

# **Human Activity Recognition based on the sensory data using Optimized KNN Algorithm**

**STUDENT NAME: Meng Tianao**

**STUDENT ID: 20810553**

**COURSE: ECE 602**

## Introduction

In this paper, I will introduce a novel model for human activity recognition (HAR) that based on signals extracted by wearing a smartphone on the waist. The dataset is a combination of several signals extracted from an accelerometer and gyroscope when with different human activities [Davide et al., 2013].

It is a key element to understand people's actions for the development of the intelligent systems. HAR is a research field that specifically deals with this issue through the integration of sensing and reasoning, in order to deliver context-aware data that can be employed to provide personalized support in many applications (Chen et al. 2012). For example, imagine an Intelligent home with sensors able to detect people's presence and the use of household appliances. We can infer the activities performed by its users based on the sensors signals along with time of the day and date (e.g. in the morning time a person stays in the kitchen while a coffee machine is on suggests that person is making breakfast). Consequently, the collected HAR information can be exploited to anticipate future people requirements and become responsive to them (e.g. by automatically pre-heating the coffee machine, controlling room lighting and temperature, etc.) [Ortiz et al., 2015].

In the HAR framework, we still need to address several issues. For example, obtrusiveness of current wearable sensors; lack of fully pervasive systems able to reach users at any location any time; privacy concerns regarding invasive and continuous monitoring of activities (e.g. by using video cameras); difficulty of performing HAR in real-time; battery limitations of wearable devices; and dealing with content extraction from sparse multi-sensor data,[Ortiz et al., 2015] and also some machine learning algorithms efficiency.

In my project, I will focus on the machine learning algorithms to solve the prediction of human activities. Based on the numerical optimization strategy to optimize the machine learning algorithm. In the following part, literature review, I will list some famous machine learning algorithms and their performance on the prediction on human activities; Furthermore, I will present the reasons why I choose k-NN to optimize and why I choose PSO strategy to optimize this algorithm.

## Literature Review

In this term paper, I will focus on the machine learning algorithms that have been exploited these years. Many learning algorithms were used for predicting human activities, such as, Support Vector Machines (SVMs), Naive Bayesian (NB), and artificial Neural Networks (ANNs), [Preece et al., 2009]. In supervised learning methods, there are two main phases, namely training and testing phases. In the training or learning phase, the parameters of a classifier are adjusted using the input data, i.e. training samples, and their corresponding outputs, i.e. targets or responses. Next, the classifier can be used to estimate the class label for an unknown sample [Luts et al., 2010]. There are different types of classifiers and each classifier has different parameters, which control the accuracy of that classifier [Bedogni et al. 2016, 2015], Reinhardt et al. [2013], Yao et al. [2008]. Table 1 shows some state-of-the-art human activity classification models. k-Nearest Neighbor (k-NN) classifier is one of the simplest classifiers. Thus, it has been used in different applications. The main idea of k-NN classifier is to choose the nearest k labeled samples around an unknown sample (or test sample) and assigns the label that most those k neighbors have. Thus, the k-NN classifier has no explicit training phase, and there is no classification model is built, and hence

Reference	Classifier	# Activities	# Subjects	Results (%)
(Anguita, Ghio, Oneto, Parra, & Reyes-Ortiz, 2012)	SVM	6	30	Recall=89
(Mantjarvi, Himberg, & Seppanen, 2001)	ANN	4	6	Accuracy 83–90
(Song & Wang, 2005)	k-NN	5	6	Accuracy 86.6
(Aminian et al., 1999)	Threshold-Based	4	5	Accuracy 89.3
(Bao & Intille, 2004)	k-NN, NB, DT	20	20	Accuracy 84
(Allen, Ambikairajah, Lovell, & Celler, 2006)	GMM	8	6	Accuracy 91.3

**Table 1.** State-of-the-art of human activity classification systems (GMM is short for Gaussian Mixture Model and DT is short for Decision Trees). [Alaa Tharwat et al., 2018]

all training samples are needed during the testing phase [Duda et al., 2012, Tharwat et al., 2013]. In the k-NN classifier, the neighborhood parameter k plays an important role in the accuracy of the k-NN classifier. Mohammed Islam et al. investigated the influence of k parameter on the accuracy of k-NN classifier using try and error method [Islam et al., 2007]. They used the Euclidean distance and different values of k. They found that the best accuracy achieved when the value of k was five. In another study, cross-validation methods were used to estimate the misclassification rate for different values of k and choose the k value which leads to the lowest misclassification rate [Lachenbruch Mickey, 1968, Stone, 1978].

Therefore, in my term paper, I will optimize the method of choosing k using PSO.

## Proposed Solution

In this term paper, I will use the KNN algorithm with PSO optimization method to choose the best  $k$  within the max iteration num to improve the geometric approach KNN mentioned ahead.

1.

The  $k$ -NN classifier is one of the famous and simple machine learning algorithms. it is widely used in a variety of applications because it makes no assumptions on the input data distribution. And it was first developed by Fix and Hodges as a non- parametric algorithm [Duda et al., 2012, Fix Hodges Jr, 1951].

2.

PSO was first introduced by Kennedy and Eberhart in 1995 [Yang, 2014]. PSO have been applied in many optimization problems by many experts. For example, Liu et al use PSO to select the most discriminative features while Lin et al used PSO as a feature selection and to find a better SVM parameters to achieve a higher accuracy [Alaa Tharwat et al., 2018] .Moreover, PSO was used to search for the optimal Proportional-Integral-Derivative (PID) controller parameters of an Automatic Voltage Regulator (AVR) system [Gaing, 2004, Kao et al., 2006]. In this term paper, I propose a PSO-based algorithm called PSO-kNN. In this proposed algorithm, PSO was used to find the  $k$  parameter in the  $k$ -NN classifier to achieve a lower error rate.

The underlying rules of the birds' movements, changing direction, flocking, and regrouping are discovered by Reynolds and Heppner [Heppner Grenander, 1990, Reynolds, 1987] and simulated by Kennedy and Eberhart [Eberhart Kennedy, 1995]. The particle swarm optimization) is also an easy algorithm like  $k$ -NN. The aim of PSO algorithm is to search in the search space to discover the positions closing to the global minimum or maximum solution. The dimension of the search space is denoted by the number of parameters we need to optimize for the position. In my term paper, the number of parameters to be optimized is one, only  $k$ ; hence, the PSO is used to search only in 1-dimension space. In the PSO algorithm, the number of particles is determined by the user and these particles are randomly placed in the search space. The  $G$  in the below equation (2) means the best position among all the particles current and historical position. If we implement the particle in OOP method, each particle should have three fields, position ( $x_i \in R_n$ ), velocity ( $v_i$ ), and the previous best positions ( $p_i$ ).

The position of each particle ( $x_i$ ) is represented using a point in the certain search space. During the search process, the current positions of all particles are evaluated comparing with the previous best positions to determine whether the current positions update the previous best positions ( $p_i$ ) or not. In other words, the previous best positions store the best positions of the all particles positions. In my case, the lower error rate determined by 10 times 10 folds cross validation is better. The particles that have better objective values are closer to the local or global, i.e., minimum or maximum, solution. The

velocity in different iteration is calculated according to Equation (1). From Equation (1), the new velocity for i-th particle in the search space is determined by:

- 1) The current velocity of i-th particle ( $w * v_i(t)$ ).
- 2) The position of the previous best position of that particle that is called particle memory or cognitive component. This term is used to adjust the velocity towards the best position visited by that particle ( $C1 * r1 * (p_i(t) - x_i(t))$ ).
- 3) The position is denoted as social component ( $C2 * r2 * (G - x_i(t))$ ) that is used to add the velocity towards the global best position of all particles current and historical positions.

The new position of any particle is then calculated by adding the velocity and the current position of that particle as in Equation (3). Hence, the PSO algorithm uses the current  $x_i$ ,  $p_i$ ,  $v_i$ , and  $G$ , to search for better positions by keep moving the particles towards the global solution.

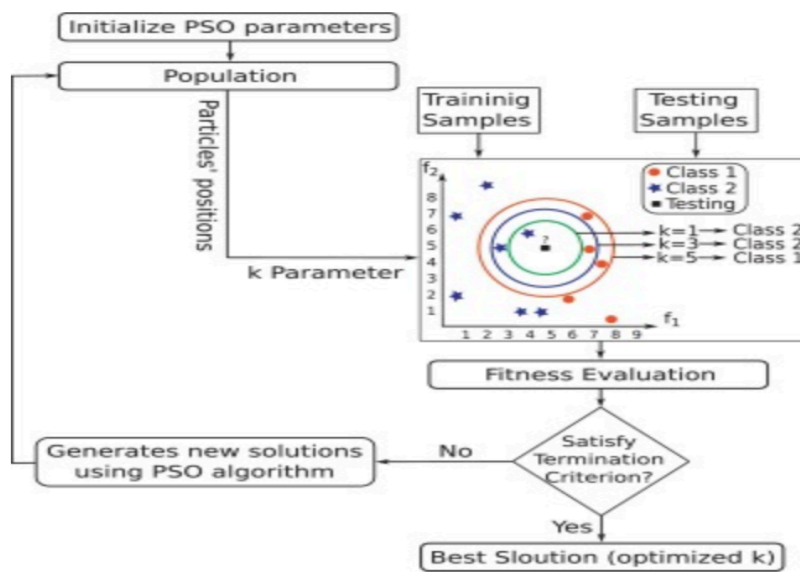
$$v_i(t+1) = w * v_i(t) + C1 * r1 * (p_i(t) - x_i(t)) + C2 * r2 * (G - x_i(t)) \quad (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

where  $r1$ ,  $r2$  are uniformly generated random numbers in the range of  $[0,1]$ ,  $w$  is the inertia weight,  $C1$  is the cognition learning factor,  $C2$  is the social learning factor, and  $p_i$  is the best solution of the i-th particle. Because in the particles' velocity calculation, it involves some random variables, all particles move randomly. If the updated velocity is too high, it could prevent the particles from converging to the optimal solution; Therefore, it might be limited to a range  $[-V_{max}, V_{max}]$  in the search area. On the other hand, a small value of velocity could lead that all particles to search in a relevant small area. Thus, leading to not find a global optimal solution. Both positions and velocity of all particles change iteratively until it reaches a predefined stopping criterion [Eberhart Kennedy, 1995, Kennedy, 2010].

### 3.

As mentioned above, the k-NN classifier has only one parameter,  $k$ . If we choose a small value of  $k$ , the noise samples could have a huge influence on the final results. On the other hand, if we select a relevant large  $k$ , it would decrease the weight of each of  $k$  chosen samples and spend more time on computation, which increases the time complexity. Thus, leading to a proposed PSO-kNN algorithm to search in given space for the value of  $k$ . The details of our proposed algorithm are shown in the Figure 1.



**Figure 1**

According to some my literature reviews on the k-NN, we can get that Mohammed Islam et al. used the Euclidean distance and different values of k. They found that the best accuracy achieved when the value of k was five; Therefore, the k =5 k-NN algorithm will be implemented as the comparative method in my project.

We also can get that there exists the cross-validation methods method to calculate the error rate, then leads to find a better results(error rate in my project); Therefore, I will do some innovation by reduction by combining the cross-validation with our original PSO-KNN algorithms; And because of the PSO optimization method used in KNN(which is applied in wide range of areas no matter in engineering or science), my method also involves innovation by model transfer, numeric, and optimization.

## Experimental Process and Experimental Results

### 1.Dataset for this experiment:

In this experiment, HAR dataset which is mentioned in the introduction part was used.

### 2.Dataset preprocessing:

Because there are total 571 features for each sample in both train and test dataset and there are total 7352 datapoint in the train dataset, I use the PCA (principal component analysis) to do the dimension reduction, which make the accuracy of the processed data is 95%. The number of dimensions after PCA processing is 67.

```
PCA len: 67  
len: 7352
```

### 3.Check my Implementation of PSO-KNN can work properly using ten-times-ten-fold cross validation:

In this part, I simply choose number of particles (nPOP) is 1 and max number of iterations is one to do a setup experiment. And Because the ten-times-ten-fold strategy takes a long running time, in the parameter investigation part I only use the single ten-fold cross validation strategy. The experiment result is shown in Figure 2.

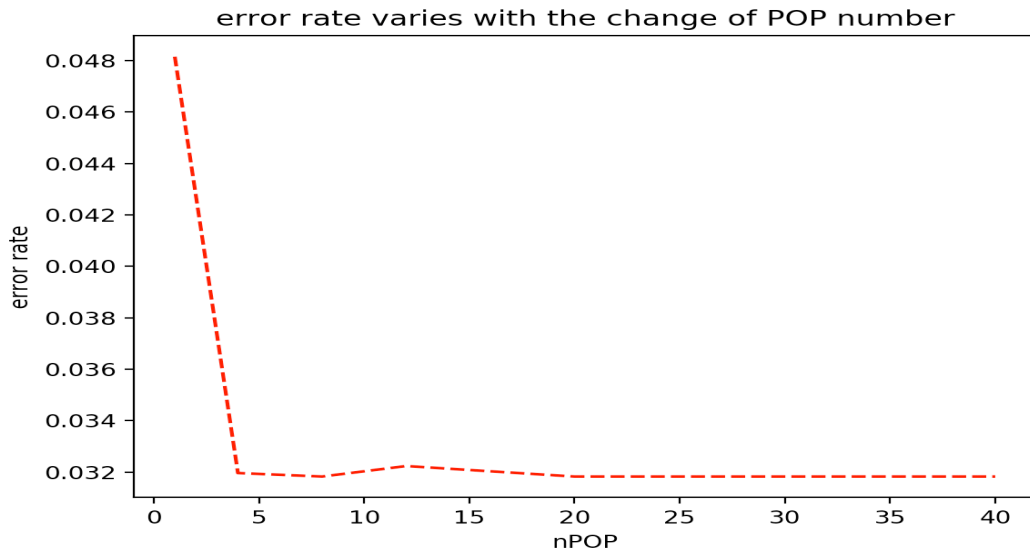
when parameter nPOP = 1 and maxiter = 1, 10-time-10-fold accuracy: 0.9514289965986394

**Figure 2: Setup experiment result using 10-10**

### 4.Parameter investigation for the PSO algorithm:

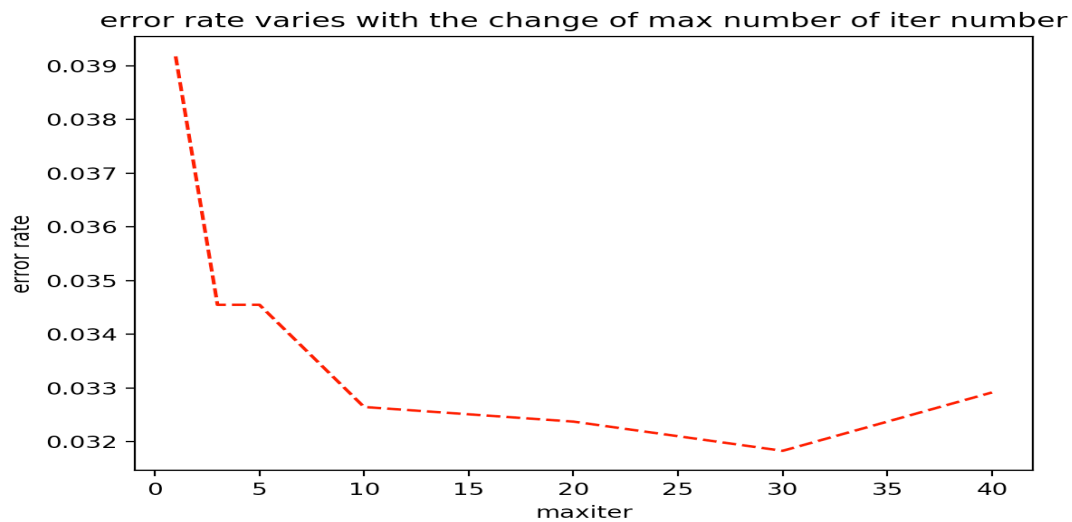
**Number of particles:** The number of particles needs to be sufficient to explore the space to find the global solution. In this experiment, the effect of the number of particles on the misclassification rate based on the 10-fold cross validation was investigated when the number of particles ranged from 1 to

40 particles (1, 4, 8, 12, 20, 30, 40) in my implementation and because of the computational time limited, I use the number of iterations is four. The result is shown in Figure 3.



**Figure 3: Parameter investigation for number of particles**

**Number of iterations:** The number of iterations also affects the performance of the proposed model. In this section, the effect of the number of iterations on the misclassification rate and computational time of the proposed algorithm was tested when the number of iterations was ranged from 1 to 40 particles (1, 3, 5, 10, 20, 30, 40) in my implementation. The misclassification rate algorithm is shown in Figure 4.



**Figure 4: Parameter investigation for number of iterations**



## 5.Choosing the best parameters for PSO algorithm:

From the Parameter investigation experiment based on ten-fold cross validation, we can see that after the number of particles reached around 4, the error rate almost keeps stable. And with the increase of number of particles, the running time of PSO-KNN increases too, I will choose nPOP = 4 for the following experiment; Furthermore, with the increase of number of iterations in the PSO-KNN, the error rate is lowest when the number of iterations is around 30. Because the number of iterations chosen in my implementation, the result is not good enough as the nPOP result. However, we can see that when the number iteration increases from 10 - 40 the error rate does not change a lot, and there are only some fluctuations; Therefore, I choose max iteration for PSO-KNN is 30.

## 6.Performance on test dataset for PSO\_KNN and KNN:

After choosing the best parameters for PSO and use those parameters to train our PSO\_KNN classifier, then we can do our experiment on the test dataset. Besides, we also do the comparative experiments on the test dataset too, which is mentioned in the proposed solution part. And for both algorithms, we also draw the confusion matrix, which can clearly reflect the performance. The experiment result is shown in Figure 5.

PSO_KNN accuracy: 1.0	Comparative Experiment accuarcy: 0.987784187309128
PSO_KNN confusion matrix:	Comparative Experiment confusion matrix:
[[496 0 0 0 0 0]	[[495 0 1 0 0 0]
[ 0 471 0 0 0 0]	[ 0 471 0 0 0 0]
[ 0 0 420 0 0 0]	[ 1 0 419 0 0 0]
[ 0 0 0 491 0 0]	[ 0 0 0 475 16 0]
[ 0 0 0 0 532 0]	[ 0 0 0 18 514 0]
[ 0 0 0 0 0 537]]	[ 0 0 0 0 0 537]]

**Figure 5: PSO\_KNN and KNN(k=5) experiment results on the test dataset**

Note: the first five step source code is in project.py, while the sixth step source code is in project\_test.py.

## Discussion and Conclusions

Seen from Figure 3 and Figure 4, we can get that when we increase the number of particles to search the space, the error rate decreases first until it reaches to 4, after which the error rate almost keep stable. And our test result on the max iteration show almost the same trends; however, because my iteration range is not large enough, it is not clearly to reflect the influence of max iteration. This part should be improved in the future.


Seen from the Figure 5, we can see that PSO\_KNN has a 100% accuracy while the KNN (k=5) achieve an around 98.8% accuracy on the test dataset. We can conclude that the optimization method that using cross validation to choose the best parameters of PSO algorithm, then using the PSO algorithm to choose k, truly has a very good optimization result.

For the future work, to improve this PSO\_KNN model, we need to do the following points:

1. Do more experiments on different datasets to see performance, for example human activity recognition (HAR) that based on signals extracted by wearing a smartphone on the chest, legs, arms, etc.
2. In my project, I focus more on the accuracy. To improve, we also need to do experiments on other aspects, for example computation time etc. Then we should make a tradeoff between the computation time and accuracy, which could lead to a different result.
3. Because of the space limitation, the number of comparative algorithms chosen in this experiment is only one. To have a more comprehensive result, we should do comparative experiments on more different algorithms, for example GA\_KNN or ABCO\_KNN.
4. In my project, I only choose the PCA method to preprocessing our dataset to reduce the running time. To achieve a better performance on running time, we could do more experiments on different data preprocessing method, e.g. LDA
5. Because of the time limit of the project, for choosing the PSO algorithm best parameters, I only do test on max iteration range I choose is 1 – 40. And the test results do not have a very clear test result; Therefore, to better illustrate the influence of the max iterations on PSO algorithm, we should increase the max iteration range, e.g. 1- 100.

## PSO-KNN Implementation

```
135 def PSO(Data, parameters):
136     # Initialization
137     GlobalBest = [-1, float("inf")] #[position, cost]
138     Population = []
139     for i in range(parameters.nPOP):
140         num_neighbors = np.random.randint(1, parameters.max_num_neighbors)
141         velocity = 0
142         test_res = cost_func(Data.attribute_train, Data.label_train, Data.attribute_test, Data.label_test, num_neighbors)
143         Best_pos = num_neighbors
144         Best_test_res = test_res
145         particle = particle_class(Position = num_neighbors, Velocity = velocity, test_tool = test_res, Best = [Best_pos, Best_test_res])
146         if (particle.Best[1] < GlobalBest[1]):
147             GlobalBest[0] = particle.Best[0]
148             GlobalBest[1] = particle.Best[1]
149         Population.append(particle)
150     GlobalBests = [GlobalBest] #hold the best globalbest at each iteration
151     #main loop of PSO
152     for i in range(parameters.MaxIt):
153         #print(i)
154         for j in range(parameters.nPOP):
155             #update velocity of ith particle
156             Population[j].Velocity = parameters.w * Population[j].Velocity + parameters.c1 * np.random.uniform(0,1) * (Population[j].Best[0] - Population[j].Position) + parameters.c2 * np.random.uniform(0,1) * (Population[j].Best[1] - Population[j].Position)
157
158             #apply velocity limit
159             Population[j].Velocity = max(Population[j].Velocity, parameters.min_vel)
160             Population[j].Velocity = min(Population[j].Velocity, parameters.max_vel)
161             #print("velocity: ", Population[j].Velocity)
162             # update position of ith particle
163             Population[j].Position = int(Population[j].Position + Population[j].Velocity)
164             #apply upper and lower bound
165             Population[j].Position = max(Population[j].Position, parameters.min_num_neighbors)
166             Population[j].Position = min(Population[j].Position, parameters.max_num_neighbors)
167             #print("position: ", Population[j].Position)
168             ##update cost of ith particle
169             test_res = cost_func(Data.attribute_train, Data.label_train, Data.attribute_test, Data.label_test, Population[j].Position)
170             Population[j].test_tool = test_res
171             # update personal best of ith particle
172             if (test_res < Population[j].Best[1]):
173                 Population[j].Best[0] = Population[j].Position
174                 Population[j].Best[1] = test_res
175
176             # update global best of ith particle
177             if (Population[j].Best[1] < GlobalBest[1]):
178
179                 GlobalBest[0] = Population[j].Best[0]
180                 GlobalBest[1] = Population[j].Best[1]
181
182             # store the best cost
183             GlobalBests.append(GlobalBest)
184             parameters.w = parameters.w * parameters.w_damp
185     return Population, GlobalBests, GlobalBest
```

 IDE and Plugin Updates  
PyCharm is ready to update.

## References:

1. Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. A Public Domain Dataset for Human Activity Recognition Using Smartphones. 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013. Bruges, Belgium 24-26 April 2013.
2. Smartphone-Based Human Activity Recognition, Ortiz, Jorge Luis Reyes, author. 2015
3. L. Chen, J. Hoey, C.D. Nugent, D.J. Cook, Z. Yu, Sensor-based activity recognition. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 42, 790–808 (2012)
4. Alaa Tharwat, Hani Mahdi, Mohamed Elhoseny, Aboul Ella Hassanien, Recognizing Human Activity in Mobile Crowdsensing Environment using Optimized k-NN Algorithm, Expert Systems With Applications (2018), doi: 10.1016/j.eswa.2018.04.017
5. Preece, S. J., Goulermas, J. Y., Kenney, L. P., Howard, D., Meijer, K., & Crompton, R. (2009). Activity identification using body-mounted sensors—A review of classification techniques. *Physiological measurement*, 30(4), R1.
6. Luts, J., Ojeda, F., Van de Plas, R., De Moor, B., Van Huffel, S., & Suykens, J. A. (2010). A tutorial on support vector machine-based methods for classification problems in chemometrics. *Analytica Chimica Acta*, 665(2), 129–145.
7. Montori, F., Bedogni, L., Di Chiappari, A., & Bononi, L. (2016). Sensquare: A mobile crowdsensing architecture for smart cities. In *Ieee 3rd world forum on internet of things (wf-iot)* (pp. 536–541). IEEE.
8. Reinhardt, A., Christin, D., & Kanhere, S. S. (2013). Predicting the power consumption of electric appliances through time series pattern matching. In *Proceedings of the 5th acm workshop on embedded systems for energy-efficient buildings* (pp. 1–2). ACM.
9. Yao, J., Kanhere, S. S., & Hassan, M. (2008). An empirical study of bandwidth predictability in mobile computing. In *Proceedings of the third acm international workshop on wireless network testbeds, experimental evaluation and characterization* (pp. 11–18). ACM.
10. Yamany, W., Fawzy, M., Tharwat, A., & Hassanien, A. E. (2015a). Moth-flame optimization for training multi-layer perceptions. In *Computer engineering conference (icenco), 2015 11th international* (pp. 267–272). IEEE.
11. Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons, Second Edition.
12. Tharwat, A., Ghanem, A. M., & Hassanien, A. E. (2013). Three different classifiers for facial age estimation based on k-nearest neighbor. In *Proceedings of 9th ieee international computer engineering conference (icenco), giza, egypt, dec. 28-29* (pp. 55 - 60).
13. Islam, M. J., Wu, Q. J., Ahmadi, M., & Sid-Ahmed, M. A. (2007). Investigating the performance of naive-bayes classifiers and k-nearest neighbor classifiers. In *Proceedings of international conference on convergence information technology* (pp. 1541–1546). IEEE.
14. Lachenbruch, P. A., & Mickey, M. R. (1968). Estimation of error rates in discriminant analysis. *Technometrics*, 10(1), 1–11.

15. Stone, M. (1978). Cross-validation: a review 2. *Statistics: A Journal of Theoretical and Applied Statistics*, 9(1), 127–139.
16. Fix, E., & Hodges Jr, J. L. (1951). Discriminatory analysis-nonparametric discrimination: consistency properties. Technical Report DTIC Document.
17. Yang, J., Zhang, D., Frangi, A. F., & Yang, J.-y. (2004). Two-dimensional pca: a new approach to appearance-based face representation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1), 131–137.
18. Gaing, Z.-L. (2004). A particle swarm optimization approach for optimum design of pid controller in avr system. *IEEE Transactions on Energy Conversion*, 19(2), 384–391.
19. Kao, C.-C., Chuang, C.-W., & Fung, R.-F. (2006). The self-tuning pid control in a slider–crank mechanism system by applying particle swarm optimization approach. *Mechatronics*, 16(8), 513–522.
20. Heppner, F., & Grenander, U. (1990). A stochastic nonlinear model for coordinated birdflocks. American Association For the Advancement of Science, Washington, DC (USA).
21. Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACMSiggraph Computer Graphics*, 21(4), 25–34.
22. Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the 6th international symposium on micro machine and human science* (pp. 39–43). New York, NY (vol. 1).
23. Kennedy, J. (2010). Particle swarm optimization. In *Encyclopedia of machine learning* (pp. 760–766). Springer.