Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 140 (2018) 324–333

Procedia Computer Science

www.elsevier.com/locate/procedia

Complex Adaptive Systems Conference with Theme: Cyber Physical Systems and Deep Learning, CAS 2018, 5 November – 7 November 2018, Chicago, Illinois, USA

Using Accuracy Measure for Improving the Training of LSTM with Metaheuristic Algorithms

Tarik A. Rashid[a,b]* Polla Fattah[a,b] and Delan K. Awla[a]

[a]*Computer Sceince and Engieering Department, University of Kurdistan Hewler, Hewler, Kurdistan, Iraq.*
[b]*Software and Infroamtics Engineering, Salahaddin University-Erbil, Hewler, Kurdistan, Iraq.*

Abstract

Recurrent Neural Networks (RNNs) are possibly the most prevailing and advantageous type of neural network. On the other hand, these networks still have some weaknesses in terms of learning speed, error convergence, and accuracy due to long-term dependencies, which need to be solved. Long-term dependencies are mainly exploding and vanishing gradients through Back Propagation Learning Algorithm. In this paper, Long Short Term Memory or LSTM is used and well structured for resolving the above concerns. Four different optimizers based on Metaheuristic Algorithms are chosen to train LSTM (these are; Harmony Search (HS), Gray Wolf Optimizer (GWO), Sine Cosine (SCA), and Ant Lion Optimization algorithms (ALOA). The suggested representations are used for classification and analysis of real and medical time series data sets (Breast Cancer Wisconsin Data Set and Epileptic Seizure Recognition Data Set). Classification accuracy measure has been used instead of error rate and mean square error methods to train LSTM with above optimizing algorithms. The experimental results are verified using the 5-fold cross validation. Details of simulations and coding in R programing language can be obtained in the following link "https://github.com/pollaeng/rnn".

Keywords: Machine Learning; Deep Learning; LSTM, Nature Inspired Algorithms, Metaheuristics; Optimization; Medical Time Series

* Corresponding author. Tel.: +964(0)-750-857-8811 Ext: 178
E-mail address: tarik.ahmed@ukh.edu.krd

10.1016/j.procs.2018.10.307

1. Introduction

Nowadays, standard neural networks or simply NNs gain great responsiveness from scholars across the globe because of their inspiring possessions, for instance, flexibility for simplification and training capacity. Back-propagation or BP is universally employed for the learning process of NNs. BP is primarily relying on the approach of gradient-descent for shrinking the calculated error or so called the mean squared error, which is resulted from the difference between the concrete outputs from the neurons at the output layer of the neural networks and the desired target. Alternatively, RNNs have been designed and adapted originally to address the problem of the simplicity and fixed architecture. The overall architecture of RNN has the cyclic relations that feed the activation from pervious time step as input to the network in order to make a decision for the current input. The time step activation from previous input is accumulated in the internal state of the network. As a result, RNN utilizes the changing of the background window dynamically of all sequence history instead of permanent window size over the sequence. The RNN capability is better for sequence modeling tasks, for example, sequence labeling and sequence prediction tasks. Both NN and RNN trained with BP, however, training RNN gradient-based back propagation through time is difficult because of the gradient fading and gradient exploding struggles. This inconvenience will limit the ability of RNN to form the long-range context dependencies between the ranges 5 -10 discrete time step from single appropriate input to output. Therefore, LSTM as an elegant architecture was proposed and designed by [1, 2, 3] to address and solve the above problems of RNN. On the other hand, some flaws make BP less attractive when it comes to learning due to its shortcomings as listed by Yi et. al. [4]; demanding differentiable transferal function for all NN neurons and the risk towards congregating and plunging into local minima, long training time hence slow convergence and very sensitive to the initial settings of the network. As a result, the main contributions of this paper are adapting LSTM network to operate as a cost function for metaheuristic algorithms, so that instead of using BP, metaheuristic algorithms are used to train LSTM network through finding optimum values for its bias and weights. In this paper, multiple metaheuristic methods, such as, Harmony Search, Ant-lion optimization Algorithm, Sine Cosine Algorithm, and Grey Wolf Optimizer are used for training LSTM (the models are symbolized as LSTMHS, LSTMALOA, LASTMSCA and LSTMGWO).

This paper is organized as follows: Details about classifiers are presented in section 2. In section 3, metaheuristic optimizers are explained in brief. Details about the process of data collection are described in section 4. In section 5, simulation and results are presented, and finally, the key conclusion ideas are presented.

2. Recurrent Neural Network Classifiers

Two different recurrent neural network structures are used namely; RNN and LSTM networks. These two networks are discussed in detail in the following subsections.

2.1 RNN

RNN is one of the models that are influential in sequential data. The methods of RNN are trained as end to end such that classification of temporal connectionist that makes it feasible to train RNNs specially for embarking upon the sequence labeling problems where the input and output configuration are unknown. Furthermore, RNNs capably are employed to undertake very specific tasks, such as, speech recognition [5], language modeling [6, 7], and machine translation [8]. The deployment of RNN is by succeeding a sequence of procedure of random size through frequently putting on transition functions to neurons in the hidden layer (internal state) via which a distinct representation of each input sequence is represented. Equation (1) is expressed as state-to-state transition function, where the work of art element that is nonlinearity with an affine transformation to x_t, which is the current input sample and h_{t-1}, which is the previous time step of the activations of hidden neurons:

$$h_t = \emptyset(Wx_t, Uh_{t-1}) \tag{1}$$

Where, *W* specifies the weight matrix between hidden and input layers, *U* specifies the weight matrix between hidden and hidden recurrent; ∅ represents activation functions, and h_t, is the current time step of the activations of hidden neurons.

2.2 LSTM

One of the improved types of RNN is LSTM network that is capable of order learning dependence in sequence prediction problems. LSTM architecture network is the improved RNN architecture with the intention of implementing suitable BP training method. As mentioned above, that LSTM model was originally generated for undertaking fading gradient that is mostly prevalent in standard RNN [1]. The vanishing or fading gradient accrues when the pervious gradient shares a problem over time due to declining or growing error through weight updating repeatedly. As a result, the path temporal evolution will connect with all error signals, and it will exponentially flow back through time according to the weights magnitude. Therefore, RNN model was unsuccessful when long time delay happens in relation to appropriate desired and input actions. Nevertheless, LSTM procedure will reduce the problem through implementing persistent error movement over persistent error carrousels via superior components, allowing for not collapsing error stream "back into time". Hochreiter and Bengio [1, 2, 3] had known the obstacles that capture long-term dependencies for training RNN for a long time. However, there have been some successful approaches to this essential challenge, for example, modifying the state-to-state function transition in order to encourage the hidden neurons to organize long-term memory, and generate paths in the time-unfolded RNN. LSTM was developed and the intention for this technique was to address the problem of capturing the long-term dependencies. Since launching of LSTM in 1997 [9], several versions of LSTM have been released as in [10, 11]. LSTM technique replaces hidden neurons architecture of sigmoid or tanh functions by memory cells as multiple gates manage their input and outputs. These gates manage information flow to hidden neurons. LSTM suffers from high complexity in the hidden layer. LSTM has four times more parameters than conventional RNN. Other methods have been attempted to resolve the problem of long-term dependencies, such as, Grid LSTM and Multi-dimensional networks [9, 12, 13]. Zaremba and Sutskever [11] delivered the latest LSTM modification. Here, C_t are the units of memory cell in LSTM, i_t represents the input gate, f_t represents the forget gate and lastly o_t *represents* the output gate as shown in Fig (1).

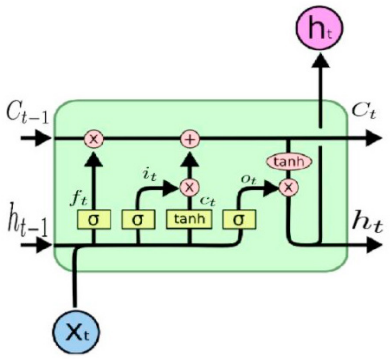

Figure 1: The Structure of LSTM.

It is imperative to determine the information that is going to be disposed from the cell state. This kind of verdict is done via the forget gate by means of a sigmoid function as shown in the below equation:

$$f_t = \sigma\left(U_f \cdot h_{t-1} + W_f x_t + b_f\right) \quad (2)$$

Where f_t is the forget layer, σ represents sigmoid activation function in the forget gate, U_f represents weight connection to previous hidden states, W_f is weight connection to the input patterns, and b_f is the biases at the forget layer. It is also essential to determine a new information that can be stored in the cell state. This process is conducted

by two sections: The input gate layer, which determines the values that will be updated and temporary cell state layer, which generates new candidate values in a form of vector that could be added to the state as shown in equations (3) and (4):

$$i_t = \sigma\left(U_i \cdot h_{t-1} + W_i x_t + b_i\right) \tag{3}$$

Where i_t is the input layer, σ represents sigmoid activation function in the input gate, U_i represents weight connection to the previous hidden states, W_i is weight connection to the input patterns, and b_i is the biases at the input layer.

$$\tilde{C}_t = tanh\left(U_c \cdot h_{t-1} + W_c x_t + b_c\right) \tag{4}$$

Where $\tilde{C}_t$ is the temporary state layer, σ is the sigmoid function at the temporary state layer, U_C is the weight connection to the previous hidden states, W_C is weight connection to the input patterns, and b_C is the biases at the temporary state layer.

Next, the new cell state is modified from the old cell state via the following equation:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{5}$$

Thus, the output layer gate will be decided using equation (6):

$$o_t = \sigma\left(U_o \cdot h_{t-1} + W_o x_t + b_o\right) \tag{6}$$

Where o_t is the output layer, σ is the sigmoid function at the output layer gate, U_o is the weight connection to previous hidden states, W_0 is weight connection to the input patterns, and b_0 is the biases at the output layer.

At the end, the output of the LSTM can be determined using equation (7):

$$h_t = o_t * \tanh(C_t) \tag{7}$$

In addition to that, the memory cell of LSTM, input, output and forget gate are adaptively expose, forget, and memorize the memory content. In case the memory content are considered important in the memory unit, then the forget gate will be cancelled and the memory content will be carried to various time steps, and that is when the long-term dependency will be captured. Furthermore, the memory unit may decide to delete the memory content by opening the forget gate.

3. Metaheuristic Optimizers

In this paper, four different types of optimizers from metaheuristic algorithms are chosen to train the weights and biases of LSTM in all its layers; each optimizer is explained briefly as follows:

3.1 HS

Harmony search method can easily handle the distinct and continuous variables. Nevertheless, many applications used HS algorithm especially in engineering and industry fields, however, there has been a noticeable improvement in the HS applications in the past. The concept of HS algorithm depending on progress of improvisation of music performers that play music instruments in a band [14]. The improvisation process takes place when performers play and experiment notes on their instruments where the consequential tones are measured to be the harmony according to aesthetic quality measure. For a new HS to be improved, each performer performs new random notes other than the

collection of approved notes, or performs already existed notes in recollection, or performs notes existed in memory, but is somewhat adapted. In the end, musicians use the only good improved harmonies until the new of better result is detected for replacing with the worst one.

3.2 ALOA

The ALOA is a meta-heuristic that is inspired by antlions and ants where the physical interaction modeled mathematically. The ALOA is originally modified to confront the optimization issues considering the ants' actual movement when catching prey. In this new approach, the search agents are both ants and ant-lions where they try to build a trap step by step to hunt the prey. The steps of the agents are random walk of ants, construct traps, set-up trap for ants, grab or catch the prey [15].

3.3 SCA

The structure of the SCA algorithm is to generate several initial random solutions and then entail them to swing towards the outside or heading for the best candidate solution through mathematical models using functions of sine and cosine. In addition, previous results of SCA algorithm proved that it can manage to explore several different regions and places of the search space, avoid local optima, converge to the global optimum, and develop the regions that were selected during optimization of the search space effectively. The SCA algorithm can be very deductive in solving a real problem with unknown search spaces [16].

3.4 GWO

The major idea behind this algorithm is simulating the performance of the grey wolves live in a grope [17]. The grey wolf has serious hierarchy, the main stage is the leader known as Alpha that is responsible for determining and making choices for the pack. The second stage of the hierarchy in the pack of grey wolf is juniors or Beta. Beta wolves are helping the alpha in other activities such as decision-making. The third stage of this pack hierarchy is Delta; this part is consisting of sentinels, scouts, hunters, elders, and caretakers. First, the scouts will conduct inspecting the borders of the regions and provide the awareness to wolves when any dangers occur. Second, the sentinels will provide the shelter and plan of wolves. Third, the elders or knowledgeable wolves are Alpha and Beta. Forth, the hunters will assist the alpha and beta in hunting and they provide nourishment for the wolves, and finally, the caretakers assist fragile and ill wolves. The lowest stage of the group is omega, where they are relying and depending on all other main wolves. The capacity of grey wolves is remembering the locations of the victim to surround them.

4. Data Collection

The process of data collection is gathering information and measuring them according to the important variables on hand. In this research work, two data sets are used, they are real data sets created and analyzed from different places, these are; Breast Cancer Wisconsin (Diagnostic), which is provided via the University of Wisconsin and Epileptic Seizure Recognition from Rochester Institute of Technology, which is a time series data set type. All the datasets are available at the University of California, Irvine UCI, fetched from UCI Web Portal and downloaded from UCI repository.

4.1 Breast Cancer Wisconsin Data Set

The Breast Cancer Wisconsin data set (BCWDS) is more than 569 records in total used for training purpose [18]. The number of attributes in the dataset is 32. The attributes are: ID, Diagnosis, and 30 real-valued input feature, where the ID attribute expressing the id number of the patient and diagnosis attribute is used for classification purpose where (M = malignant, B = benign). Furthermore, the pre-processing technique is also adapted to produce worthy data intended at further data mining process.

4.2 Epileptic Seizure Recognition Data Set

The second data set is a time series data set, which is called Epileptic Seizure Recognition Data Set (ESRDS), which has been collected by Qiuyi Wu and Ernest Fokoue from School of Mathematical Sciences, Rochester Institute of Technology [19].The data set consists of continuous record of 500 individuals' brain activity for 23.6 seconds. From each second of this contentious data 178 data points of equal intervals are selected to transform the whole EEG recordings for the mentioned time into 4097 consequent recordings. Qiuyi et al., took EEG recordings of brain activity for various individuals with different recording situations so that five different classes are created representing each situation and individual's health condition as follows:

1) Class 1: Represents brain activity recordings of patients diagnosed with epileptic seizure.
2) Class 2: Represents recordings of the EEG for patients, which are diagnosed with a tumor in their brain and the recording is made from the area where the tumor was located.
3) Class 3: Represents brain activity recordings of the EEG for patients, which are diagnosed with a tumor in their brain, and the recording is made from a healthy area other than the location of the tumor.
4) Class 4: Represents brain activity recordings of normal healthy persons with their eyes closed at the recording time.
5) Class 5: Represents brain activity recordings of normal healthy persons with their eyes opened at the recording time.

As it can be noticed only individuals in class 1 have epileptic seizure so that for this study purpose we have used a binary classifier model by representing class 1 as positive and class 2 as negative result.

5. Simulations and Experiments

The database needs to be simplified before processing, the clarification can be accomplished by raising the most important and relevant questions and that are first, is the data available for the task? Second, is the data applicable? Finally, yet importantly, is the available data relevant? The ESRDS from the original data set was fetched initially from UCI university web portal. Then, the pre-processing is conducted on the data set, where the value of the first column that indicates the patient information and id number contains more than one value, which was combined with other attribute values and shows a complex view of the attribute. To crack this problem, we divided the column into three attributes and remove the unneeded value to make the data set clear and ready to process. The experiments have been performed on two data sets BCWDS with 699 records and ESRDS with 4097 data point using 12 computers. The training procedures of weights and biases are conducted with the proposed approaches (see Fig 2). Details of simulations and coding can be obtained in the following link: "https://github.com/pollaeng/rnn".

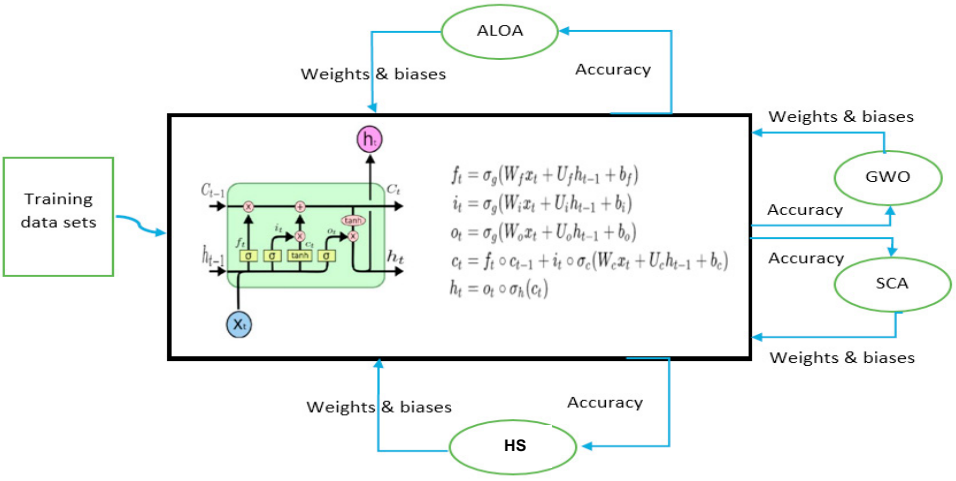

Figure 2: LSTM with four Training optimizers.

The test datasets are examined with the accuracy of global models. The overall results of basic comparison clearly show that accuracy, training time, communication overhead and other parameters have been optimized. The work of this research is conducted on several computers to test the accuracy of each algorithm with LSTM. The following software configurations are used: the programing language is R; the software program is R studio. As mentioned earlier, LSTM trained with four optimizers (GWO, SCA, HS, ALOA) are proposed; these are LSTMGWO, LSTMSCA, LSTMHS, and LSTMALOA. Both data sets; BCWDS and ESRDS on each model are used for performance evaluation. The main purpose of these experiments is to compare all models on LSTM neurons comparatively; same parameters, such as, no. of hidden neurons, population, and iteration for each model are chosen (See Tables 1 and 2). The overall model is small enough just to avoid overfitting, which can easily distract the comparison. The training of the entire network has been completed on BCWDS with 699 records about 90 hours of testing, while ESRDS with 4097 record tested nearly 100 hours.

Table 1. Breast Cancer Wisconsin Data set.

Algorithms trained on BCWDS	Iteration	Population	No. Hidden Neurons
LSTMGWO	50	50	50
LSTMSCA	50	50	50
LSTMHS	50	50	50
LSTMALOA	50	50	50

Table 2. Epileptic Seizure Recognition Data Set

Algorithms trained on ESRDS	Iteration	Population	No. Hidden Neurons
LSTMGWO	30	30	30
LSTMSCA	30	30	30
LSTMHS	30	30	30
LSTMALOA	30	30	30

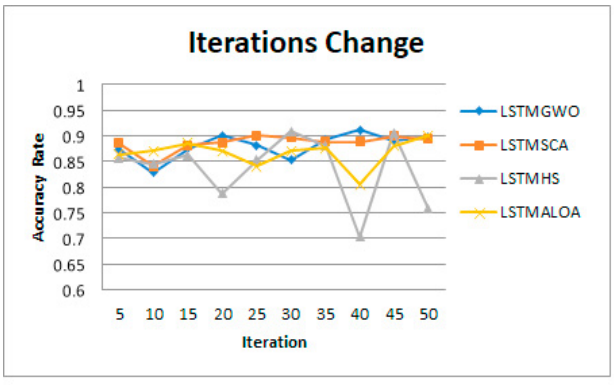

Fig. 3. Accuracy rate of LSTM network for classifying BCWDS, trained by various metaheuristic search algorithms using different iteration numbers. The Population size is fixed at 30 for all algorithms.

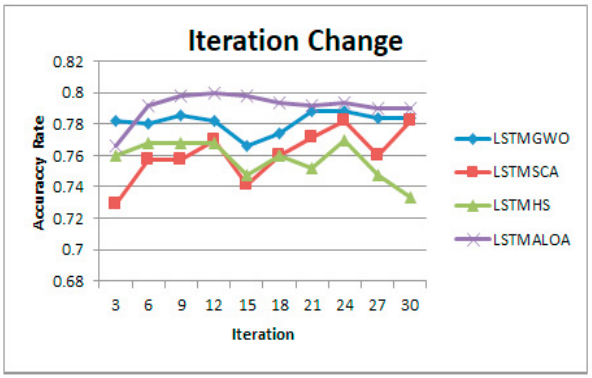

Fig. 4. Accuracy rate of LSTM network for classifying ESRDS, trained by various metaheuristic search algorithms using different iteration numbers. The Population size is fixed at 30 for all algorithms.

Figures 3 and 4 show the accuracy of the classifier for five folds of train/tests for both data sets, namely; breast cancer and epileptic seizure. In these two settings, we test different number of maximum iteration numbers for searching algorithms to find the optimum solution. The iterations start from five and end at fifty with five steps different each time (see Figure 3). For all these tests, the number of population and the number of neurons in the hidden layer remained constant for the sake of comparison. Both figures show slight increase of accuracy with the increased number of iterations. However, for the ESRDS (the iterations start from three and end at thirty with three steps different each time, See Figure 4), grater difference can be seen by using different search algorithms with LSTMALOA leading the accuracy and LSTMSCA being the lowest.

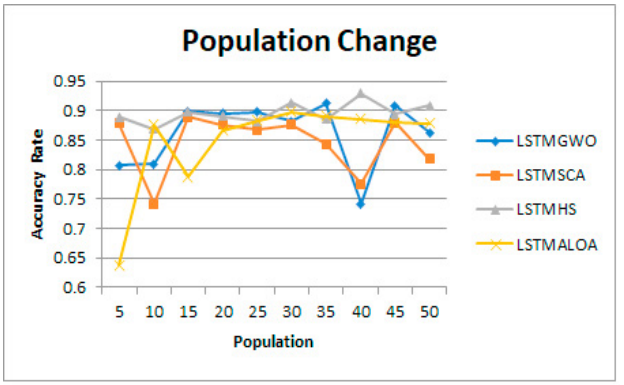

Fig. 5: Accuracy rate of LSTM network for classifying BCWDS, trained by various metaheuristic search algorithms using different population sizes. The iteration number is fixed at 30 for all algorithms.

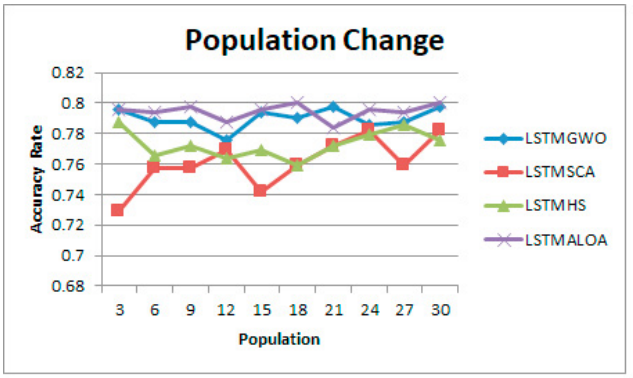

Fig. 6: Accuracy rate of LSTM network for classifying ESRDS, trained by various metaheuristic search algorithms using different population sizes. The iteration number is fixed at 30 for all algorithms.

Figures 5 and 6 show the accuracy of the classifier with the same setting of five folds for the data sets. In these two settings, we test different number of population numbers for metaheuristic searching algorithms to find the optimum solution. The populations start from five and end at fifty. Again, the number of iterations and the number of neurons in the hidden layer remained constant for the sake of comparison. While all algorithms perform better with BCWDS than ESRDS. Nevertheless, both figures show a little or no change of average accuracy while the number of population increases. With one surprising exception that the accuracy of LSTMSCA algorithm increases steadily with the increase of the search population.

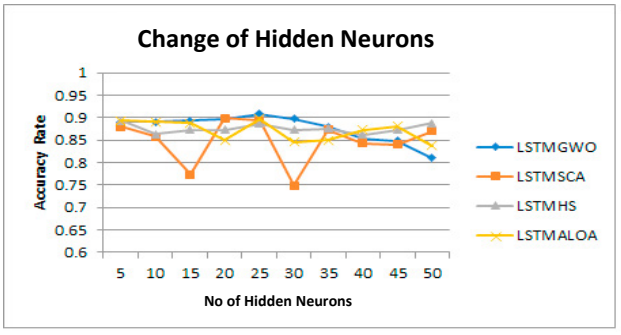

Fig. 7. Accuracy rate of LSTM network for classifying BCWDS, using different number of neurons in the hidden layer. The iteration number and population size for the search algorithms are fixed at 30 for all algorithms.

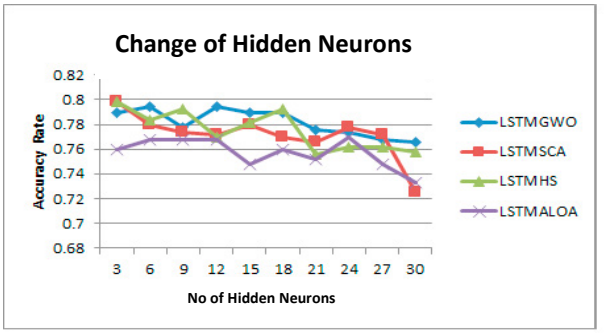

Fig. 8. Accuracy rate of LSTM network for classifying ESRDS, using different number of neurons in the hidden layer. The iteration number and population size for the search algorithms are fixed at 30 for all algorithms

Figures 7 and 8 show the accuracy of the classifiers with various number of neurons in the hidden layer. Accordingly, in this setting we test various changes in the structure of the neural networks rather than changing the search algorithm parameters. So that the number of iterations and populations remained constant throughout this test. For the BCWDS (Figure 7) the accuracy of algorithms does not change with the different number of neurons in the hidden layer. However, surprisingly, the accuracy of the algorithms on ESRDS declines with increasing the number of neurons in the hidden layer. This might be due to the increased complexity in the network's structure, which requires more training time to converge compared with simpler data set of BCWDS. The results were as follows: the models trained on BCWDS produce 86.67 % as average accuracy for the classification. Whereas the models trained on ESRDS, produce 77.14 % as average accuracy for the classification.

We have used Friedman ranked test to determine either the results of using different algorithms with the change of population, iteration, and hidden layer size can affect the accuracy results. The Friedman Test is a statistical non-parametric ranked test, which can treat multiple dependent samples sets. The null hypothesis of the Friedman Test is that there is no difference between variables. The null hypothesis cannot be rejected if the result of the treated testes is higher than the pre-appointed significance value [20].

The p-value of Friedman test for breast cancer data set with variation of the population, iteration and hidden layer size are respectfully 0.023, 0.323 and 0.136, which means null hypothesis is true for the tests of variable iteration and hidden layer size. This indicates that different algorithms produce similar results for these two variables. While for the results of the temporal data set the p-value for all three tests are too low which can be rounded to zero, hence we can reject the null hypothesis. This indicates that the different algorithms produce significant different accuracies while classifying the underlying data set.

6. Conclusion

In this paper, problems of RNN architecture are indicated and LSTM architecture is used as an alternative to RNN to deal with problems of learning speed and accuracy due to long-term dependencies. Instead of BP, this paper uses four different optimizers based on metaheuristic algorithms such as Harmony search, Ant-lion optimization, Sine Cosine, and Grey Wolf Optimizer to train LSTAM. Two data sets are used to examine all models. Same parameters are used for all models. These experimental results were verified using the 5-fold cross validation. The following key points are concluded:

1) When the number of population and the number of neurons in the hidden layer kept fixed. Minor increase of accuracy with the greater than before number of iterations was noticed on both data sets. The larger transformation can be seen by using different search algorithms with LSTMALOA leading the accuracy and LSTMSCA being the lowest.
2) When the number of iterations and the number of neurons in the hidden layer kept fixed, every algorithm performed better with BSWDS than ESRDS. Although, a little or no change of average accuracy, while the number of population increases. With one unanticipated exclusion in which the accuracy of LSTMSCA algorithm increases gradually with the increase of the search population.
3) In addition, when the number of iterations and populations kept fixed. The accuracy of algorithms does not change with the different number of neurons in the hidden layer on BCWDS. On the other hand, the accuracy of the algorithms on ESRDS remarkably declines with increasing the number of neurons in the hidden layer; hence, there is a negative correlation between the number of neurons in the hidden layer and the accuracy of the overall network. This might be due to the increased complexity in the network's structure, which requires more training time to converge compared with simpler data set of breast cancer.
4) The p-value of Friedman test for breast cancer data set with variation of the population, iteration and hidden layer size are respectfully 0.023, 0.323 and 0.136, which means null hypothesis is true for the tests of variable iteration and hidden layer size. This indicates that different algorithms produce similar results for these two variables. While for the results of the temporal data set the p-value for all three tests are too low which can be rounded to zero, hence we can reject the null hypothesis. This indicates that the different algorithms produce significant different accuracies while classifying the underlying data set.

Acknowledgements

The authors would like to thank the University of Kurdistan Hewler (UKH) for providing their continuous effort and support.

References

[1] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Juergen Schmidhuber (2001) "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies". In S. C. Kremer and J. F. Kolen, eds., *A Field Guide to Dynamical Recurrent Neural Networks. IEEE press.*
[2] Juergen Schmidhuber, D. Wierstra, M. Gagliolo, and F. Gomez (2007) "Training Recurrent Networks by Evolino" *Neural Computation*, **19(3):** 757-779. PDF (preprint). Compare Evolino overview (since IJCAI 2005).
[3] Juergen Schmidhuber (1997) "Discovering neural nets with low Kolmogorov complexity and high generalization capability". *Neural Networks*, **10(5):**857-873.
[4]Jiao-hong Yi, Wei-hong Xu, and Yuan-tao Chen (2014) "Novel Back Propagation Optimization by Cuckoo Search Algorithm," *The Scientific World Journal*, vol. 2014, Article ID 878262, 8 pages,. https://doi.org/10.1155/2014/878262.
[5]Alex Graves (2014) "Generating Sequences With Recurrent Neural Networks", *arXiv*:1308.0850v5 [cs.NE] 5 Jun.
[6]Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur (2011) "Extensions of recurrent neural network language model" *In Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference*, 5528–5531. IEEE.
[7]Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget and Jan Cernocky(2011) "Strategies for Training Large Scale Neural Network Language Models" *In Proc. Automatic Speech Recognition and Understanding.*
[8] Nal Kalchbrenner Phil Blunsom (2013) "Recurrent Continuous Translation Models", *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, 18-21, October 2013 Association for Computational Linguistics.
[9] Sepp Hochreiter and Juergen Schmidhuber (1997) "Long Short-Term Memory" *Neural Computation* **9(8):**1735-1780, 1997.
[10]Felix A. Gers, Juergen Schmidhuber, and Fred Cummins (1999) " Learning to Forget: Continual Prediction with LSTM" *Technical Report* IDSIA-01-99 January, 1999.
[11] Wojciech Zaremba, and Ilya Sutskever (2105) " Recurrent Neural Network Regularization", *arXiv*:1409.2329v5 [cs.NE] 19 Feb.
[12] Q. V. Le, N. Jaitly, and G. E. Hinton (2015) "A simple way to initialize recurrent networks of rectified linear units," *arXiv preprint arXiv:*1504.00941.
[13]T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. Ranzato (2014) "Learning longer memory in recurrent neural networks," *arXiv preprint arXiv:*1412.7753,
[14] Geem Z.W., Kim J.H., and Loganathan G.V.(2001) "A new heuristic optimization algorithm: Harmony search**"**, *Simulations*, **76**: 60-68
[15] Mirjalili S. (May 2015) "The Ant Lion Optimizer", *Advances in Engineering Software*, **83:** 80-98, DOI: http://dx.doi.org/10.1016/j.advengsoft.2015.01.010.
[16] Mirjalilim S (March 2016) "SCA: A Sine Cosine Algorithm for solving optimization problems" *Knowledge-Based Systems*, **96**: 120-133, Knowledge-Based Systems https://doi.org/10.1016/j.knosys.2015.12.022.
[17] Mirjalili S, Mirjalili S. M. and Lewis A (2104) "Grey Wolf Optimizer", *in Advances in Engineering Software*, **69**: 46-61, DOI: http://dx.doi.org/10.1016/j.advengsoft.2013.12.007.
[18] Mangasarian O. L. and Wolberg W. H. (1990) "Cancer diagnosis via linear programming", *SIAM News***, 23 (5)**: 1 & 18.
[19]Andrzejak RG, Lehnertz K, Rieke C, Mormann F, David P, and Elger CE (2001) "Indications of nonlinear deterministic and finite dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state", *Phys. Rev. E*, 64, 061907.
[20]M. Friedman (1940), "A Comparison of alternative tests of significance for the problem of m rankings," *The Annals of Mathematical Statistics*, **11(1):** 86–92.