Classifying Time Series Data Using LSTM and Genetic Algorithm

Yanxi Lu

College of Engineering and Computer Science, Australian National University
u5952700@anu.edu.au

Abstract. Time series classification(TSC) is an important and challenging problem in this era[1]. Traditional neural network approaches such as deep learning[1] and Recurrent Neural Network(RNN)[2] have been developed to carry out this task. However, most of these approaches cannot represent long-term dependencies in the data set. This paper proposes a Long Short-Term Memory(LSTM) model, which is a specific RNN architecture designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs[3]. The model is tested on a complex classification problem involving predicting whether a picture is manipulated given time series data of human eye gaze on it. The focus of investigation lies on performance of the model as compared to Constructive cascade network[4] which is said to be able to model the complexity of the problem. The result shows the superiority in generalization power of the LSTM model. A genetic algorithm is then applied to the LSTM model to reduce the redundancy in the data features[5] and exhibited positive results. Based on these observations, it is reasonable to assume that LSTM is a better approach to TSC tasks. When combined with appropriate techniques such as feature extractions, it can be used to handle time series data of high dimensionality.

Keywords. Neural network, LSTM, Genetic algorithm, Time series data, Constructive cascade network, Classification problem

1 Introduction

In recent decades, there has been an explosion of interest in modeling time series data. Hundreds of papers have introduced new algorithms to classify time series.[6] The main characteristic of time series data is the temporal sequence it possesses, which results in data points being related and classification results dependent on the context of the data. Traditional approaches such as standard deep feed-forward networks only exploit a fixed length of data for prediction, hence cannot effectively represent the context of the data.[7] Recurrent Neural Network(RNN) on the other hand, can take into account all the previous data points but is difficult to train and tends to forget, therefore is unlikely to show the full potential of recurrent models.

Long Short-Term Time(LSTM) architecture builds on standard RNN and provides memory blocks to store the state of at each time step. Each memory block contains one or more memory cells along with input, output and forget gates which control flow of information into and out of the memory cell. Due to its unique design structure, LSTM can control what to store, when to memorize and when to forget with its memory cells. It has solved artificial long-time-lag tasks, which was never achieved by standard RNN[8].

This paper proposed an LSTM model to solve a complex time series classification task on predicting whether an image is manipulated given a set of data of human eye gaze on the image[9]. The data set records the time and duration of eye gaze of viewers hence should be categorized into time series data. It is intuitive that a sequence of eye gaze performed by views instead of a single data point is related to whether the picture is manipulated. By comparing the performance of LSTM model to constructive cascade network, which is a standard neural networks approach, the paper aims to investigate the importance of context of data in performing classification tasks and the ability of LSTM to model this context. Positive result would show prove the difference between ordinary and time series classification tasks as well as the viability of LSTM in handling these special tasks. The paper also investigates the performance of hybrid approaches between LSTM and genetic algorithm to examine the effectiveness of feature extraction in this task.

2 Method

2.1 Data Set

There are three sets of data for this problem. They were originally proposed by Caldwell et al. based on her eye-gaze tracking experiment of 80 participants viewing a combination of manipulated and unmanipulated images[9][10]. The first data set is a condensation of the time series data which only lists the number of eye gaze fixations by participants. The detailed information of each fixation is listed in the second set. The final set contains information on the pixels manipulated in manipulated images, which is also an extension to the first set. In this problem, the third data set is not considered because they explicitly state that the images are manipulated.

Since the second data set extends the first one and does not contain the target of this task(if the image is manipulated), it is combined with the first set. The resultant data set includes 31114 lines of data for the 80 participants' eye gaze on manipulated and original versions of 5 images. It contains 11 features, where feature 1 and 2 are the fixation's id and participant id. Feature 3 is the image id. Feature 4 to 9 describe the X, Y index, starting and ending time, duration and number of samples taken in the duration of that eye gaze respectively. The last two features indicate whether the picture is manipulated and the vote by the participant on whether the picture is manipulated.

In this classification task, the goal is to determine whether the participant is looking at a manipulated image based on their eye gaze on it. Hence, the first nine features are taken as inputs and the feature indicating if the image is manipulated is the output.

All data used for this classification task is in numerical form but with different domains. The output feature uses 1 to denote a manipulated image and 0 otherwise. While for the nine input features, a wide range of values is observed. Sola and Sevilla [11] points out that data normalization is crucial to obtaining good results as well as to accelerate the learning process. The input data is hence normalized using z-score normalization[12] given by formula (1). It is

$$z = \frac{x - \mu}{\sigma} \qquad (1)$$

worth noting that this technique is sensitive to outliers; hence outlier removal strategies can be used to improve the classification result.

The data set is split into three parts, where data on the first three images are the training data, and the last two images are for validation and test sets respectively. The size of each part is large hence applying stochastic gradient descent(SGD) may be time-consuming. Therefore, the data is further processed to speed up the training process.

2.2 Data Preprocessing

The training set contains around 20000 data points, which is time-consuming to train. Therefore, a different strategy, mini batch gradient descent is used.[13] Mini batch strategies are known to conserve computation power than SGD. Instead of back-propagating the error for each training data in SGD, it only back-propagates after each batch is trained. In this model, the size of batch is set to 32, which is a traditional practice to utilize computer memory[13].

Furthermore, although LSTM has memory cells to control what to remember and when to forget, it is not desirable to pass sequences of thousands of data points to it at once because of the nature of gradient descent. Similar to standard to RNN, when back-propagating gradients through long time windows the gradients are likely to explode or vanish. On the other hand, passing data points one at a time would deviate from the purpose of training the model with the context of data. Therefore, ten consecutive data points are put together in a sequence to be passed to the model to avoid both extremes. Each batch now contains 320 sequences. Therefore, there are around 60 batches for the training data.

2.3 LSTM Topology

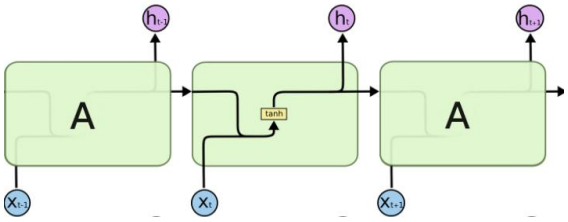

Fig. 1 RNN

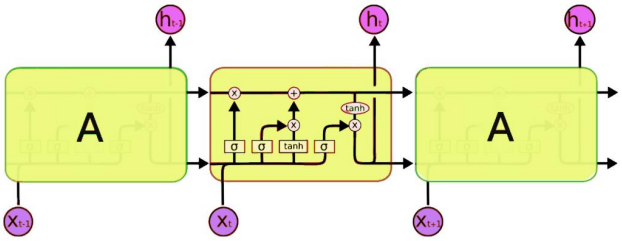

Fig. 2 LSTM

Fig.1 and Fig.2[14] shows the standard RNN structure and LSTM architecture respectively. A represents a full RNN cell that takes the current input of the sequence x_i, and outputs the current hidden state, h_i, passing this to the next RNN cell. The inside of a LSTM cell is a lot more complicated than a traditional RNN cell. It contains input, output and forget gates which control flow of information into and out of the memory cell. With the gates acting as the control agents, memory cells can store information until it is relevant. For each gate, there is a separate criterion on when it opens and closes. The exact formulas are not mentioned here because of the heavy involvement of mathematical notations.

2.4 Genetic Algorithm

The data set has nine features, and some are potentially redundant. For example, the duration of eye gaze is proportional to the number of samples collected in that duration. Genetic algorithm could be an approach here to select relevant features and reduce the dimensionality of input, therefore reduce the complexity of the problem. It has several key components, including the DNA representation of inputs, a population, a fitness function to evaluate the fitness of a DNA, a process to select DNA based on their fitness, a crossover function to produce next-generation DNA and a mutation function to introduce uncertainty to the model. Here, the purpose is to select features; hence, DNA is encoded as a sequence of 9 bits to represent the selection, where each bit represents if the feature is selected. The population is constituted by 10 randomly generated bit strings and another DNA with all bits set to 1. The later is to make sure there is a basis to the selection. The fitness of a DNA is evaluated by its validation loss after training is completed. For each generation, DNA is selected with probably based on their fitness to balance explore and exploit. Selected DNA are then crossed over at randomly chosen crossover bits and the resultant child has a probability of mutation, where the corresponding bit is reversed. The algorithm terminates when either it converges or the maximum number of generations is reached. Then the best DNA would be selected as the final result.

2.5 Constructive Cascade Network

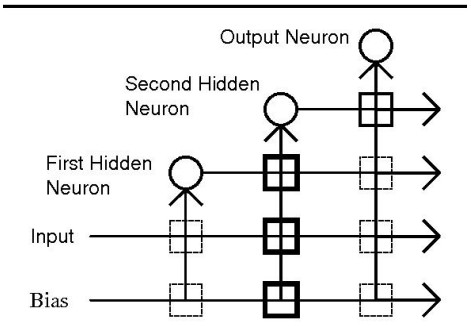

Fig. 3

The performance of the LSTM model is compared with a constructive cascade network. Constructive cascade networks are known to able to construct its network topology during training, hence being able to model the inherent complexity of the problem. Fig. 3 shows the structure of a typical constructive network, Cascade Correlation(CasCor). It starts with fully connected input and output layers. During training, hidden neurons are dynamically added. In the case of each new hidden neuron forming a new layer, each neuron added is connected fully to the input and output layer, as well as all the hidden neurons before. The process continues until a satisfactory solution is found.

The constructive network used as the referent here is a slight modification of the algorithms mentioned above. Instead of adding one hidden neuron, a cascade layer containing several hidden neurons is added each time. The rationale of doing so is to speed up the convergence at the expense of higher computational cost each layer.

2.6 Training Methodology

The methodology adopted is tuned to fit the purpose of this training task. For the LSTM, The weights and bias of the initial network are initialized to small random values by default. There are two LSTM layers stacked on each other to increase the generalization power of the network. Each layer has 6 hidden units so that it is smaller than the number of inputs in order to restrict the functionality of each layer and reduce the total number of hidden neurons. The number of epochs is set to 200 because some preliminary experiment shows that convergence is usually around 70 epochs. The optimizer chosen is Adam because it exhibited much better performance in experiment than SGD in terms of convergence speed and final loss. The learning rate is set to 0.01 which is standard for mini batch training. The loss function used is cross entropy loss since this is a two-class classification problem.

For the genetic algorithm, the crossover rate is set to be 0.8 to encourage exploitation of good DNA and mutation rate is set to 0.002 to allow some degree of exploration. The number of generations is set to be 5 because each generation requires a long time to train. Furthermore, to decrease the overall duration of training, only one of 10 data points are trained with the LSTM model.

For the referent constructive cascade network, the activation function used after each hidden layer is sigmoid and binary cross entropy loss is used for evaluation as this is a two-class classification problem in the range [0,1]. Each layer has 6 hidden neurons to match that of LSTM. Resilient propagation(RPROP) [15] is used for learning. The main difference between RPROP and traditional back-propagation is that RPROP uses the sign of gradients only. This allows the learning process to become smoother and less prone to outliers.

During training, the LSTM model checks the validation error every 10 epochs. It terminates when the maximum epoch is reached or the validation error fails to decreases after two more periods. The second criterion aims to account for the uncertainty in the validation set. However, if it did not get better after 20 epochs, it is likely that learning is complete. The cascade network starts from the initial state containing only the input and output layers for a maximum of 800 epochs. A new cascade layer is added whenever the maximum number epoch is reached or the training error is smaller than 0.2. The values are chosen based on prior testing with typical multilayer networks. Each time when a new cascade needs to be added, the network is used on the validation set to measure the validation error. Similarly, it terminates when the maximum number of cascades is reached or the validation fails to decrease after two additional cascades. Both networks adopt an early stopping approach which has been proven to be able to increase network generalization ability[16][17]. For each model, the output layer comprises 2 units for the classes manipulated and unmanipulated, and a winner-takes-all strategy is used to decide the output of the network.

When training LSTM with genetic algorithm, similar settings as mentioned above is adopted except the batch size is set to 10. This is because only one of ten data points are used, using large batch size would produce a small number of batches, making the training process similar to full batch gradient descent which is undesirable.

3 Results and Discussion

Both the LSTM and the constructive cascade network are tested on the classification task described in Section 2.1. Their performances are compared and analyzed. Then, genetic algorithm is used in hybrid with LSTM and the result is compared to using LSTM only. As mentioned before, the focus of the investigation is the generalization power of each model, though other benchmarks such as computational time are also considered.

3.1 LSTM and Constructive cascade network

For the first experiment, the genetic algorithm is not used. The performance of raw LSTM and constructive cascade networks on the classification task are compared, each taking the average result of 20 runs. Table1 shows the result of

the experiment. It is clearly observed that LSTM has a higher generalization power than the constructive cascade network implemented despite being trained for a much smaller number of training epochs. However, the huge difference in the number of epochs performed does not account for the actual training. Firstly, most of the epochs of the constructive cascade network are performed when the number of layers is small, which leads to shorter time per epoch. Secondly, the number of back-propagation performed in the LSTM model is much bigger than that in the constructive cascade network because of the mini batch strategy used. For each epoch, there are around 60 batches to perform back-propagation after, hence the total number of times performed is around 2500. While for the constructive cascade network, Rprop is perform only once per epoch, and for most of the time, the number of weights to update is small because of the small number of layers. In the actual experiment, the time taken for 20 runs to finished is much longer when training the LSTM model.

	Termination Epoch	Test Accuracy/%	Remark
LSTM	44	84.61	Usually, it terminates with 40 epochs, only once has it gone beyond 60 steps.
Constructive Cascade	1063	81.80	70% of time, the resultant network structure has the maximum number of cascades(5).

Table1. Result of LSTM and CCN

There is another notable observation from the result. The stability of LSTM in performing this task is very high, most of the runs terminate between 30 and 50 epochs and the test accuracy is around 83%, which is a relatively good result given the large amount of data and uncertainty in each participant's behaviour. For constructive cascade network, however, the stability is relatively low. The resultant network structure has a variety of number of cascades and in many runs restricted by the maximum number of cascades. Furthermore, its test accuracy ranges from 70% to near 90%.

Based on the result above, it can be concluded that despite the improvement of accuracy of LSTM over constructive cascade networks is only by several percentages, it is a notable 15 percent decrease in error rate. This may also show that the context of data indeed affects the classification outcome in this problem and LSTM is capable of modeling this context.

To summarize, it can be concluded that this LSTM model with mini batch has a longer training time than constructive cascade networks. However, LSTM is superior in both generalization power and stability in performing this classification task.

3.2 Comparing against typical neural networks

The LSTM result obtained above is compared to typical neural networks. The result Sabrina presented in her thesis on this classification problem[10] adopts a neural network designed with a feed-forward back-propagation algorithm. The overall accuracy of successfully classifying the pictures is 66%. This confirms that the LSTM model is much better in modelling time series data than standard networks.

3.3 LSTM with genetic algorithm

The LSTM model is then used in hybrid with the genetic algorithm to explore the feasibility of feature extraction from the data. The result is surprising. There are around 25 batches of size 10, with randomly initialized DNA, both the training loss and validation loss are very low and the accuracies are close to 100%. It is probable that this is due to the small size of training data since only one in ten is taken. Also, the small number of batches may have caused the model to adjust to their losses easily.

Despite the abnormal observation, the result confirms that some features can be extracted from the original list of features. It is occasionally observed that random DNA performs better in the number of epochs to raise validation

accuracy to above 90% than having all the features. The experiment is also performed when setting the batch size to 1, and the result is similar.

4 Conclusion and Future Work

This paper investigates the possibility of using LSTM in classification problems with time series data. In general, it produces better results than typical architectures, including constructive cascade network and standard feedforward network. It can monitor the temporal sequence with the data set and take into account the context of the data set when making the classification. It also produces stable outcomes. However, there is a tradeoff between the generalization power and the computational time. Furthermore, when combined with feature selection algorithms such as genetic algorithm, the model can effectively extract relevant features from a pool of raw ones and yield a very high accuracy on the classification task.

In the future, I will try to validate the effectiveness of LSTM on more complex data sets and problems beyond classification, where I can explore the possibility of using deep LSTM networks which have even stronger generalization power. In addition, though unobserved in this experiment, LSTM is sometimes unstable because of the long chain of error back-propagation. I will explore small variants such as gradient clipping and reversing inputs to make the model more stable. Furthermore, I will investigate other possibilities of hybrid, for example, LSTM with convolutional neural networks, to model time series data of images for more complicated tasks such as movement prediction.

References

1. Hassan I. F., Germain F, Weber J., Idoumghar L., Muller P. A.: Deep learning for time series classification: a review. In: Arxiv Data Mining and Knowledge Discovery. 2019.
2. Mikolov, T., Karafiát M., Burget L., Černocký J., Khudanpur S.: Recurrent neural network based language model. In INTERSPEECH-2010, pp. 1045-1048, 2010.
3. Sak, H., Senior, A., Beaufays, F.: Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. In: In Fifteenth annual conference of the international speech communication association. 2014
4. Khoo, S., Gedeon, T. Generalisation Performance vs. Architecture Variations in Constructive Cascade Networks. In: Köppen M., Kasabov N., Coghill G. (eds) Advances in Neuro-Information Processing. ICONIP 2008. Lecture Notes in Computer Science, vol 5507. Springer, Berlin, Heidelberg 2009.
5. Yang, J., Honavar, V.: Feature Subset Selection Using A Genetic Algorithm. In: Feature extraction, construction and selection. pp. 117-136. 1998.
6. Keogh, E., Kasetty, S.: On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. In: Data Mining and Knowledge Discovery Volume 7, Issue 4, pp 349 – 371, 2003.
7. Sundermeyer, M., Schlüter, R., Ney, H., LSTM Neural Networks for Language Modeling. In: INTERSPEECH-2012, pp.194-197, 2012.
8. Sepp, H., Jürgen, S.: Long Short-term Memory. In: Neural computation. 9, 1997.
9. Caldwell, S., et al.: Imperfect understandings: a grounded theory and eye gaze investigation of human perceptions of manipulated and unmanipulated digital images. In: Proceedings of the World Congress on Electrical Engineering and Computer Systems and Science. Vol. 308. 2015.
10. Caldwell, S..: Framing digital image credibility: image manipulation problems,perceptions and solutions. In: ANU Theses Open Access.
11. Sola, J., Sevilla, J.: Importance of input data normalization for the application of neural networks to complex industrial problems. In: IEEE Transactions on Nuclear Science, vol. 44, no. 3, pp. 1464-1468, June 1997.
12. Jain, A., Nandakumar, K., Ross, A.: Score normalization in multimodal biometric systems. Pattern recognition, 38(12), pp.2270-2285. 2005

13. Li, M., Zhang, T., Chen, Y., Smola, A. J.: Efficient Mini-batch Training for Stochastic Optimization. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 661-670). ACM, 2014
14. Hoffman, G.: Introduction to LSTMs with TensorFlow.
Available at: https://www.oreilly.com/ideas/introduction-to-lstms-with-tensorflow, 2018.
15. Riedmiller, M., Braun, H.: A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: IEEE Int. Conf. on Neural Networks, pp. 586–591 1993.
16. Finnoff, W., Hergert, F., Zimmermann, H.G.: Improving model selection by nonconvergent methods. In: Neural Networks, Vol 6. pp.771-783. 1993.
17. Treadgold, N.K., Gedeon, T.D.： Exploring constructive cascade networks. In: IEEE Transactions on Neural Networks, Vol 10, pp.1335-1350. 1999