Recurrent Neural Networks' Configurations in the Predictive Maintenance Problems

LA Demidova

1 MIREA Russian Technological University, Moscow, Russia
E-mail: demidova.liliya@gmail.com

Received xxxxxx
Accepted for publication xxxxxx
Published xxxxxx

Abstract

The possibilities of various configurations of the recurrent neural networks in solving the problems of the maintenance performance based on the multidimensional time series have been investigated. The typical examples of the maintenance performance problems from technical and medical diagnostics have been considered. The configurations' examples of the one- and two-layer recurrent neural networks with the RNN, LSTM, and GRU neurons for the aircraft engine maintenance problems have been given, the graphical dependencies of the development results of the neural network models, the estimates of the development time, and the estimates of the accuracy indicator have been presented. The conclusions about the advantages of the recurrent neural networks with the LSTM and GRU neurons have been made.

Keywords: predictive maintenance, deep learning, machine learning, recurrent neural network, long short term memory, gated recurrent unit, neural network structure

1. Introduction

The problem of forecasting and classifying the state of objects is relevant in various application areas. A herewith, the state of object at any time can be described using the readings of the large number of sensors. Therefore, such problem can be considered as a non-destructive testing problem and is the problem of analyzing multidimensional time series. A solution to this problem can be obtained using the modern data mining tools and the machine learning technologies, for example, using the classifiers and regression models based on the SVM algorithm (Support Vector Machine algorithm) [1–3], RF algorithm (Random Forest algorithm) [4, 5], artificial neural networks [6–9]. In recent years, the deep learning technologies based on the neural networks [9] have been actively used in solving the problems of forecasting and classifying the state of objects on the basis of the large number of sensors.

Predictive maintenance is a domain where data is collected over time to monitor the state of the object with the goal of finding the patterns to predict the failures [10]. Predictive maintenance includes a variety of aspects, for example: failure prediction, failure diagnosis (root cause analysis), failure detection, failure type classification, and recommendation of mitigation or maintenance actions after failure.

The effective solution to the problems of the proactive intellectual maintenance can be obtained precisely with the use of the deep learning technologies.

Among the deep learning technologies, LSTM (Long Short Term Memory) and GRU (Gated Recurrent Unit) networks, which are essentially recurrent networks (RNN), are especially appealing to the predictive maintenance domain due to the fact that they are very good at learning from sequences [11, 12]. This fact allows the use of time series data in the development of the neural networks, operating with longer periods of time from the past to detect the failure patterns.
2. Formulation of the typical problems

The typical examples of the proactive intelligent maintenance problems are the following problems, mentioned below. The first two of these problems belong to the sphere of technical diagnostics, the third problem belongs to the sphere of medical diagnostics.

The first problem can be formulated with the aim to predict the remaining useful life of aircraft engines. The created template must use the aircraft sensors’ values to predict when an aircraft engine will fail in the future so that maintenance can be planned in advance [13–15].

The experimental dataset, which can be used to test new methods of maintenance in this sphere, contains the engine id for 100 engines, the cycle number of a specific engine (a herewith, the number of the last cycle is the point of failure of stable operation of the equipment); the initial wear and production differences (3 parameters in total), the readings of 23 sensors during the engine operation over the operation cycle. A herewith, each engine can have its own last cycle describing the point of failure of stable operation of the equipment.

The second problem can be formulated with the aim to propose the early failure detection methods for hard disk drives in order to improve storage systems availability and avoid data loss [16–18]. Failure prediction in such circumstances would allow for the reduction of downtime costs through anticipated disk replacements. Hard disk failure prediction plays an important role in reducing equipment downtime and improving service reliability. Many of the known methods, which applied for this problem, consider it as the classification problem and perform the incipient failure detection thus not allowing for proper planning of the maintenance problems. Others perform well only under a limited prediction horizon. It is actually to develop the methods, which can be capable to directly predict the remaining useful life (RUL) estimation for the hard disk drives based on SMART parameters. These methods must be able to predict the failures in both long and short term intervals by leveraging the capabilities of the LSTM and GRU neurons.

The experimental dataset, which can be used to test new methods of maintenance in this sphere, is comprised of the daily observation of 92,348 hard disk drives (HDD) during some years. These observations contain information regarding the serial number, model, capacity, fault, and 90 SMART attributes of each device. According to the Backblaze Company, a device is labeled as faulty if it stops working (does not turn on or does not receive commands), or if SMART self-test fails for attributes 5, 187, 188, 197, or 198. Most HDD models do not report all SMART attributes. In this case, the values not reported are left blank. In addition, different manufacturers and device models may report different attributes. All this information is consolidated in the experimental dataset.

The third problem can be formulated with the aim to propose types of therapeutic intervention for patients [19–21]. The heterogeneity among patients with various diseases usually leads to the different progression patterns and may required different types of therapeutic intervention. Hence, it is important to study patient subtyping, which is grouping of patients into disease characterizing subtypes. Subtyping from complex patient data is challenging because of the information heterogeneity and temporal dynamics. Obviously, that the capabilities of the LSTM and GRU neurons can be applied for analyzing longitudinal patient records.

One limitation of the standard LSTM- and GRU-neurons is that they cannot deal with irregular time intervals. But, the time irregularity is common in many healthcare applications. To illustrate this, we can consider patient records, where the time interval between consecutive visits or admissions varies, from days to months and sometimes a year. Such varying time gaps could be indicative of certain impending disease conditions. For example, the frequent admissions might indicate a severe health problem and the records of those visits provide a source to study progression of the condition. If there are months between the two successive records, the dependency on the previous visits or admissions should not play an active role to predict the current outcome.

As was mentioned above, the standard LSTM- and GRU-neurons are designed to handle data with constant elapsed times between consecutive elements of a sequence. Given that time lapse between successive elements in patient records can vary from days to months, the design of the standard LSTM- and GRU-neurons may lead to nonoptimal performance. The novel LSTM-neuron called Time-Aware LSTM-neuron (T-LSTM) to handle irregular time intervals in the longitudinal patient records was proposed in [19]. The subspace decomposition of the cell memory, applied in the T-LSTM-neuron, enables time decay to discount the memory content according to the elapsed time. A herewith, the patient subtyping model uses the proposed T-LSTM-neuron in an auto-encoder to learn a powerful single representation for sequential records of patients, which are then used to cluster patients into clinical subtypes. The architecture of the T-LSTM-neuron captures the underlying structures in the sequences with time irregularities.

The experimental dataset, which can be used to test new methods of maintenance in this sphere, describes so-called the Parkinson’s Progression Markers Initiative (PPMI), which is an observational clinical and longitudinal study comprising of evaluations of people with Parkinson’s disease (PD) with data for people with high risk, and data for people who are healthy. PPMI aims to identify biomarkers for the progression of Parkinson’s disease. PPMI data contains clinical and behavioral assessments, visualization data, and biological samples; therefore, PPMI is a unique PD archive [20].
PPMI is a longitudinal dataset with unstructured elapsed time. In [20], data on 654 patients with idiopathic PD or without PD were studied. The dataset contains the imputed missing values. Also, a unique sign form for categorical values and a data deviation coding of 1 and 0 are used. As a result, the dataset contains 15636 records for 654 patients with an average sequence length of 25 (the minimum sequence length is 3). In [21], data were also classified into features and goals, where the features are associated with the characteristics of the patient, and the goals correspond to the progression of PD.

A herewith, 319 features, such as motor symptoms/complications, cognitive functions, autonomic symptoms, psychotic symptoms, sleep problems, depressive symptoms and anxiety and depression scales in the hospital, and 82 goals, which are associated with motor signs, motor symptoms, cognition and others non-motor factors, were identified [21]. In this dataset, elapsed time is measured in months, and the gap between consecutive patient records is from 1 month to almost 2 years.

It is possible to train and evaluate the regression models, he binary or multi-classification models for all these problems.

In the future, the first problem will be considered in detail.

3. Approaches to solving the problems of proactive intellectual service

As the analysis of literature shows, in the context of working with TSs, the recurrent neural networks have been actively used. A herewith, various modifications of the recurrent neural networks’ models, such as the LSTM network and the GRU network, allows more efficient work with network memory, and, in particular, to solve the so-called problem of the vanishing gradient, which in turn allows them to learn more efficiently (minimizing the problem of overfitting) and solve classification, prediction and analysis problems.

It should be note, that Microsoft provides a template (as part of the Azure Machine Learning offering) which helps data scientists easily build and deploy a predictive maintenance solution [15].

This predictive maintenance template focuses on the techniques used to predict when an in-service machine will fail, so that maintenance can be planned in advance. It allow to solve the following problems [15].

Regression: Predict the Remaining Useful Life (RUL), or Time to Failure (TTF).

Binary classification: Predict if an asset will fail within certain time frame (e.g. days).

Multi-class classification: Predict if an asset will fail in different time windows: e.g., fails in window \([1,w_0]\) days; fails in the window \([w_0+1,w_1]\) days; not fail within \(w_1\) days.

The template includes a collection of pre-configured machine learning modules, as well as custom R scripts in the Execute R Script module, to enable an end-to-end solution from data processing to deploying of the machine learning model. The same solutions can be found in Python.

In this paper we consider various configurations of one- and two-layer networks on the basis of the RNN-, LSTM- and GRU-neurons.

4. Experimental results

The experimental studies were performed for the problem of the aircraft engines’ maintenance using the Turbofan Engine Degradation Simulation Data Set [13]. The problem of binary classification of the aircraft engines states was performed. It was necessary to decide: the engine of the considered type is broken or the engine of the considered type is in the working condition during the \(q\)-th cycle \((q = 1, 2, ...\).

In order to determine the class of the aircraft engine failure, it is necessary to develop the neural network model and train it at the dataset containing both information about the trouble-free work of the engine and information that a failure has occurred during the work.

For training the neural network model, the \(PM\) (Predictive Maintenance) dataset was used. It was made publicly available by NASA Ames Research Center [13] and contains: the training set \(PM_train\) with data on the engine work before the failure; the test set \(PM_test\) with data on the engine work without registering the failure event; the \(GTD\) (Ground Truth Data) set with information about the remaining cycles before failure for each engine test set. A herewith, the training and test sets have the similar structure.

The \(PM_train\) set contains information from several multidimensional time series generated according to the readings of various sensors. It is assumed that each engine starts with varying degrees of the initial wear and production, and this information is unknown to the user. In the training set \(PM_train\), it is stated that the engine is working normally at the beginning of each time series. At some point in time during the working cycles, the engine readings begin to deteriorate. Upon reaching the set threshold value, the engine is considered as the unsafe for further work. In other words, each last cycle in each time series can be considered as the failure point of the corresponding engine. One part of this set is used as the training subset, and another part is used as the validation subset during the training of model.

The \(PM_test\) set also contains information from the several multidimensional time series generated by the readings of various sensors. A herewith, various degrees of wear and deviation at the start of the engine work are also assumed. It is used to estimate the model after training.

At the beginning of each time series, the engine runs normally. At some point in time, engine performance starts to
deteriorate, but the last duty cycle for the particular engine is not necessarily a failure point (that is, this set does not show how many cycles this engine can run before it fails).

The neural network model must learn to determine the state when the readings of the engine will be considered as the unsafe for further work.

GTD data determines the number of remaining duty cycles for the engines in the test set.

The development of the neural network model and its training were carried out in Python using the Keras library. This language is widely used for data analysis, and the Keras library is designed specifically for working with the neural networks model. A herewith, the Google Colab was used to minimize the time costs. In general, this allowed to reduce the time costs by 5–6 times.

Previously, on the basis of the training dataset \textit{PM}$_{\text{train}}$, the training set \textit{Train} was created by supplementing the training dataset \textit{PM}$_{\text{train}}$ with values for the additional characteristic: the label that defines the class label (“0”, if the engine is OK; “1”, if the engine is broken). The values of the additional characteristic were determined taking into account the information that the last cycle determines the engine failure point.

During the experiments, the comparative analysis of the different recurrent neural networks configurations was carried out. We considered one-level and two-level configurations (table 1). Figures 1–9 show the graphic dependencies for the model \textit{Accuracies} and the model \textit{Losses} for these configurations.

The structure of the the neural network models in the most cases was defined as follows:

- the \textit{Recurrent layer} with 100 neurons;
- the \textit{Dropout layer} with parameter equal to 0.2 solving the problem of overfitting (it determines the share of the excluded neurons, that is, the neurons that do not contribute to the learning process);
- the \textit{Dense layer} with one neuron and the sigmoidal activation function for solving the classification problem.

If it is necessary, another the \textit{Recurrence layer} and the associated \textit{Dropout layer} are added.

The \textit{Recurrent layer} can be defined using the RNN-, LSTM- or GRU-neurons.

Only one model, the two-layer LSTM+LSTM (100+50 neurons) model, has 50 neurons at the second layer. Figures 10 and 11 show the structure of this model and the form of the output shape with the number of parameters at the each layer respectively [14].

| Model structure | Time (Train) | Accuracy (Train) | Accuracy (Test) |
|-----------------|-------------|-----------------|-----------------|
| one-layer RNN (100 neurons) | 6 min 36 s | 0.995650 | 0.989247 |
| two-layer RNN+RNN (100+100 neurons) | 11 min 10 s | 0.995584 | 0.967742 |
| one-layer GRU(100 neurons) | 12 min 43 s | 0.994242 | 0.978495 |
| two-layer GRU+GRU (100+100 neurons) | 26 min 15 s | 0.996865 | 0.978495 |
| one-layer LSTM (100 neurons) | 19 min 46 s | 0.985158 | 0.978495 |
| two-layer LSTM+LSTM (100+50 neurons) | 30 min 30 s | 0.993922 | 0.989247 |
| two-layer LSTM+LSTM (100+100 neurons) | 30 min 45 s | 0.979720 | 0.967742 |
| two-layer LSTM+GRU (100+100 neurons) | 28 min 56 s | 0.996685 | 0.978495 |
| two-layer GRU+LSTM (100+100 neurons) | 28 min 43 s | 0.995970 | 0.978495 |

Figure 1. The graphic dependencies for the one-layer RNN model.
During the experiments, we used the \textit{Adam} algorithm as the optimization algorithm [22]. Also, we considered the \textit{Accuracy indicator} as the objective function, and the \textit{binary cross-entropy function} which returns the classification error as the loss logistic function \textit{Loss}. We used 100 epochs to observe the model development process.

Table 1 shows the model structures, time costs, and values of the \textit{Accuracies indicator} at the training and test (validate) sets.

It is known, that the results of training the neural network substantially depend on the correct initialization of the neuron
weights. A herewith, it is required that the found solution will always be close to suboptimal.

The minimum development time, as was expected, corresponds to one-layer and two-layer RNN network models (figure 1 and 2). A herewith, it is very difficult to track overfitting in the case of one-layer RNN network (figure 1) when overfitting occurs and say when it is advisable to perform the early stopping of the learning process. In this context, one-layer GRU (figure 3) and LSTM (figure 5) network models behave noticeably better. The two-layer RNN network model (figure 2) also behaves better in this context than the one-layer RNN network model (figure 1), but it has one of the worst value of the Accuracy indicator at the test set.

The two-layer GRU network model (figure 4) allows making the stopping earlier than the one-layer GRU network model (figure 3) halving the number of epochs (about 30 epochs against 60).

The LSTM network models require fewer epochs for the early stopping: about 40 epochs for the one-layer LSTM network model (figure 5) and about 30 epochs for the two-layer LSTM network model (figure 6 and 7). A herewith, the time costs increase slightly. Selection of the optimal number of neurons in the second layer (figures 6 and 7) requires additional attention: it is important to minimize the number of neurons (and, therefore, the time costs on developing the model) without impairing the Accuracy indicator of the model.

The hybrid network models (figures 8 and 9) work similarly in the context of accuracy, but, at first glance, the two-layer LSTM+GRU network model (figure 9) allows the earlier stop, however, the additional research is needed in this direction.
Figure 7. The graphic dependencies for the two-layer LSTM (100+100 neurons) model.

Figure 8. The graphic dependencies for the two-layer GRU+LSTM model.

Figure 9. The graphic dependencies for the two-layer LSTM+GRU model.

A comparative analysis of the results of training of the neural networks models shows the equal or close values of the Accuracy indicator for the GRU- and LSTM- network models. A herewith, the learning speed of the network model based on the GRU-neurons exceeds the learning speed of the network model based on the LSTM-neurons (as for one-layer network model as for two-layer network model).
Figure 10. The form of the output shape with the number of parameters at the each layer of the two-layer LSTM+LSTM (100+50 neurons) model.

| Layer (type) | Output Shape | Param # |
|--------------|--------------|---------|
| lstm_1 (LSTM)| (None, 50, 100) | 50400 |
| dropout_1 (Dropout)| (None, 50, 100) | 0 |
| lstm_2 (LSTM)| (None, 50) | 30200 |
| dropout_2 (Dropout)| (None, 50) | 0 |
| dense_1 (Dense)| (None, 1) | 51 |

Total params: 8,451
Trainable params: 8,451
Non-trainable params: 0

None

Figure 11. The structure of the two-layer LSTM+LSTM (100+50 neurons) model. Moreover, these neural networks solve the problem of overfitting and early stopping during the development of the neural network.

5. Conclusion

The experimental results show that the LSTM- and GRU-networks demonstrate their high efficiency in solving the classification problem of the probable class of failure in the work of aircraft engines. It is planned to explore the possibilities of these networks to solve the classification problem of the probable class of failure of equipment taking into account the aspects of multicriteria optimization.

References

[1] Vapnik V 1998 Statistical Learning Theory
[2] Demidova L A, Nikulchev E V and Sokolova Y S 2015 International Review on Modelling and Simulations 4 446–57
[3] Demidova L, Nikulchev E and Sokolova Y 2016 International Journal of Advanced Computer Science and Applications (IJACSA) 7 16–24
[4] Scheibelhofer P, Leditzky G, Berger M, Schrottner P, Gleispach D and Hayward R 2011 Statistical Methods Applied in Microelectronics
[5] Elghazel Wiem, Medjahi Kamal, Zerhouni Noureddine, Babi Jacques, Farhat Ahmad, et al. 2015 IEEE Aerospace Conference pp 1–9
[6] Hopfield J and Tank D 1985 Biological Cybernetics 52 141–52
[7] Rosenblatt F 1992 Principle of neurodynamics
[8] Williams R and Zipser D 1989 Neural Computers 1 270–80
[9] Sigov A S, Andrianova E G, Zhukov D O, Zykov S V and Tarasov I E 2019 Rossisskijy tekhnikhicheskijy journal (Russian Technological Journal) 1 5–37 DOI: 10.32362/2500-316X-2019-1-1-5-37
[10] Ding H, Yang L and Yang A Zh 2019 IEEE Access 7 10.1109/ACCESS.2019.2933676
[11] Hochreiter S and Schmidhuber 1997 Neural computation 8 1735–80
[12] Aydn O and Guldamlasigilo S 2017 4th International Conference on Electrical and Electronic Engineering (ICEEE) doi: 10.1109/ICEEE2.2017.7935834
[13] Saxena A and Goebel K NASA Ames Prognostics Data Repository NASA Ames Research Center, Moffett Field, CA https://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/
[14] https://github.com/Azure/lstms_for_predictive_maintenance
[15] https://gallery.azure.ai/Collection/Predictive-Maintenance-Template-3
[16] Lima F D S, Amaral G M R, Leite L G M, Gomes J P P and Machado J C 2017 Brazilian Conference on Intelligent Systems pp 222–7
[17] Anantharaman P, Qiao M and Jadav D 2018 IEEE International Congress on Big Data (BigData Congress) pp 251–4
[18] https://www.backblaze.com/b2/hard-drive-test-data.html
[19] Baytas I M, Xiao C, Zhang X, Wang F, Jain A K and Zhou J 2017 Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining pp 65–74
[20] Che Ch, Xiao C, Liang J, Jin B, Zhou J and Wang F 2017 Proceedings of the 2017 SIAM International Conference on Data Mining pp 198–206
[21] Dinov I D, Heavner B, Tang M, Glusman G, Chard K and Darcy M et al. 2016 PLoS ONE https://doi.org/10.1371/journal.pone.0157077
[22] Kingma D P and Ba J L 2014 Preprint/arXiv.org № 1412.6980