, 63.1%, 82.7%, and 84.8% by 1.4, 1.1, 1.8, 1.9, 2.5, 1.7, 2.1, and 2.8, respectively. In terms of the social adjustment weight hyperparameter, the accuracy rates and corresponding parameter values of the Haberman, HTRU2, Ionosphere, Madelon, Mozilla4, Parkinson Speech, QSAR Biodegradation, Vertebral Column data sets are approximately 75.3%, 97.8%, 82.2%, 56.4%, 93.4%, 63.1%, 82.7%, and 84.4% by 1.6, 0.3, 1.5, 1.9, 2.4, 2, 1.9, and 1.9, respectively. Consequently, we observe that the classification accuracy varies along with the change in the self-adjustment weight and social adjustment weight hyperparameters. The classification accuracies are seen to be higher as compared to the results obtained from the default parameter values of PSC. Further, the classification accuracy fluctuates instead of regularly increasing while the self-adjustment weight and social adjustment weight hyperparameters increase.

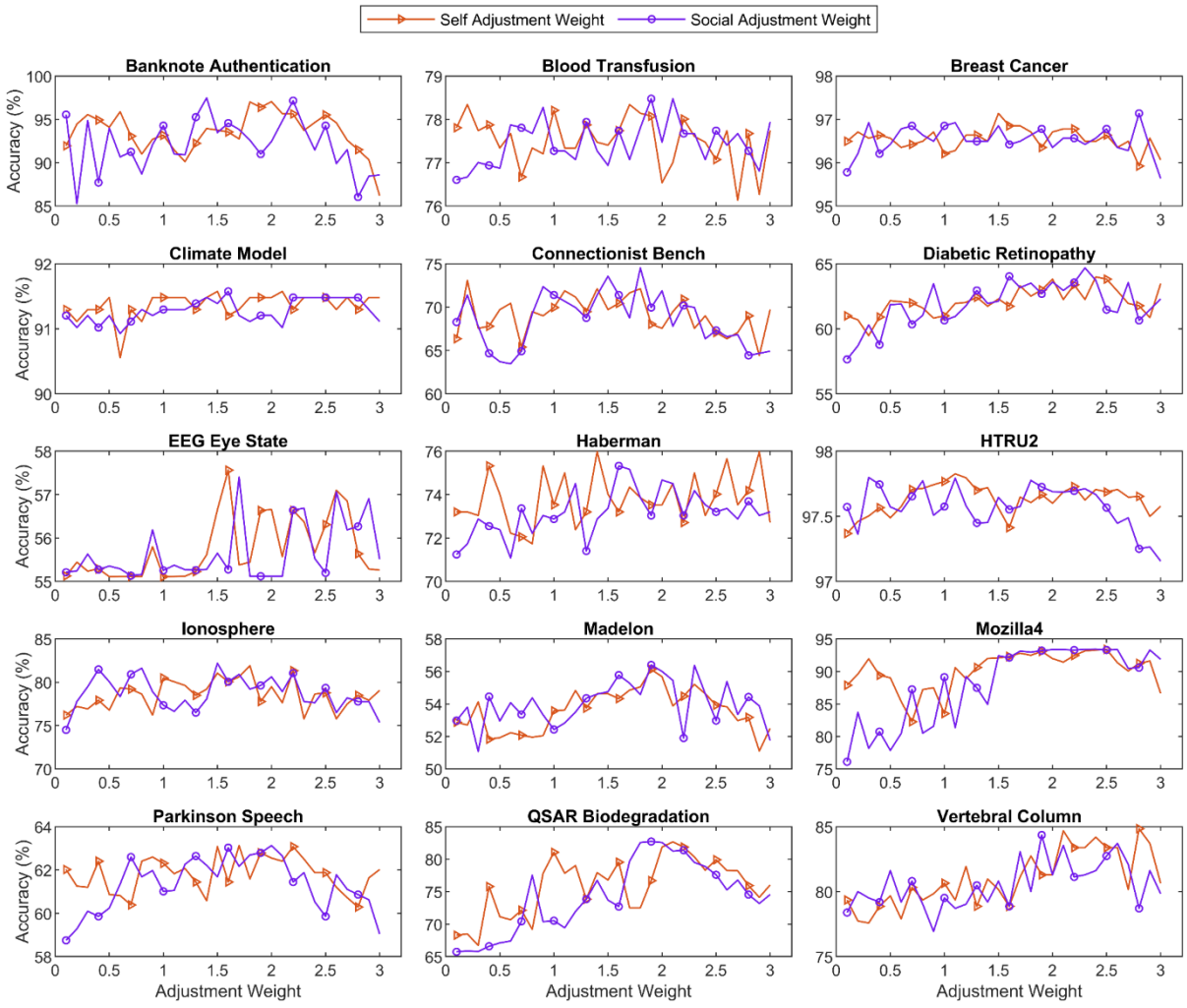

Figure 5. The change in the classification accuracy according to the adjustment weight

Table 3 shows the change in the maximum accuracy rates on the data sets according to the minimum neighbors' fraction hyperparameter. Accordingly, the minimum neighbors' fraction hyperparameter also influences the classification accuracy.

Table 3. The change of the maximum accuracy rates on the data sets according to the minimum neighbors' fraction hyperparameter.

Data set	Accuracy rate	Parameter value
Banknote Authentication	95.1%	0.8
Blood Transfusion	78.4%	0.6
Breast Cancer	96.6%	0.2, 1.0
Climate Model	91.5%	0.1, 0.2, 0.3, 0.5, 0.8, 1.0
Connectionist Bench	73.6%	0.3
Diabetic Retinopathy	64.9%	0.7
EEG Eye State	57.5%	0.8
Haberman	75.8%	0.9
HTRU2	97.8%	0.8
Ionosphere	82.2%	0.2
Madelon	54.9%	0.6
Mozilla4	92.9%	0.3
Parkinson Speech	62.7%	0.5
QSAR Biodegradation	80.7%	0.6
Vertebral Column	82.6%	0.8

Figure 6 shows the variation in the classification accuracy with both adjustment weights at the same time. As the color scale in the colormap approaches white, the accuracy rate rises. As the color scale approaches black, the accuracy rate decreases. The classification accuracy ranges from 50% to 100%. Generally speaking, and neglecting the exceptions, when the values of the self-adjustment weight and social adjustment weight hyperparameters are simultaneously increased or decreased, a considerable rise in the classification accuracy is not seen. On the large values of the social adjustment weight hyperparameter and the small or medium values of the self-adjustment weight hyperparameter, high classification accuracies can be obtained. For the small values of the social adjustment weight hyperparameter, we have acquired low classification accuracies on the data sets. We specify that higher classification accuracies can be obtained when the values of the self-adjustment weight and social adjustment weight hyperparameters simultaneously vary. Considering the data sets in detail in terms of both adjustment weights, the self-adjustment weight and social adjustment weight hyperparameters significantly affect classification accuracies on the Banknote Authentication, Ionosphere, Connectionist Bench, Mozilla4, QSAR Biodegradation, and Vertebral Column data sets. Further, the self-adjustment weight and social adjustment weight hyperparameters have little effect in terms of classification accuracies on the Blood Transfusion, Climate Model, Haberman, and Parkinson Speech data sets. Finally, the self-adjustment weight and social adjustment weight hyperparameters have a bit of an effect in terms of classification accuracies on the Breast Cancer, Diabetic Retinopathy, EEG Eye State, HTRU2, and Madelon data sets. To sum up, the effect of the self-adjustment weight and social adjustment weight hyperparameters on the data sets changes. Hence, it is needed to observe the effect by changing the values of these hyperparameters.

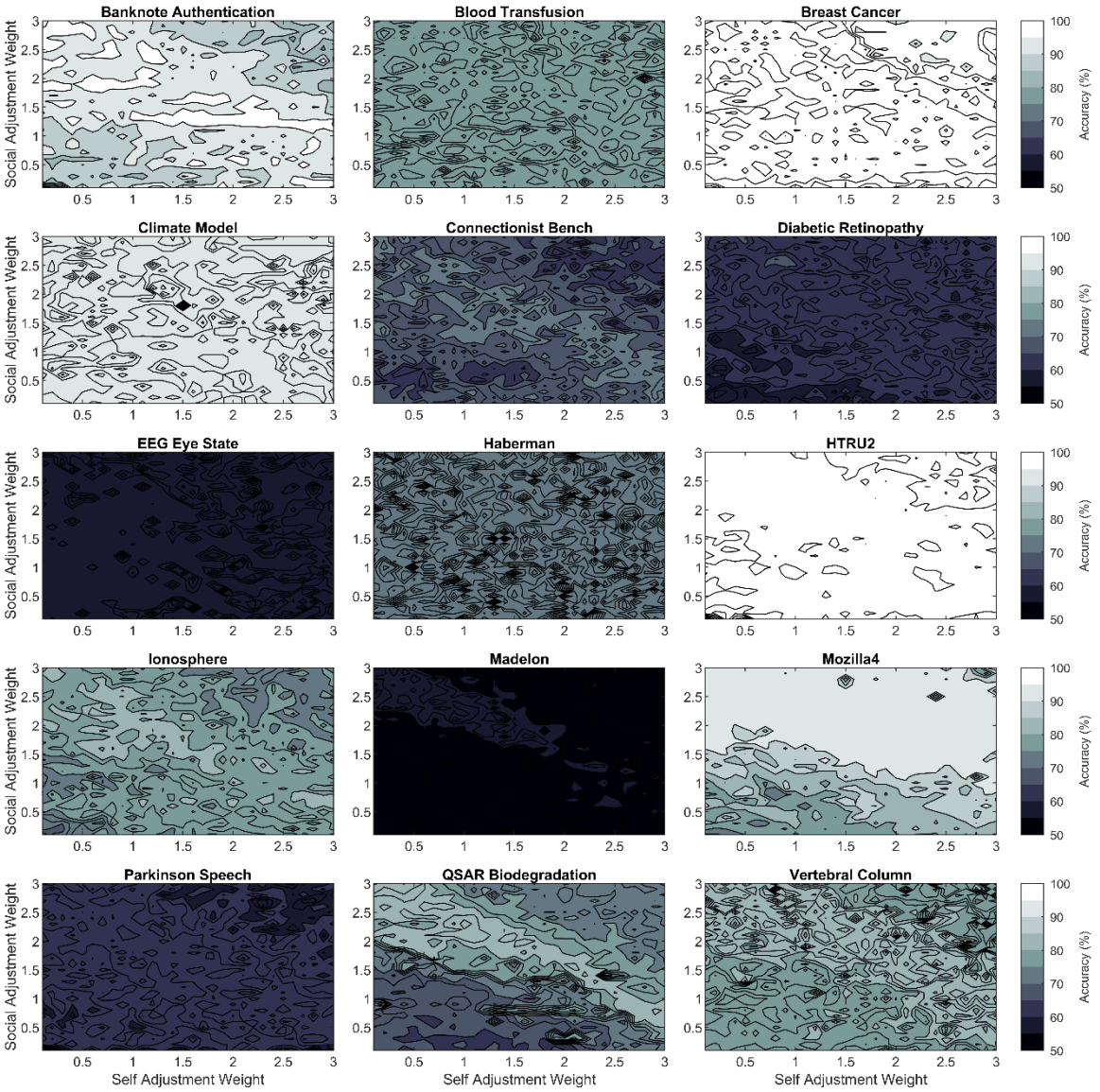

Figure 6. The variation in the classification accuracy with both adjustment weights

Table 4 shows the accuracy rate of PSC on the data sets after setting its adjustment weight parameters. According to the results, the average classification accuracy of PSC increases by 3.0% in comparison to the default parameter values.

As a result, the average performances of PSC, SVM, and LR are very close to each other and the difference between them is not statistically significant. Therefore, we remark that a linear classifier developed by using PSO can compete with the other linear classifiers. In order to increase the classification performance of PSC, more different objective functions can be enhanced, or the existing ones can be improved more. Moreover, tighter constraints can be constructed with the other meta-heuristic methods and the classification accuracy can be provided to be increased even a bit more. In a nutshell, the experimental results show that meta-heuristics can be exploited to build linear models.

Table 4. The accuracy rate of PSC on the data sets after setting its adjustment weight parameters.

Data set	PSC (tuned)	Self-adjustment weight	Social adjustment weight
Banknote Authentication	98.7% ± 1.0	2.3	1.3
Blood Transfusion	79.1% ± 0.2	0.4	2.2
Breast Cancer	97.4% ± 0.1	1.7	1.9
Climate Model	92.0% ± 0.0	2.9	1.8
Connectionist Bench	75.2% ± 2.7	1.4	1.7
Diabetic Retinopathy	66.1% ± 0.7	0.8	2.6
EEG Eye State	58.1% ± 0.2	2.3	1.7
Haberman	75.8% ± 1.2	2.3	1.1
HTRU2	97.9% ± 0.1	0.5	2.3
Ionosphere	83.5% ± 2.1	0.6	2.2
Madelon	60.0% ± 0.3	1.4	2.1
Mozilla4	93.4% ± 0.5	0.3	3.0
Parkinson Speech	64.6% ± 0.9	2.8	0.7
QSAR Biodegradation	84.5% ± 0.8	0.9	2.3
Vertebral Column	85.8% ± 1.9	2.0	2.1
Avg.	80.8% ± 13.9	1.5	1.9

6. Conclusion

In this study, we first attempt to reply to the question of whether meta-heuristics can be exploited to build linear models or not. To this end, Particle Swarm Optimization (PSO) has been employed to handle linear classification problems. A determined objective function has been tried on 15 data sets and compared to well-known three linear classifiers (i.e., SVM, PLR, and LR). The experimental results show that the Particle Swarm Classifier (PSC) is competitive with the other classification algorithms, and for a few binary classification problems, it turns out to be superior to other classifiers. In terms of running time, PSC is undoubtedly faster than the other classifiers. In particular, the speed of PSC can be apparently seen as compared to the other classifiers on large and high-dimensional data sets. According to the experimental results, the average performances of PSC, SVM, and LR are similar to each other and the difference between their performance is statistically insignificant. From a statistical viewpoint, the average classification accuracies of PSC (tuned), SVM, and LR are 80.8%, 80.6%, and 77.4%, respectively. To put it short, the experiments show that meta-heuristics can be used to generate linear models. Concerning future works, more efficient and effective objective functions will be considered and evaluated. In order to increase the classification performance of PSC, more advanced objective functions can be developed, or the current objective functions can be improved satisfactorily. Further, tighter constraints can be formed by means of the other meta-heuristic methods and the classification accuracy can be increased more and more.

References

Ahmad, Z., Li, J., & Mahmood, T. (2023). Adaptive Hyperparameter Fine-Tuning for Boosting the Robustness and Quality of the Particle Swarm Optimization Algorithm for Non-Linear RBF Neural Network Modelling and Its Applications. *Mathematics*, *11*(1), 242. https://doi.org/10.3390/math11010242

Almasi, O. N., & Khooban, M. H. (2018). A parsimonious SVM model selection criterion for classification of real-world data sets via an adaptive population-based algorithm. *Neural Computing and Applications*, *30*(11), 3421–3429. https://doi.org/10.1007/s00521-017-2930-y

Asif, M., Nagra, A. A., Ahmad, M. Bin, & Masood, K. (2022). Feature Selection Empowered by Self-Inertia Weight Adaptive Particle Swarm Optimization for Text Classification. *Applied Artificial Intelligence*, *36*(1). https://doi.org/10.1080/08839514.2021.2004345

Athira Lekshmi, B. A., Linsely, J. A., Queen, M. . F., & Babu Aurtherson, P. (2018). Feature Extraction and Image Classification Using Particle Swarm Optimization by Evolving Rotation-Invariant Image Descriptors. *2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR)*, 1–5. https://doi.org/10.1109/ICETIETR.2018.8529083

Ay, Ş., Ekinci, E., & Garip, Z. (2023). A comparative analysis of meta-heuristic optimization algorithms for feature selection on ML-based classification of heart-related diseases. *The Journal of Supercomputing*.

https://doi.org/10.1007/s11227-023-05132-3

Aydin, F. (2023). Unsupervised instance selection via conjectural hyperrectangles. *Neural Computing and Applications*, *35*(7), 5335–5349. https://doi.org/10.1007/s00521-022-07974-z

Aziz, Y., & Memon, K. H. (2023). Fast geometrical extraction of nearest neighbors from multi-dimensional data. *Pattern Recognition*, *136*, 109183. https://doi.org/10.1016/j.patcog.2022.109183

Balamurugan, R., Natarajan, A. M., & Premalatha, K. (2015). Stellar-Mass Black Hole Optimization for Biclustering Microarray Gene Expression Data. *Applied Artificial Intelligence*, *29*(4), 353–381. https://doi.org/10.1080/08839514.2015.1016391

Baygin, N., Baygin, M., & Karakose, M. (2019). A SVM-PSO Classifier for Robot Motion in Environment with Obstacles. *2019 International Artificial Intelligence and Data Processing Symposium (IDAP)*, 1–5. https://doi.org/10.1109/IDAP.2019.8875921

Berkson, J. (1944). Application of the Logistic Function to Bio-Assay. *Journal of the American Statistical Association*, *39*(227), 357–365. https://doi.org/10.1080/01621459.1944.10500699

Bratton, D., & Kennedy, J. (2007). Defining a Standard for Particle Swarm Optimization. *2007 IEEE Swarm Intelligence Symposium*, 120–127. https://doi.org/10.1109/SIS.2007.368035

Bumin, M., & Ozcalici, M. (2023). Predicting the direction of financial dollarization movement with genetic algorithm and machine learning algorithms: The case of Turkey. *Expert Systems with Applications*, *213*, 119301. https://doi.org/10.1016/j.eswa.2022.119301

Chamkalani, A., Zendehboudi, S., Bahadori, A., Kharrat, R., Chamkalani, R., James, L., & Chatzis, I. (2014). Integration of LSSVM technique with PSO to determine asphaltene deposition. *Journal of Petroleum Science and Engineering*, *124*, 243–253. https://doi.org/10.1016/j.petrol.2014.10.001

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

De Falco, I., Della Cioppa, A., & Tarantino, E. (2007). Facing classification problems with Particle Swarm Optimization. *Applied Soft Computing*, *7*(3), 652–658. https://doi.org/10.1016/j.asoc.2005.09.004

Elshamy, W., Emara, H. M., & Bahgat, A. (2007). Clubs-based Particle Swarm Optimization. *2007 IEEE Swarm Intelligence Symposium*, 289–296. https://doi.org/10.1109/SIS.2007.367950

Eshtay, M., Faris, H., Heidari, A. A., Al-Zoubi, A. M., & Aljarah, I. (2021). AutoRWN: automatic construction and training of random weight networks using competitive swarm of agents. *Neural Computing and Applications*, *33*(11), 5507–5524. https://doi.org/10.1007/s00521-020-05329-0

Hu, J., Ou, X., Liang, P., & Li, B. (2022). Applying particle swarm optimization-based decision tree classifier for wart treatment selection. *Complex & Intelligent Systems*, *8*(1), 163–177. https://doi.org/10.1007/s40747-021-00348-3

Karakurt, M., Oymak, E. A., Hark, H., Erdoğan, M. C., & Karcı, A. (2022). Karcı Sinir Ağlarının Uygulaması ve Performans Analizi. *Computer Science*, *7*(2), 68–80. https://doi.org/10.53070/bbd.1194017

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, *4*, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

Khennak, I., Drias, H., Drias, Y., Bendakir, F., & Hamdi, S. (2023). I/F-Race tuned firefly algorithm and particle swarm optimization for K-medoids-based clustering. *Evolutionary Intelligence*, *16*(1), 351–373. https://doi.org/10.1007/s12065-022-00794-z

Kotary, D. K., & Nanda, S. J. (2020). Distributed robust data clustering in wireless sensor networks using diffusion moth flame optimization. *Engineering Applications of Artificial Intelligence*, *87*, 103342. https://doi.org/10.1016/j.engappai.2019.103342

Lai, D. T. C., & Sato, Y. (2021). An Empirical Study of Cluster-Based MOEA/D Bare Bones PSO for Data Clustering †. *Algorithms*, *14*(11), 338. https://doi.org/10.3390/a14110338

Lawrence, T., Zhang, L., Rogage, K., & Lim, C. P. (2021). Evolving Deep Architecture Generation with Residual Connections for Image Classification Using Particle Swarm Optimization. *Sensors*, *21*(23), 7936. https://doi.org/10.3390/s21237936

Lenat, D. B., & Feigenbaum, E. A. (1991). On the thresholds of knowledge. *Artificial Intelligence*, *47*(1–3), 185–

250. https://doi.org/10.1016/0004-3702(91)90055-O

Ma, T., Wang, C., Wang, J., Cheng, J., & Chen, X. (2019). Particle-swarm optimization of ensemble neural networks with negative correlation learning for forecasting short-term wind speed of wind farms in western China. *Information Sciences*, *505*, 157–182. https://doi.org/10.1016/j.ins.2019.07.074

Mason, K., Duggan, J., & Howley, E. (2018). A meta optimisation analysis of particle swarm optimisation velocity update equations for watershed management learning. *Applied Soft Computing*, *62*, 148–161. https://doi.org/10.1016/j.asoc.2017.10.018

Mezura-Montes, E., & Coello Coello, C. A. (2011). Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, *1*(4), 173–194. https://doi.org/10.1016/j.swevo.2011.10.001

Oliveira, M., Pinheiro, D., Andrade, B., Bastos-Filho, C., & Menezes, R. (2016). Communication Diversity in Particle Swarm Optimizers. In *Lecture Notes in Computer Science* (pp. 77–88). Springer, Cham. https://doi.org/10.1007/978-3-319-44427-7_7

Omran, M. G., Engelbrecht, A. P., & Salman, A. (2002). Image Classification Using Particle Swarm Optimization. *The Fourth Asia–Pacific Conference on Recent Advances in Simulated Evolution and Learning*, 347–365. https://doi.org/10.1142/9789812561794_0019

Omran, M. G. H., Engelbrecht, A. P., & Salman, A. (2006). Particle Swarm Optimization for Pattern Recognition and Image Processing. In A. Abraham, C. Grosan, & V. Ramos (Eds.), *Swarm Intelligence in Data Mining* (pp. 125–151). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-34956-3_6

Pedersen, M. E. H. (2010). *Good parameters for particle swarm optimization*. https://citeseerx.ist.psu.edu/doc/10.1.1.298.4359

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519

Russell, S. J., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall.

Sivrikaya, Ö. E., Yüksekgönül, M., & Baydoğan, M. G. (2021). Learning prototypes for multiple instance learning. *Turkish Journal of Electrical Engineering & Computer Sciences*, *29*(7), 2901–2919. https://doi.org/10.3906/elk-2101-103

Sousa, T., Silva, A., & Neves, A. (2004). Particle Swarm based Data Mining Algorithms for classification tasks. *Parallel Computing*, *30*(5–6), 767–783. https://doi.org/10.1016/j.parco.2003.12.015

Taherkhani, M., & Safabakhsh, R. (2016). A novel stability-based adaptive inertia weight for particle swarm optimization. *Applied Soft Computing*, *38*, 281–295. https://doi.org/10.1016/j.asoc.2015.10.004

Tan, T. Y., Zhang, L., & Lim, C. P. (2019). Intelligent skin cancer diagnosis using improved particle swarm optimization and deep learning models. *Applied Soft Computing*, *84*, 105725. https://doi.org/10.1016/j.asoc.2019.105725

Telikani, A., Tahmassebi, A., Banzhaf, W., & Gandomi, A. H. (2022). Evolutionary Machine Learning: A Survey. *ACM Computing Surveys*, *54*(8), 1–35. https://doi.org/10.1145/3467477

Yilmaz, A., & Yolcu, U. (2022). Dendritic neuron model neural network trained by modified particle swarm optimization for time- series forecasting. *Journal of Forecasting*, *41*(4), 793–809. https://doi.org/10.1002/for.2833

Yin, P.-Y., Glover, F., Laguna, M., & Zhu, J.-X. (2011). A Complementary Cyber Swarm Algorithm. *International Journal of Swarm Intelligence Research*, *2*(2), 22–41. https://doi.org/10.4018/jsir.2011040102

Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). Recent Advances of Large-Scale Linear Classification. *Proceedings of the IEEE*, *100*(9), 2584–2603. https://doi.org/10.1109/JPROC.2012.2188013