Predicting Cervical Cancer with Metaheuristic Optimizers for Training LSTM

Andre Quintiliano Bezerra Silva(✉)

Instituto Federal de Mato Grosso do Sul, Jardim, Brazil
andre.bezerra@ifms.edu.br
http://www.ifms.edu.br

Abstract. Disease prediction can be extremely helpful in saving people, especially when we are diagnosed with cancer. Cervical cancer, also known as uterine cancer, is the fourth most frequent cancer in women with an estimated 570,000 new cases in 2018 representing 6.6% of all female cancers. In accordance with World Health Organization (WHO), the mortality rate for cervical cancer reaches 90% in the underdeveloped nations and that the high mortality rate found in it could suffer a substantial reduction if there were: prevention, effective screening, treatment programs and early diagnosis. Artificial Neural Networks (ANN) has been helping to provide predictions in healthcare for several decades. Most research works utilize neural classifiers trained with backpropagation (BP) learning algorithm to achieve cancer diagnosis. the traditional BP algorithm has some significant disadvantages, such as training too slowly, easiness to fall into local minima, and sensitivity of the initial weights and bias. In this work, we use a type of Recurrent Neural Network (RNN), known as Long Short-Term Memory (LSTM), whose main characteristic is the ability to store information in a series of temporal data. Instead of training the network with the backpropagation, the LSTM network was trained using five different metaheuristic algorithms: Cuckoo Search (CS), Genetic Algorithm (GA), Gravitational Search (GS), Gray Wolf Optimizer (GWO) and Particle Swarm Optimization (PSO). From results obtained can be observed that metaheuristic algorithms had performances above 96%.

Keywords: Machine learning · Cervical cancer · Metaheuristic algorithms · Long Short-Term Memory (LSTM)

1 Introduction

Cancer of the uterus is a type of pathology that develops in the lining of the uterus (endometrium). This type of cancer affects the female reproductive organs, located in the lower part of the uterus, near the vaginal canal [1]. According to a survey conducted by WHO [2], Cervical cancer is the fourth most frequent cancer in women with an estimated 570,000 new cases in 2018 representing

6.6% of all female cancers. Approximately 90% of deaths from cervical cancer occurred in low and middle-income countries. The proliferation of cancer cells occurs by the non-treatment of lesions that are mostly caused by the transmissible HPV virus [1].

The development of cervical cancer is usually slow and preceded by abnormalities in the cervix. However, the absence of early stage symptoms might cause carelessness in prevention. Additionally, a lack of effective screening programs aimed at detecting and treating precancerous conditions is a key reason for the much higher cervical cancer incidence in developing countries.

Several studies have been working to detect this type of cancer using different types of machines learning classifiers such as Decision Tree [3], Support Vector Machines (SVM) [4], Gaussian Naive Bayes [5] and Artificial Neural Network [6]. RNN is one of the underlying network architectures used to develop other deep learning architectures. However, RNNs can suffer from two problems: vanishing gradients or exploding gradients. The gradients carry information used in the RNN parameter update, thus, when the gradient becomes too small, the parameter updates become insignificant which means in real learning is done. The LSTM is a special type of recurring networks and was created by [7] to solved the above problems of RNN. Its popularity has grown in recent years as an RNN architecture for various applications, such as: text compression in natural language [8,9], text recognition [10,11], speech recognition [12–14], gesture recognition [15–17].

The traditional method for performing recurrent network training is Back Propagation. BP is primarily relying on the approach of gradient-descent for shrinking the calculated squared error. However, traditional training algorithms have some drawbacks such as slow speed of convergence, easy to fall into local minimum and the training process takes longtime [18].

This work intends to design an adaptation of the LSTM network in order to apply these metaheuristic algorithms to searching for optimal values for its bias and weights. In addition, the study can be used in the Oncological area, aiming to subsidize the clinical decision in patients with suspected cervical cancer.

The rest of the article is organized as follows: after this introduction, we explain the essential RNN and LSTM fundamentals. In Sect. 3, metaheuristic optimizers are explained in brief. Section 4, presented details about the process of data collection. Section 5, provides the results of the comparison followed by final remarks and future development in Sect. 6.

2 Neural Networks

2.1 RNN

Recurrent Neural Networks have the ability to receive signals from both the input layer and the hidden layer at the previous time iteration [19]. In some ways, the hidden layer simulates the operation of a memory. In addition, Recurrent Neural Networks work with sequential data on both the input layer and the output layer, fitting perfectly in the context of time series.

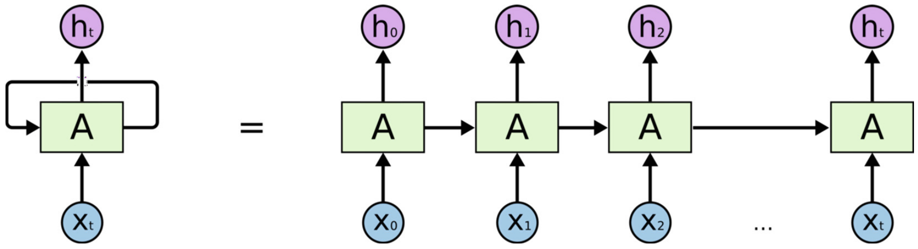

Fig. 1. Neuron of the Hidden layer of the Recurrent Neural Network [Olah, 2014].

In [20] explains that the hidden state of the Recurrent Neural Network receives information from the independent variables, as well as from its own processing result from the previous iteration, as shown in (Fig. 1). In this example, X_t is represented by the input data at time iteration t, which will be processed by the neuron A. h_t is represented by the output of neuron A in time iteration t, which can be used in the next iterations by the same neuron, as behavior. The practical effect of this is similar to the behavior of a short-term memory and is very useful in the context of time-series prediction, given that the data of the input and output layers are sequential data [21].

2.2 Long Short-Term Memory (LSTM)

Long Short-Term Memory networks are a recurrent and deep model of neural networks. It is a typical neural network architectures based on neurons and introduced the concept of memory cell. The memory cell can keep its value for a short or long time as a function of its inputs, which allows the cell to remember what is important and not only of its last computed value, that is, it can capture long dependencies range and nonlinear dynamics.

This type of network has been widely used and been able to achieve some of the best results when placed compared to other methods [23]. This fact is especially observed in the field of Natural Language Processing, and in calligraphy recognition is considered the state-of-the-art [24]. The architecture of LSTM cell is displayed in (Fig. 2).

Each LSTM block consists of a forget gate, input gate and an output gate. In Fig. 2, a basic LSTM cell with a step wise explanation of the gates is shown and on the top an other illustration of the cell connected into a network is shown. In the first step the forget gate looks at h_{t-1} and x_t to compute the output f_t (Eq. 1) which is a number between 0 and 1. This is multiplied by the cell state C_{t-1} and yield the cell to either forget everything or keep the information. In the next step the input gate is computing the update for the cell by first multiplying the outputs i_t and $\tilde{C}_t$ (Eqs. 2 and 3) and then adding this output to the input $C_{t-1} * f_t$, which was computed in the step before. Finally the output value has to be computed, which is done by multiplying o_t with the *tanh* of the result of the previous step, which can be seen in (Eqs. 5 and 6).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4}$$

$$o_t = \sigma(W_o \cdot x_t[h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t * tanh(C_t) \tag{6}$$

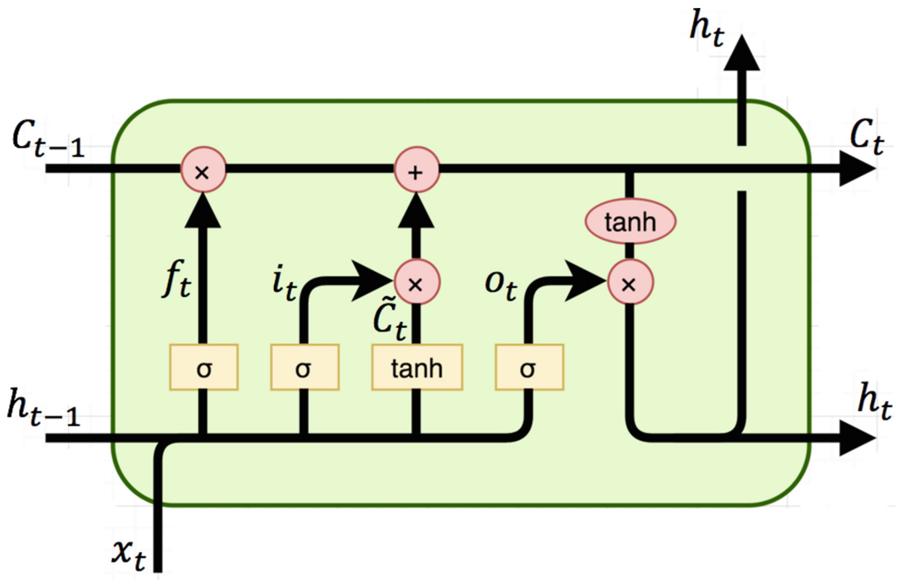

Fig. 2. Basic architectures of LSTM cells.

3 Metaheuristic Optimizers

3.1 Cuckoo Search Algorithm - CSA

The Cuckoo Search Algorithm is an evolutionary metaheuristic algorithm developed by [25], based on the behavior of the reproduction of some cuckoo bird species. In the process, these species lay their eggs in the nests of other birds. Some of these eggs, which are very similar to the eggs of the host bird, have the opportunity to grow and become adult cuckoos. This technique has been widely applied in global optimization problems, due to its simplicity and effectiveness, as well as its rapid convergence and the ability to avoid local minimums [25,26]. CS can be described using following three idealized rules: 1. Each cuckoo lays one egg at a time and dumps it in a randomly chosen nest. 2. The best nests with the high quality of eggs will carry to the next generations. 3. The number of available host nest is fixed and if a host bird identifies the cuckoo egg with the probability of pa $\in [0, 1]$ then the host bird can either throw them away or abandon them and build a new nest.

3.2 Genetic Algorithm - GA

The Genetic Algorithm is one of the most consolidated methods of computational intelligence [27]. It is inspired by the theory of evolution of species. In this case, the best solution for a given problem is found from the combination of possible solutions that are improved at each iteration. At each iteration or generation, a new population of possible solutions or individuals is created from the genetic information of the best individuals of the previous generation population. The algorithm represents a possible solution using a simple structure, called a chromosome, to which genetically inspired operators of selection, crossing and mutation are applied, simulating the evolution process of the solution. Each chromosome is made up of genes, each being represented by bits.

3.3 Gravitational Search Algorithm - GSA

GSA belongs to the class of non-deterministic algorithms based on a population of candidates for solving the optimization problem. This algorithm, as in other evolutionary approaches, is characterized by generating an initial (random) population and defining a strategy for updating the candidate population. In GSA the development of this strategy is based on the laws of gravity and movement [28]. In this context, agents are considered as objects and their performance is measured by their masses. All mass attracts and is attracted to other masses due to gravitational force, causing the overall movement of the whole set (population) considered in the problem. Thus the masses intertwine using a direct form of communication, the gravitational force. To describe the GSA algorithm, consider a system with N agents (masses), the position of the agent i is defined by:

$$X_i = (X_i^1, \cdots, X_i^d, \cdots, X_i^n), \qquad for \quad i = 1, 2, \cdots, N \tag{7}$$

where x_i^d presents the position of the agent i in the dimension d and n is the search space dimension. After evaluating the current population fitness, the mass of each agent is calculated as follows:

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{8}$$

where

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{9}$$

where $fit_i(t)$ represent the fitness value of the agent i at time t. The $best(t)$ and $worst(t)$ are the best and worst fitness of all agents, respectively and defined as follows:

$$best(t) = min_{j \in (1, \cdots N)} fit_j(t), \quad worst(t) = max_{j \in (1, \cdots N)} fit_j(t) \tag{10}$$

To evaluate the acceleration of an agent, total forces from a set of heavier masses applied on it should be considered based on a combination of the law of gravity according to:

$$F_i^d(t) = \sum_{j \in kbest, j \neq i} rand_j G(t) \frac{M_j(t) \times M_i(t)}{R_{i,j}(t) + \epsilon} (x_j^d(t) - x_i^d(t)) \tag{11}$$

where $rand_j$ is a random number in the interval [0, 1], $G(t)$ is the gravitational constant at time t, M_i and M_j are masses of agents i and j, ϵ is a small value and $R_{i,j}(t)$ is the Euclidean distance between two agents, i and j. $kbest$ is the set of first K agents with the best fitness value and biggest mass, which is a function of time, initialized to K_0 at the beginning and decreased with time. Here K_0 is set to N (total number of agents) and is decreased linearly to 1. By the law of motion, the acceleration of the agent i at time t, and in direction d, $a_i^d(t)$, is given as follows:

$$a_i^d(t) = \frac{F_i^d(t)}{M_i^d(t)} = \sum_{j \in kbest, j \neq i} rand_j G(t) \frac{M_j(t)}{||X_i(t), X_j(t)||_2 + \epsilon} (x_j^d(t) - x_i^d(t)) \tag{12}$$

Finally, the searching strategy on this concept can be described by following equations:

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \tag{13}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{14}$$

where x_i^d, v_i^d and a_i^d represents the position, velocity and acceleration of ith agent in dth dimension, respectively. $rand_i$ is a uniform random variable in the interval [0, 1]. This random number is applied to give a randomized characteristic to the search. It must be pointed out that the gravitational constant $G(t)$ is important in determining the performance of GSA and is defined as a function of time t:

$$G(t) = G_0 \times exp\left(- \beta \times \frac{t}{t_{max}} \right) \tag{15}$$

3.4 Gray Wolf Algorithm - GWO

The Gray Wolf Algorithm (GWO) is a computational optimization technique created by [29] based on the hunting behavior of gray wolves (Canis lupus). This species usually lives in packs of 5 to 12 individuals. These adopt a well-defined and narrow hierarchy. The leader of the wolves is called the Alpha (α), who is responsible for making decisions related to hunting, time, place of rest, etc. The second level is called Beta (β), which supports the Alphas making decisions. These are also strong candidates to take the lead in losing an Alpha. The lowest level of the hierarchy is occupied by Omega (ω), who play the role of scapegoat and must satisfy the whole group. The third level is occupied by the Delta (δ), responsible for the safety of the pack [30]. The grey wolves encircling behavior to hunt for a prey can be expressed as:

$$\boldsymbol{D} = \left| \boldsymbol{C} \cdot \boldsymbol{X}_p(t) - \boldsymbol{X}_{(t)} \right| \tag{16}$$

$$\boldsymbol{X}(t+1) = \boldsymbol{X}_p(t) - \boldsymbol{A} \cdot \boldsymbol{D} \tag{17}$$

where t indicates the current iteration, $\boldsymbol{A} = 2\boldsymbol{a} \cdot \boldsymbol{r}_1 - \boldsymbol{a}, \boldsymbol{C} = 2 \cdot \boldsymbol{r}_2$, where components of $\boldsymbol{a}$ are linearly decreased from 2 to 0 over the course of iterations and r_1, r_2 are random vectors in [0, 1], $\boldsymbol{X}_p$ is the prey's position vector, and $\boldsymbol{X}$ indicates the position vector of a grey wolf. The second main phase is the hunting phase and it can be modeled as:

$$\boldsymbol{D}_\alpha = \left|\boldsymbol{C}_1 \cdot \boldsymbol{X}_\alpha - \boldsymbol{X}\right|, \boldsymbol{D}_\beta = \left|\boldsymbol{C}_2 \cdot \boldsymbol{X}_\beta - \boldsymbol{X}\right|, \boldsymbol{D}_\beta = \left|\boldsymbol{C}_3 \cdot \boldsymbol{X}_\delta - \boldsymbol{X}\right| \tag{18}$$

$$\boldsymbol{X}_1 = \boldsymbol{X}_\alpha - \boldsymbol{A}_1 \cdot (\boldsymbol{D_\alpha}), \boldsymbol{X}_2 = \boldsymbol{X}_\beta - \boldsymbol{A}_2 \cdot (\boldsymbol{D_\beta}), \boldsymbol{X}_3 = \boldsymbol{X}_\delta - \boldsymbol{A}_3 \cdot (\boldsymbol{D_\delta}) \tag{19}$$

$$\boldsymbol{X}(t+1) = \frac{\boldsymbol{X_1} + \boldsymbol{X_2} + \boldsymbol{X_3}}{3} \tag{20}$$

3.5 Particle Swarm Optimizatio - PSO

The PSO algorithm came from the observation of the social behavior, birds in a flock and fish shoals [31]. The problem is expressed through the objective function. The quality of the solution represented by a particle is the value of the objective function at the position of this particle. The term particle is used, in analogy to physics, to have well defined position and velocity vector but has no mass or volume. Already the term swarm, represents a set of possible solutions. The velocity which takes the particle position close to pbest (own best position) and gbest (overall best position among the particles) is given by:

$$v_{id}^{k+1} = w_{id}^k + c_1 \times rand \times (pbest_{id} - s_{id}^k) + c_2 \times rand \times (gbest_{id} - s_{id}^k) \tag{21}$$

The current searching position of the particle can be modified by:

$$s_{id}^{k+1} = s_{id}^k + v_{id}^{k+1} \tag{22}$$

where s_{id}^k is the current searching point, c_1 and c_2 are learning factors w_{id}^k is weight function for velocity which is given by:

$$w_i = w_{max} - \frac{w_{max} - w_{min}}{k_{max}} \cdot k \tag{23}$$

where w_{max} and w_{min} are the maximum and minimum weights respectively, k_{max} and k are the maximum and current iteration number [32].

4 Data Description

The dataset was collected at Hospital Universitario de Caracas in Caracas, Venezuela and it was published in 2017 by University of California, Irvine. According to the study [33], most of the patients belong to the lowest socioeconomic status with low income and educational level, being the population with the highest risk. The age of the patients spans between 13 and 84 years old.

All patients are sexually active and 98% of them have been pregnant at least once. The dataset comprises demographic information, habits, and historic medical records of 858 patients and 32 features as well as four targets (Hinselmann, Schiller, Cytology and Biopsy). This paper focuses on studying the Biopsy target as it recommended by the literature review. The attributes in the dataset have been presented in the Table 1.

Table 1. Inputs with their respective type and valid amount of data.

Attributes	Types	Valids	Attributes	Types	Valids
Age	int	858	STDs: genital herpes	int	753
First sexual intercourse	int	851	AIDS	bool	753
Number of pregnancies	int	802	STDs: molluscum contagiosum	bool	753
Smokes	bool	845	HIV	bool	753
Smokes (years)	int	845	Hepatitis B	bool	753
Smokes (packs/year)	int	845	HPV	bool	753
Hormonal Contraceptives	bool	750	Number of diagnosis	int	858
Hormonal Contraceptives (years)	bool	750	STDs: syphilis	bool	753
STDs: cervical condylomatosis	bool	753	STDs: vulvo-perineal condylomatosis	bool	753
Number of sexual partners	int	832	Pelvic inflammatory disease	bool	753
STDs: vaginal condylomatosis	bool	753	Dx: Cancer	bool	858
STDs	bool	753	Dx: CIN	bool	858
STDs (number)	int	753	Dx: HPV	bool	858
STDs: condylomatosis	bool	753	Dx	bool	858
IUD	bool	741	IUD (years)	int	741

4.1 Preprocessing

Real-world information mostly tend to contain low quality data which could not be used directly in mining process without pre-processing. Numerous aspects may influence the performance of a learning system due to data quality. There ate two steps in preprocessing:

Data Cleaning. The cervical dataset suffers from a vast number of missing cells due to the lack of information. Removes all data for an observation that has one or more missing values is a quick solution and typically is preferred in cases where the percentage of missing values is relatively low. Because the amount of data lost is very small relatively to the size of the data set, the omission of the few samples with missing features was the best strategy for not influencing the analysis. The final data set was 668 instances and 34 attributes.

Normalization. Since the data of cancer cervical has features with different ranges, normalization was necessary. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. Thus, all the features in our experiments using [0,1] normalization.

5 Simulation and Experiments

This section demonstrates the main test scenarios that have been conducted. All neural networks models used in this article were built in framework, called Keras [?]. Keras is a framework for building deep neural networks with Python and enables us to build state-of-the-art, deep learning systems. All computations are performed using a PC/Intel Core i7-5775R processor with 32 GB DDR4 RAM, GPU GeForce GTX 1080 and 1 hard drive SSD of 300 GB SATA 5 Gb/s.

The data set was used to feed the models: LSTM-CS, LSTM-GA, LSTM-GSA, LSTM-GWO and LSTM-PSO, then weights and biases are updated to get better accuracy results and less error rate. The main objective of these experiments is to compare all models of LSTM using different parameters, such as, numbers of hidden neurons, population and iteration for each model are chosen (Table 2).

The training and testing on the same set, the mean accuracy is determined using five-fold cross-validation.

Figures 3 show the accuracy of the classifiers with different numbers of neurons in hidden layer, but without changing the number of iterations and populations, remained fixed (total of 40) for all architectures presented. LSTM-PSO

Table 2. Parameters that will be modified in the scale of 4 *to* 40. In each test, one parameter changes while the others remain fixed.

Trained networks	Iteration	Population	Number of hidden neurons
LSTM-CS	4-40	4-40	4-40
LSTM-GA	4-40	4-40	4-40
LSTM-GSA	4-40	4-40	4-40
LSTM-GWO	4-40	4-40	4-40
LSTM-PSO	4-40	4-40	4-40

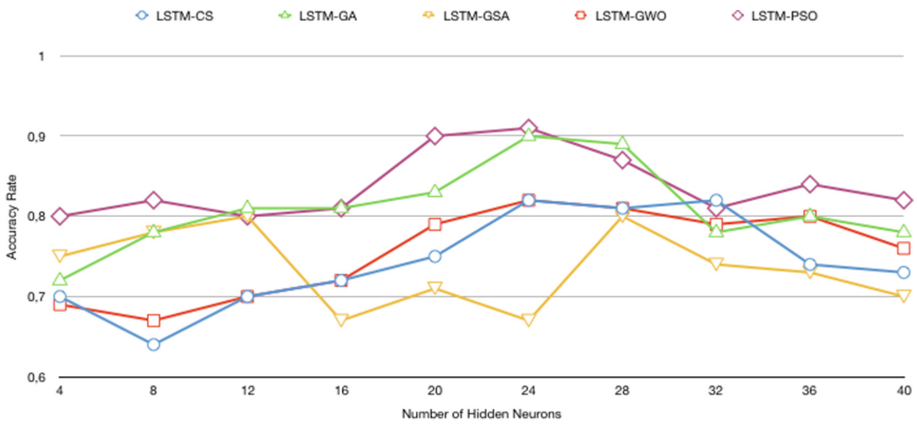

Fig. 3. Variations of the LSTM networks with the metaheuristic algorithms and their respective accuracy rate, with different number of neurons.

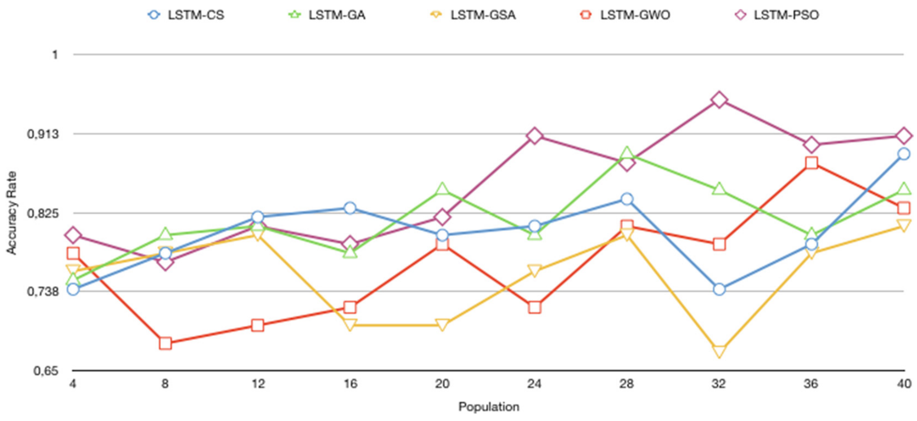

Fig. 4. Variations of the LSTM networks with the metaheuristic algorithms and their respective accuracy rate, with different population size.

and LSTM-GA arrived close to their accuracy, reaching 92% and 91% success, respectively, while LSTM-GSA had the worst result. In this test, the increase in the number of neurons did not increase the accuracy of the models. The hidden neuron can influence the error on the nodes to which their output is connected. Thus, too much hidden neurons cause too much flexibility and this leads overfitting, and therefore, the neural networks have overestimate the complexity of the target problem.

Figure 4 shows the accuracy of neural models increasing the population size. Each LSTM was tested with different population numbers to find the ideal solution, increasing the population at each step. For the classification procedures consistently different population sizes (from 5 to 40) were used. The number of

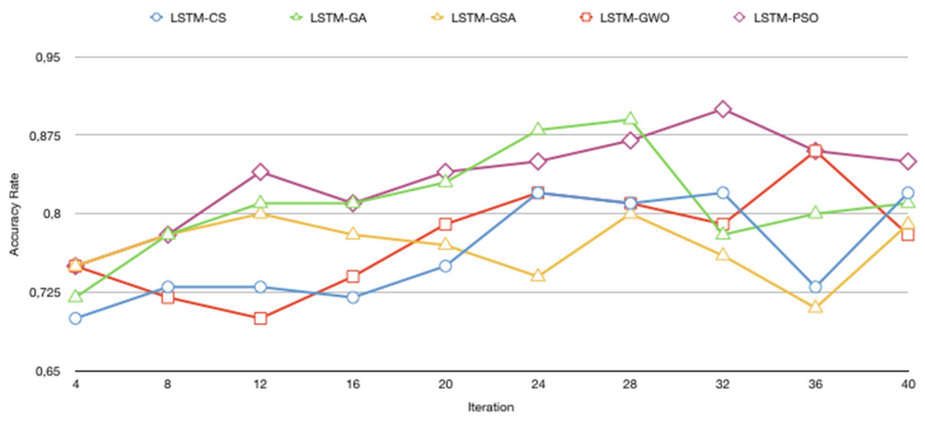

Fig. 5. Variations of the LSTM networks with the metaheuristic algorithms and their respective accuracy rate, with different number of iterations.

iterations and the number of neurons in the hidden layer were fixed to 40. In this test, we show that increasing the population size increases the accuracy of all models. The LSTM-PSO network, achieved an accuracy rate of 96%.

Figure 5 shows increase the number of iterations, while the other parameters remained fixed at 40. Despite the increase in the accuracy of the models, this improvement was quite small. A high number of iterations allows that the algorithm converges to the optimal solution, although in this case study, the performance of all models except LSTM-GSA is roughly equivalent. Again, the LSTM-PSO model obtained the best result, reaching 91% accuracy.

6 Conclusions and Future Work

The mortality rate from cervical cancer worldwide has increased significantly in recent years. The cervical tumor, threatens not only the motherhood of women, but especially their lives. This paper has proposed a LSTM network, as an alternative to deal with slow convergence and convergence to local minimum in neural networks trained with backpropagation. The paper used five different optimizers based on Nature Inspired Algorithms, such as: Cuckoo Search, Genetic Algorithm, Gravitational Search, Gray Wolf Optimizer and Particle Swarm Optimization for training LSTM neural networks and apply all models in a cervical cancer dataset. The tests were performed on a dataset containing data from Venezuelan women and they were performed by changing three parameters: the number of neurons, the number of iterations and the population size. The developed neural networks were applied to diagnose cancer patients based on a number of content related features. The LSTM-BPTT approach was evaluated and compared with other neural networks trained by five different optimizers. After that, the five optimizers were compared based on their accuracy results. The experiments showed that LSTM with PSO performed better than LSTM with the other algorithms in the term of optimizers. The accuracy for LSTM with PSO was 0.92 by varying the number of neurons, 0.96 when there was modification in the population and 0.91 changing the number of iterations.

In addition, other observations could be made according to the experiments performed. The increase in the number of neurons was not interesting for the experiment. The accuracy declines with increasing the number of neurons in the hidden layer, and this might be due to the increased complexity in the network's structure, which requires more training time to converge. On the other hand, the increase of the population was beneficial for the performance of models. Meanwhile, the increase in iterations did not show significant improvements.

Finally, this paper has provided a method for detecting women with cervical cancer with higher accuracy resulting by applying LSTM to overcome the back-propagation problems. The future work for this paper will include using other ways to address the missing data problem, verification of the optimization algorithms used in other types of neural networks and investigating the effectiveness of LSTM and algorithms with larger datasets. Prediction of different diseases with different datasets using LSTM neural networks could also be considered.

Acknowledgments.. This research is supported by Instituto Federal de Mato Grosso do Sul.

References

1. National Cancer Institute Homepage. https://www.inca.gov.br/. Accessed 15 Dec 2018
2. World Health Organization Homepage. https://www.who.int/cancer/prevention/diagnosis-screeningcervical-cancer/. Accessed 12 Dec 2018
3. Sharma, S.: Cervical cancer stage prediction using decision tree approach of machine learning. Int. J. Adv. Res. Comput. Commun. Eng. **5**(4) (2016)
4. Zhang, J., Liu, Y.: Cervical cancer detection using SVM based feature screening. In: Barillot, C., Haynor, D.R., Hellier, P. (eds.) MICCAI 2004. LNCS, vol. 3217, pp. 873–880. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30136-3_106
5. Alwesabi, Y., Choudhury, A., Won, D.: Classification of cervical cancer dataset. In: IISE Annual Conference (2018)
6. Devi, M.A., Ravi, S., Vaishnavi, J., Punitha, S.: Classification of cervical cancer using artificial neural networks. Procedia Comput. Sci. **89**, 465–472 (2016)
7. Hochreiterm, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
8. Sakti, S., Ilham, F., Neubig, G., Toda, T., Purwarianti, A., Nakamura, S.: Incremental sentence compression using LSTM recurrent networks. In: IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Scottsdale, AZ, pp. 252–258 (2015)
9. Gogate, M., Adeel, A., Hussain, A.: A novel brain-inspired compression-based optimised multimodal fusion for emotion recognition. In: IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, pp. 1–7 (2017)

10. Breuel, T.M.: High performance text recognition using a hybrid convolutional-LSTM implementation. In: 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, pp. 11–16 (2017)
11. Su, M., Wu, C., Huang, K., Hong, Q.: LSTM-based text emotion recognition using semantic and emotional word vectors. In: First Asian Conference on Affective Computing and Intelligent Interaction (ACII Asia), Beijing, pp. 1–6 (2018)
12. Sarma, K., Sarma, M.: Acoustic modelling of speech signal using artificial neural network: a review of techniques and current trends, pp. 287–303. IGI Global (2015). https://doi.org/10.4018/978-1-4666-8493-5.ch012
13. Sak, H., Senior, A., Beaufays, F.: Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. In: Interspeech, pp. 338–342 (2014)
14. Billa, J.: Dropout approaches for LSTM based speech recognition systems. In: IEEE International Conference on Acoustics. Speech and Signal Processing (ICASSP), Calgary, AB, pp. 5879–5883 (2018)
15. Lin, C., Wan, J., Liang, Y., Li, S.Z.: Large-scale isolated gesture recognition using a refined fused model based on masked res-C3D network and skeleton LSTM. In: 13th IEEE International Conference on Automatic Face Gesture Recognition, pp. 52–58 (2018)
16. Zhu, G., Zhang, L., Shen, P., Song, J.: Multimodal gesture recognition using 3-D convolution and convolutional LSTM. IEEE Access **5**, 4517–4524 (2017)
17. Xu, S., Xue, Y.: A long term memory recognition framework on multi-complexity motion gestures. In: 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, pp. 201–205 (2017)
18. Park, H.: Part 2: multilayer perceptron and natural gradient learning. New Gener. Comput., 79–95 (2006)
19. Sundermeyer, M., Ney, H., Schluter, R.: From feedforward to recurrent lstm neural networks for language modeling. IEEE/ACM Trans. Audio Speech Lang. Process. **23**(3), 517–529 (2015)
20. Barreto, J.M.: Introdução as redes neurais artificiais. In: V Escola Regional de Informáttica. Sociedade Brasileira de Computação, Regional Sul, Santa Maria, Florianópolis, Maringá, pp. 5–10 (2002)
21. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning, 1st edn. Massachusetts Institute of Technology, Cambrigde (2016)
22. Schmidhuber J.: Deep learning in neural networks: an overview. In: Neural Networks, pp. 85–117 (2015)
23. Graves, A.: Supervised Sequence Labelling with Recurrent Neural Networks. Studies in Computational Intelligence, vol. 385. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-24797-2
24. Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A novel connectionist system for unconstrained handwriting recognition. IEEE Trans. Pattern Anal. Mach. Intell., 855–868 (2009). https://doi.org/10.1109/TPAMI.2008.137
25. Yang, X.S., Deb, S.: Cuckoo search: recent advances and applications. Neural Comput. Appl. **24**(1), 169–174 (2014)
26. Biswas, S., Kundu, S., Das, S.: Inducing niching behavior in differential evolution through local information sharing. IEEE Trans. Evol. Comput. **19**(2), 246–263 (2015)
27. Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor (1975)

28. Rashedi, E.: Gravitational Search Algorithm. M.Sc. thesis, Shahid Bahonar University of Kerman, Kerman, Iran (2007)
29. Mirjalili, S., Mohammad, S., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014)
30. Long, W., Songjin, X.: A novel grey wolf optimizer for global optimization. In: Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), pp. 1266–1270. IEEE (2016)
31. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Network, Perth, Australia (1995)
32. Yang, X.S., Deb, S.: Engineering optimisation by cuckoo search. J. Math. Model. Numer. Optim. **1**(4), 330–343 (2010)
33. Fernandes, K., Cardoso, J.S., Fernandes, J.: Transfer learning with partial observability applied to cervical cancer screening. In: Alexandre, L.A., Salvador Sánchez, J., Rodrigues, J.M.F. (eds.) IbPRIA 2017. LNCS, vol. 10255, pp. 243–250. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58838-4_27