Intelligent Aggregation for Ensemble LSTM

Ashima Elhence, Abhishek(✉), and Shekhar Verma

Department of Information Technology, Indian Institute of Information Technology, Allahabad, Allahabad 211012, Uttar Pradesh, India
{ism2013015, rsi2016006, sverma}@iiita.ac.in

Abstract. Analysis of time series data requires extracting hidden patterns and accurate detection of outliers and anomaly. Long Short Term Memory based neural network's ability to hold relevant information for extended time makes it suitable for such analysis. A single network increases the prediction accuracy but is limited by the group of similar patterns it can learn which results in overfitting. Ensemble of thinned networks using dropout regularization solves this problem but due to lack of efficient aggregation method, its capability gets restricted. In this paper, we explore two intelligent ensemble aggregation methods which allows to maximize the thinned networks' performance. Extensive experiments on Yahoo! benchmark dataset show that both aggregation techniques are capable of handling unwanted effects in data to improve the average performance by ≈52% as compared to single network.

Keywords: Time series prediction · LSTM · Ensemble · Weighted aggregation Window based aggregation

1 Introduction

Successive equally spaced data instances varying against time is defined as time series data e.g. IoT [1], health care monitoring [1], financial monitoring, anomaly detection [2], scene labeling [3] etc. Time series data analysis requires behavioral study of the data by extracting the underlying pattern for accurate predictions on future values. Presence of anomaly and outliers pose major challenges in time series analysis. An accurate identification of anomaly and outliers conditions would allow underlying model to plan preventive or reactive measures accordingly. As time series data is non-stationary and frequently varying, multiple conditional behaviors can be observed [4]. It is very important to build a robust and reliable model which is able to handle different aspects of data and make accurate predictions accordingly. At the same time underlying model should not be biased to some specific behaviors.

A RNN (Recurrent Neural Network) [5] uses its internal memory to process sequence of inputs unlike feedforward neural network which makes them efficient for time series analysis. A basic RNN can be constructed using single instance of the model but it may not be capable of extracting all patterns inside the given data. Time series analysis remains delay intolerant and needs real time prediction. One network overloaded with whole analysis takes more time to process and give result which makes it inefficient. A possible solution would be create multiple instances of the

S. Verma et al. (Eds.): CNC 2018, CCIS 839, pp. 466–475, 2019.
https://doi.org/10.1007/978-981-13-2372-0_41

model but since identical models will learn similar functions and would extract same patterns, this also becomes inefficient. Thus, a generalized model can be constructed by creating separate basic models optimized for different patterns and thereafter creating an ensemble of them. The predictive performance and the generalizability of the model is improved significantly with ensemble learning [6].

LSTM (Long Short Term Memory) [7] is one of the most frequently used RNN model used for time series analysis, natural language processing, sequence modelling and learning. RNN taps into the temporal and sequential aspects of the data in a fixed number of computational steps making them fast and suitable for delay intolerant applications. Along with advantages of traditional RNN, LSTMs simple model, capability to handle potentially infinite dynamic data and ability to store important information for a significantly longer duration makes it preferable model for time series analysis e.g. time series prediction [8], event forecasting [9], anomaly detection [10] etc. Multiple LSTM networks replacing single deep LSTM network [2, 5] increase the accuracy by analyzing single or a group of similar behaviors. Ensemble LSTM independently trains each of the component models such that no two models are identical.

Final prediction from ensemble LSTMs require aggregation. A basic aggregation technique can be done by averaging prediction of different networks. As different LSTM networks are trained for different functions, giving them equal importance leads to inaccurate predictions. In this paper, we propose two weighted aggregation techniques which gives appropriate importance to different LSTM networks which further increases the models accuracy. Rest of the paper is organized in following sections, Sect. 2 describes the problem faced in ensemble LSTM aggregation followed by Sect. 3 containing the proposed intelligent aggregation methods. Results obtained from extensive experiments are shown in Sect. 4. Conclusion and findings of the proposed work is described in Sect. 5.

2 Problem Description

Given time series data set $\{x_i, y_i\}_{i=1}^{t-1}$ containing labels or predictions uptil time instance $(t-1)$. Time (t) onwards, for a given data instance $\{x_t\}$, we need to predict accurate $\{y_t\}$. Ensemble LSTM network containing n differently trained LSTMs $\{L_k\}_{k=1}^{n}$ are built. The existing aggregation is done by

$$y_t = \frac{1}{n}\sum_{k=1}^{n} L_k(x_t)$$

As this aggregation gives equal weight to all networks hence, even the least accurate network is able to adversely affect the final prediction. Our aim is to develop aggregation technique to give appropriate importance to underlying LSTMs based on their individual performance. Additionally the aggregation should be able to give higher preference to recent instances in order to extract the current trend.

3 Aggregation of Ensemble Thinned LSTMs

A time series predictor harness the power of LSTM and takes advantage of ensemble networks to provide accurate next time instance prediction. Given a time series data till $(t-1)$, the point at time step (i) is represented by $\{x_i\}$, the next predicted data point $\{x_t\}$ at time t is given by

$$x_t = f(x_1, x_2, \ldots x_{t-1} : \phi^*_{t-1}) \tag{1}$$

Here f is the function responsible for capturing the behavior of the time series and $\phi(t-1)$ are the parameters of the trained model till $(t-1)$. The parameters get updated with each new observation. As a data instance arrives, the loss function of the model is updated by

$$\phi^*_{t-1} = \underset{\phi_{t-1}}{argmin} \parallel x_{t-1} - f\left(x_1, x_2, \ldots x_{t-2} : \phi^*_{t-2}\right) \parallel_2^2 \tag{2}$$

3.1 Long Short Term Memory

LSTM is an advanced version of RNN which is capable of remembering the extracted information for an extended interval of time as compared to the basic RNN used for Deep Learning. LSTM has the capability to store relevant or indicative information for a sufficiently long time. It also selects important and reduces irrelevant features automatically to avoid curse of dimensionality [11].

A single LSTM unit consists of current, input, output and forget gate. A single layer of LSTM consist of multiple such units and the whole LSTM network contains several such layers.

- **Current Memory**
  It is the currently stored information present in the LSTM cell which is used to make predictions. The update that is made on the previous cell state $\mathbf{C_{t-1}}$ which results in the new cell state $\boldsymbol{C_t}$

$$C_t = f_t * C_{t-1} + i_t * C_t \tag{3}$$

  Output from the current time step is prediction for the immediately following step. LSTM also stores the necessary cell state and information for the future predictions thereafter. After the cell state is updated for the future time instance, the output prediction for the immediate time step is given using a Sigmoid function

$$O_t = W_0 * [h_{t-1}, x_t] + b_0 \tag{4}$$

  here, with respect to output gate, $\boldsymbol{W_o}$ is its weight, $\boldsymbol{b_o}$ represents bias, and $\boldsymbol{O_t}$ is the output given by the cell at time *(t)*.

- **Forgetting Mechanism**
  It enables the model to decide whether to remember or discard the previously acquired information. The sigmoid function is used for forget gate is as described

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{5}$$

  here, $\boldsymbol{W_f}$ is the forget gate's weight, $\boldsymbol{b_f}$ denotes bias, and $\boldsymbol{h_{t-1}}$ represents the output of previous time instance$(t-1)$.
- **Saving Mechanism**
  It allows the model to extract new information from just arrived data. The logic used for the function is

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{6}$$

  here, $\mathbf{W_i}$ and $\mathbf{b_i}$ represents weight and bias of the input gate respectively. A tanh activation function is used for updating state of the LSTM cell. The vector of new values as stored in the LSTM can be defined as

$$C_t = tanh(W_c * [h_{t-1}, x_t] + b_c) \tag{7}$$

  here, $\mathbf{W_c}$ and $\mathbf{b_c}$ are the weight and bias of the current memory gate respectively.

The architecture used for prediction is an LSTM network is similar to the baseline [12] as shown in Fig. 1.

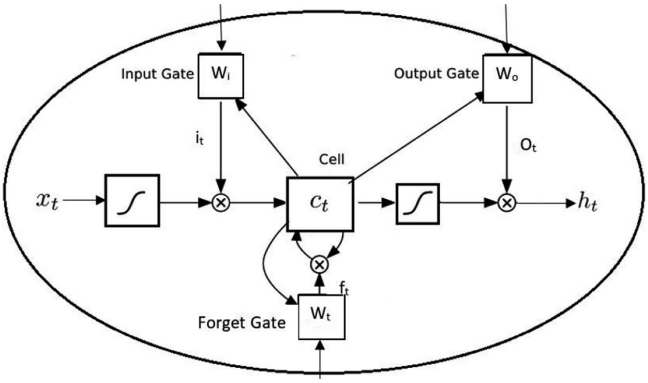


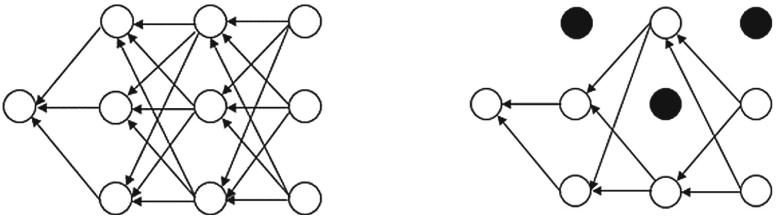
Fig. 1. LSTM representation

3.2 Ensemble LSTM

As a single network specializes in specific or group of similar patterns, they tend to become less accurate as time grows. Ensemble of weak models make more powerful model than individual deep models. As a single deep model takes more time to train

and predict, latency intolerant analysis get adversely affected from this. It may also become particularly overfitted according to some peculiar trend in the data. Presence of outliers and anomaly also have a higher impact on single instance. On the other hand a collection of independent models, each differently trained are able to solve the above problems. Independent models are also capable of controlling the biasness by bringing in variance and making overall model independent of both the training data as well as the model's architecture.

In a time series prediction using LSTM neural networks, variance among the different models of the ensemble is done by using dropout regularization [13]. Dropout regularization allows neural network layers to drop units along with their connections during training as shown in Fig. 2b. These thinned networks are hence trained independently and differently from each other as different neurons are dropped from each of them. As due to different trainings, the models do not co-adapt together. Also, dropping different neurons helps avoiding function overfitting problem.



(a) A typical LSTM neural network **(b) LSTM with dropouts**

Fig. 2. Dropout regularization

The final prediction from such thinned networks need an aggregation technique to improve the performance. The simple possible way to aggregate a fixed-sized model is to average the predictions of all possible settings of the parameters [14]. This increases the combined model's error as the least accurate network also gets the same importance to affect the prediction as that of highly accurate network.

3.3 Intelligent Aggregation

In multiple thinned network $\{L_k\}_{k=1}^{n}$, each L_k gives its prediction for $\{x_t\}$. The final prediction is done by aggregating those individual predictions. A simple aggregation is done by averaging the predictions. Equal importance to all networks defies the logic behind ensemble LSTM. As worst performing LSTM also gets the same power to affect the final result as compared to best performing network. Also, not all the functions learnt through thinned networks contains the same information gain hence, giving then equal weight does no t always increase the model's accuracy.

Weighted Aggregation. Rather than simply taking the mean of all the predictions and giving equal importance, an alternative technique is to give them weights according to their current performance and take a weighted mean of all the predictions to generate

the output. The models which are performing better in terms of accuracy as measured by low $\varsigma/RMSE$ (Root Mean Square Error) should be given higher weight than those with higher ς. Hence, the importance coefficient maintains inverse proportional relationship with ς. Given labeled data $\{x_i, y_i\}_{i=1}^{t-1}$ till time $(t-1)$, it is divided in two parts for training and coefficient calculation respectively.

$$\{x_i, y_i\}_{i=1}^{t-1} = \{x_i, y_i\}_{i=1}^{t-l} \cup \{x_j, y_j\}_{j=t-l+1}^{t-1}$$

Each $L_k \in \{L_k\}_{k=1}^{n}$ is trained using $(t-l)$ labeled data $\{x_i, y_i\}_{i=1}^{t-l}$. A prediction is obtained from all thinned LSTMs using l labeled data $\{x_j, y_j\}_{j=t-l+1}^{t-1}$ as

$$\hat{y}_{kj} = L_k(x_j)$$

where $\hat{y}_{kj}$ is the label predicted by LSTM $\{L_k\}_{k=1}^{n}$ for data $\{x_j, y_j\}_{j=t-l+1}^{t-1}$, ς is calculated from

$$\varsigma_k = \sqrt{\frac{\sum_{j=t-l+1}^{t-1} \| \hat{y}_{kj} - y_j \|^2}{t-l}}$$

Weighted *m* Window Aggregation. In weighted aggregation, we consider the whole prediction of thinned networks but in time series analysis, recent data have more importance than previous data. By keeping the λ fixed based on previous experience may lead to erroneous prediction. The models training is done similar to previous method using $t - 1$ instances. Labeled data from $t - l + 1$ to $t - 1$ is divided in m sized time windows. λ is calculated on a these m sized time windows and accordingly the coefficients are changed for enhancing recent trend prediction. After each m instances, the ς is calculated using

$$\varsigma_k = \sqrt{\frac{\sum_{j=t-m-1}^{t-1} \| \hat{y}_{kj} - y_j \|^2}{m}}$$

As each ς is on different scale, it needs to be normalized for both aggregation methods. The normalized & is obtained by

$$\Rightarrow \varsigma_k = \frac{\varsigma_k - \mu_\varsigma}{\sigma_\varsigma}$$

where, μ_ς is the mean of all n LSTMs and σ_ς is its standard deviation. Importance coefficient λ for each thinned network is calculated from

$$\Rightarrow \lambda_k = \frac{1}{\varsigma_k}$$

where, λ_k is coefficient of k^{th} LSTM and $1 \leq k \leq n$. Aggregation of data instance from time instance t onwards is obtained using this λ weighted LSTMs

$$\hat{y}_t = \frac{1}{n}\sum_{k=1}^{n} \lambda_k L_k(x_t) \tag{8}$$

where, $\hat{y}_t$ is the final prediction of weighted aggregation (Fig. 3).

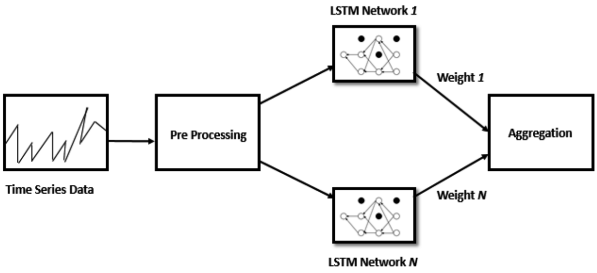


Fig. 3. Weighted aggregation of LSTM

4 Result and Analysis

The analysis was conducted on the Yahoo! Webscope [15], a time series benchmark dataset. It consist of four (A1–A4) benchmark data. In our analysis, real [1–3] and synthetic [1–3] dataset were used. Each dataset contains ≈1500 entries distributed among three fields (timestamp, value and is anomaly).

The Real dataset from A1 benchmark is based on real production traffic to the Yahoo! Properties while the synthetic is generated from artificial time-series with random seasonality, trend and noise. Former dataset has a periodic interval of one hour while the outliers in the later dataset are inserted at random positions. The underlying base architecture consists of three layered LSTM network [12]. A performance comparison of the proposed ensemble architectures weighted and m window with the baseline and simple mean aggregation model has been done using both real world and synthetic dataset. Various parameters taken for experiment are shown in Table 1.

Table 1. Parameters

Parameter	Value
Total instances	1500
$t - 1$	500
l	100
n	3
m	7

Figure 4 shows the result obtained from different aggregation techniques on the A2-synthetic 2 dataset when mapped with the actual data. Performance of single LSTM is shown in Fig. 4a, as the graph shows that this method is not able to predict the ground values hence gives an error $\varsigma = 173{:}439$. Simple mean aggregation on the other hand is able to predict more accurate values as shown in Fig. 4b. The convergence with the actual values is much better than the single model. It increases accuracy by ≈45% than single LSTM. Weighted aggregation further enhances the prediction accuracy by ≈67% and gives better prediction as shown in Fig. 4c than simple mean aggregation. An *m* window performs even better than weighted aggregation based LSTM by about ≈65%. As can be seen in the case of real data in the given table, giving preference to recent data window does not go well since the recent trend does not follows well with the underlying actual annotation.

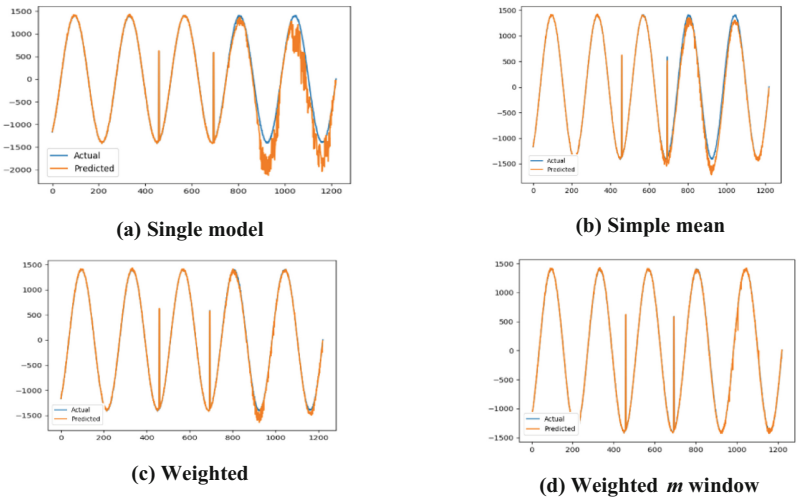


Fig. 4. Aggregation result of synthetic time-series

A comprehensive comparison of the results can be seen in Table 2. As evident from the result, weighted and m window clearly outperform baseline and simple mean aggregation. Among overall result, basic single model gives least performance followed by simple mean aggregation. On real dataset, weighted aggregation performs best among all as it is able to do a better approximation on inconsistent manual annotation. A window based aggregation on the other hand works best on synthetic data as extracting recent trend helps tackle random seasonality and noise.

Weighted and m window aggregation helps restrict model uncertainty, misspecification and inherent noise [16] to tolerable limits. Incorrect or ignored parameters based model leads to uncertainty. Dropout regularization helps solve uncertainty as it makes each of the models different from one another. The probability based neurons dropping in each network remains different. These thinned models makes the overall ensemble avoid wrongly trained or overfitted parameters.

Table 2. Predition error (ς)

Dataset	Base	Mean	Weighted	*m* Window
Real 1	0.076	0.036	0.031	0.034
Real 2	457.682	279.867	191.93	214.114
Real 3	0.28	0.123	0.113	0.117
Synthetic 1	39.514	34.062	25.467	18.788
Synthetic 2	173.439	93.867	39.596	12.662
Synthetic 3	173.521	128.994	106.737	77.616

A training dataset containing non-uniform samples that does not cover the whole sample space leads to model misspecification. Proposed methods considering the local performance of each thinned network and ne tune their importance helps build a generalized model which removes model misspecification. The uncertainty in data results in inherent noise. It is something which cannot be solved at the model level since the problem is with the data being given as input rather than the model itself. It is solved by systematically incorporating the feature of online Learning. It allows the model to learn with each prediction and fine tune its parameters for better result on upcoming dataset. Thus, the dynamic nature of the data is automatically considered during model training.

5 Conclusion

The study proves that single LSTM gives unsatisfactory predictions and ensemble of thinned LSTMs is able to increase the accuracy. The simple mean aggregation of ensemble LSTMs gives better results than single LSTM but with equal importance to all individual networks, the performance degrades. Proposed weighted aggregation method is able to define appropriate importance to each thinned network by analyzing their correctness. Especially in manual annotated real data, importance based on exhaustive past data counters the inconsistency in annotation. Weighted aggregation's performance degrades in presence of random outliers and noise. A m window based aggregation solves this problem by calculating thinned networks' importance based on last m data instances. Recent trend based importance factor is able to segregate outliers and noise accurately.

References

1. Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., Gao, R.X.: Deep learning and its applications to machine health monitoring: a survey. arXiv preprint arXiv:1612.07640 (2016)
2. Malhotra, P., Vig, L., Shro, G., Agarwal, P.: Long short term memory networks for anomaly detection in time series. In: Proceedings, p. 89. Presses universitaires de Louvain (2015)

3. Byeon, W., Breuel, T.M., Raue, F., Liwicki, M.: Scene labeling with LSTM recurrent neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3547–3555 (2015)
4. Sundermeyer, M., Schluter, R., Ney, H.: LSTM neural networks for language modeling. In: Thirteenth Annual Conference of the International Speech Communication Association (2012)
5. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
6. Dietterich, T.G.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45014-9_1
7. Gers, F.A., Schraudolph, N.N., Schmidhuber, J.: Learning precise timing with LSTM recurrent networks. J. Mach. Learn. Res. **3**(Aug), 115–143 (2002)
8. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: continual prediction with LSTM (1999)
9. Laptev, N., Yosinski, J., Li, L.E., Smyl, S.: Time-series extreme event forecasting with neural networks at uber
10. Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., Shro, G.: LSTM-based encoder-decoder for multi-sensor anomaly detection. arXiv preprint arXiv:1607.00148 (2016)
11. Assaad, M., Bone, R., Cardot, H.: A new boosting algorithm for improved time-series forecasting with recurrent neural networks. Inf. Fusion **9**(1), 41–55 (2008)
12. Guo, T., Xu, Z., Yao, X., Chen, H., Aberer, K., Funaya, K.: Robust online time series prediction with recurrent neural networks. In: 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 816–825. IEEE (2016)
13. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
14. Xiong, H.Y., Barash, Y., Frey, B.J.: Bayesian prediction of tissue-regulated splicing using rna sequence and cellular context. Bioinformatics **27**(18), 2554–2562 (2011)
15. Laptev, N., Amizadeh, S.: Yahoo anomaly detection dataset s5
16. Zhu, L., Laptev, N.: Deep and confident prediction for time series at uber. In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 103–110. IEEE (2017)