Generating Proactive Decisions Using a Method Based on LSTM and Classification

Maxim Shcherbakov[1], Alexey Golubev, Van Cuong Sai
Computer Aided Design Department
Volgograd State Technical University
Volgograd, Russia
[1]maxim.shcherbakov@vstu.ru

***Abstract*— Increasing of complexity of distributed multi-objects system-of-interest leads to the need to revise the existing mechanisms and methods by which decision-making mechanisms for securing these systems are provided. One of the foremost areas of improving support processes is the use of intelligent systems. The use of these systems allows to move from planned provision to functioning according to the state, taking into account the prediction of changes in system states. In the framework of this study, the problem of synthesis of predictive management solutions is considered to provide effective support for the operation of complex multi-object systems in the concept of proactive computing.**

Keywords— proactive decision support; LSTM; RNN; classification

I. INTRODUCTION

According to best practices in system engineering, the main function of supportive systems is to support the proper operation of systems-of-interest during the certain phase of life cycle [1]. In case of system complexity or if system-of-interest is large scaled, the supportive systems need meet requirements according to complexity of a systems-of-interest. For instance, when a systems-of-interest is a complex multi-object distributed system, like a pipeline system. The effectiveness of supporting system during the operational phase is expressed through a system-of-interests owning cost. Therefore, the objective of the supportive system is either to increase the duration of the operational phase without changing the expected costs of ownership, or to reduce owning cost. A complex multi-object distributed systems-of-interest design process forms requirements for supportive systems across all life cycle phases.

The increasing complexity of distributed multi-object system-of-interest leads to revise existing mechanisms for implementing decision support systems and decision-making procedures. One of the directions for improving support processes is the development and implementation of intelligent systems that perform the functions of predictive analytics and predictive maintenance [2]. These systems allow you to move from rough planned provisioning to ensuring the functioning of the state, taking into account the prediction of changes in system states. For this class of systems, we will use the name – proactive decision support systems, based on the ideas of proactive computing [3, 4]. Classically, proactive systems implement the scheme: detection – forecast – solution – action [3]. The scheme provides for the detection of a proactive situation, an event is a trigger for decision-making initialization. With this approach, we can identify a number of fundamental issues. The first issue is the timely determination of the proactive situation by the providing system, based on the analysis of data on the state of the system-of-interest. The second issue is to improve the mechanisms for forecasting changes in the state of the system-of-interest under various operating conditions. The third issue is connected with the synthesis and selection of the management decision aimed at improving the efficiency of the operation of the system-of-interest. In the framework of this study, the third problem is considered.

The development of ubiquitous systems and the reduction in the cost of data collection systems on the one hand allow us to collect new data with high frequency about multi-object distributed system-of-interest. On the other hand, there is the problem of preparing, assessing data quality and adapting decision support mechanisms [5]. In this case, a review of decision support methods is required.

In the framework of this study, the problem of synthesis of predictive management solutions is considered to provide effective support for the operation of complex multi-object systems in the concept of proactive computing. The main contribution of the article is a new approach to the synthesis of predictive management solutions, based on (i) analysis of accumulated data on solutions in the past, (ii) based on the use of the mechanism of recurrent neural networks and classifiers, which allowed solving the problem of not only choosing a specific control action, but also the time of application of the impact.

II. BACKGROUND

The problem under consideration relates to the problem of the formation (synthesis) of control actions. The implementation of approaches refers to the tasks of ensuring the functioning of systems, informing and making quick decisions. Represent a control action as a sequence of solutions, each of which can be represented as a tuple d =

The reported study was partially supported by RFBR research projects 16-37-60066 mol_a_dk.

978-1-5386-5824-6/18/$31.00 ©2018 IEEE

<*a*,*t*,τ,*h*>, where *a* – action from a given set of actions, *t* – action time, τ – duration of the action and *h* – intensity of the action (a characteristic proportional to the resources expended). Note that there exists an element of the set *A* characterizing the absence of an action (no action) [3].

This task can be considered as a decision or planning task, which is solved using, for example, dynamic programming approaches or even reduced to the task of scheduling [6]. However, in this case a formalized statement of the problem is required, which is not the case in the case of the study under consideration.

One of the alternative ways to solve the problem is to apply approaches based on machine learning. If a training sample with characteristic and response values is given, then a similar problem can be reduced to the classification problem. In a number of works, the application of algorithms of machine learning for the formation of control actions in tasks of predictive maintenance [7, 8]. It should be noted that often the algorithm of machine learning forms a control action, but does not solve the problem of determining the time for the application of the impact.

III. A METHOD

Let there be a multiobject distributed system – system of interest, which is described by the set of features { *x* }. There is a mechanism for collecting features values in time and fixing control actions in the format <*a, r, ts, tk*>, where *a* – action from a set of admissible actions, *r* – expended resources (can be reduced to one value), *ts* – timestamp of the beginning of the application of the action, *tk* – timestamp of the end of the application of the action. Let the *value function* is given as a function of the state of the object at some point in time. The value function can be calculated for an action (in this case it takes negative values). It is necessary to develop a mechanism that creates predictive management actions which maximize the value over defined planning horizon.

To solve the problem, it is necessary: (i) to determine the time of the need for decision-making (proactive situation identification); (ii) generate management decisions in accordance with formalization and (iii) choose the most preferable solution. The first task is solved by methods of identifying changes in the behavior of objects.

In this case it is necessary to solve two problems:

1. determining the change in the state of the system-of-interest;

2. synthesis of a management decision to transfer to a target state as a pair <a,t>, where a – action from a variety of possible actions, t – time of application of action a.

To generate management action, we propose method based on a two-component model involving the recurrent LSTM neural network and classification network The generation of control actions is reduced to the classification problem on the basis of the input sequence of features values and the sequence of predicted values generated by the LSTM neural network.

The method is described by the following sequence of steps:

- Get a marked sample with management decisions labels. In this case, we generate data using RNG and several classes for which the boundaries of their change are defined.
- Create a prediction model based on the LSTM neural network, in which the input is a fragment of a time series (a sequence of values of a given length), the output is also a sequence of a given length.
- Set h – the number of discrete samples characterizing the length of the input vector, and implicitly specifying a discrete count of the application of the control action.
- Convert the data to a normalized form and divide them into two samples: training and validation.

3. Create a classification model whose inputs are: (i) a sequence of features values, (ii) the predicted value of features (the result of LSTM prediction), outputs the degree of confidence in the use of a particular control action (the number of outputs is equal to the number of possible control actions).

To predict the signal of the working system, a two layers network was built using the Sequential model. The first layer is a long short-term memory (LSTM) consisting of 4 cells and N inputs. The second layer is the Dense block with M outputs. To optimize the forecasting model, the adam optimizer and the MSE loss function were used. The scheme of the forecast model is shown in Fig. 1 for N = 30 and M = 1.

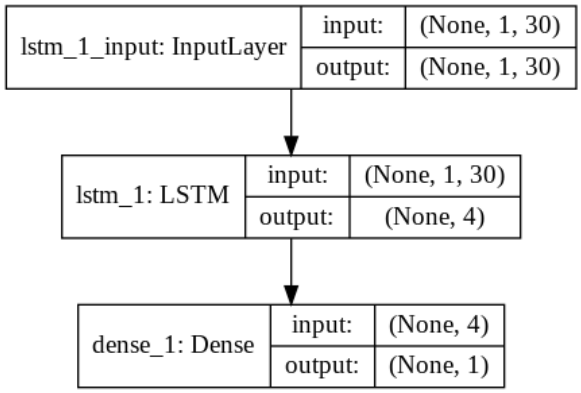

Fig. 1. Scheme of the layers of the prediction model with N = 30 and M = 1

To predict the signal of the system, a network based on the Sequential model using a single Dense layer was used. The number of inputs in the model corresponds to the number of samples in the signal block, and the output number is the number of classes. To fit the classifier, the activation function of the Dense layer – softmax, the optimizer – adam, the loss function – categorical_crossentropy was used. The scheme of the classifier is shown in Fig. 2.

Data for training was generated using a random number generator in a given range for each of the classes [9]. Both models (forecasting and classification) were trained in 10 epochs.

To obtain a synthetic data set, a signal generation function was written that takes a dictionary describing each of the classes, that is, the description of the "generators" is given as follows:

{ 'm1': [6, 10], 'm2': [3, 5], 'm3': [0, 1], 'm4': [0, 2], 'm5': [0, 3] }

where m1..m5 – name of the class, and the list items are the left and right boundaries for the range of values generation. The mode of operation of the system is given in the form of a list of tuples.

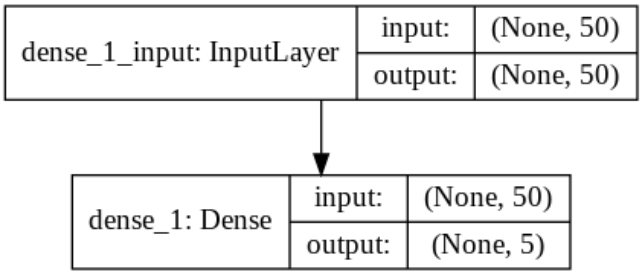

Fig. 2. Scheme of classifier layers for a block of 50 samples.

The description of the signal is as follows:

[('m1', 50), ('m2', 20), ('m3', 15), ('m2', 30), ('m4', 10), ('m5', 20), ('m2', 70),

('m4', 30), ('m2', 40)]

where m1..m5 – class of the signal, and the second value in the tuple describes the number of samples that need to be generated.

IV. Results

Experiments were carried out using the data generation system and the Keras deep learning framework [10]. For the first network, the sequence of layers includes: (i) a LSTM network with N inputs and M outputs, where N is the number of considered past values, M is the number of predicted values, and (ii) a neural network with M inputs and L outputs (L – horizon of forecasting). For the second network, the Dense layer with N1 and N2 inputs and K outputs, where K is the number of potential actions was used.

The forecast model was trained using a signal generator. The behavior of the system was described in terms of the presented generator. The signal itself, which is shown in Figure 3, was generated using the presented generator. The signal was divided into two samples: training and validation. The model error on the test sample is of the order of 10E-3 for 10 epochs.

To train the classifier, several sets of signals were generated for each of the presented classes, using the generator described earlier. The training was done on 45,000 samples for 10 epochs. The accuracy of the classifier on a validation sample of 5000 samples averaged 98%.

The general results of experiments on synthetic data showed high accuracy of prediction (maximum MAPE error of 0.19%) and classification (maximum classification error of 6.78% in the shortest sequence). Fig. 4 shows an illustration of the prediction and classification problem.

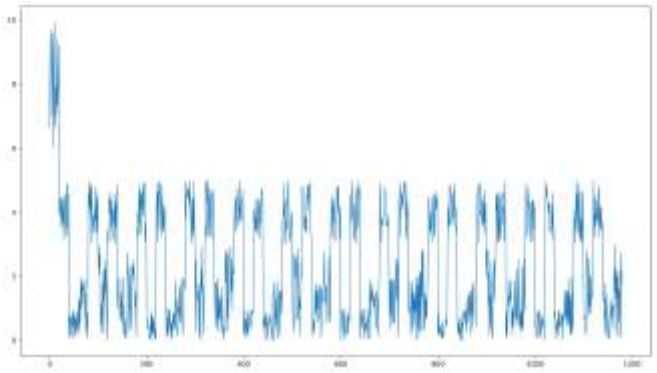

Fig. 3. The generated signal

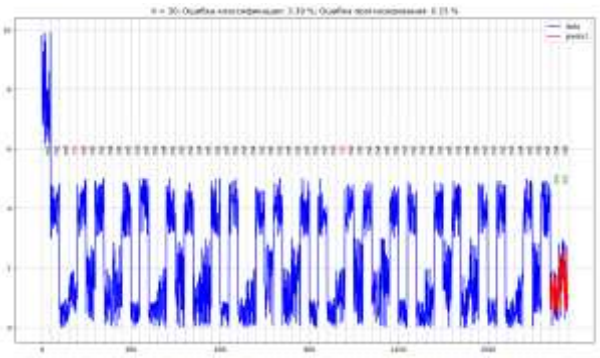

Fig. 4. The operating mode of the system (blue line), the predicted mode of operation (red line), the mode of operation of the system m1..m5 (where the red text is an error, the green text is a coincidence)+

Models of forecasting and classification were trained at different block sizes. The accuracy of the models for different block sizes is shown in Table 1.

TABLE I. Accuracy of models with different block size

Block size (h)	Classification error, %	Forecast error, %
20	6.78	0.17
30	3.39	0.15
40	3.39	0.17
50	0.00	0.17
60	1.69	0.19
70	1.69	0.17
80	0.00	0.17
90	1.69	0.14

In contrast to models based on neural network methods, a classifier model was also built using the scikit-learn library, where the OneVsRestClassifier multiclass classifier and the classifier C-Support Vector Classification (SVC) of the libsvm library were used. Training was also conducted using the above-described signal generator, based on several classes. To assess the quality of the resulting classifier, the AUC/ROC characteristic was used. On a relatively small set of input data for training about 500 samples (100 samples per class), the following characteristic was obtained for 5 classes. The results are shown in Fig. 5.

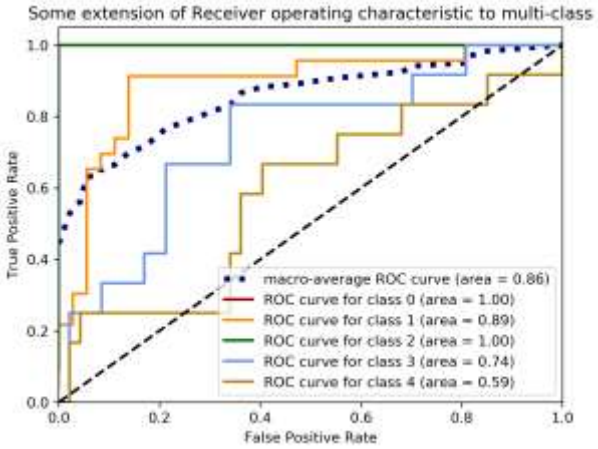

Fig. 5. ROC for classification

This classifier showed quite good accuracy on this data set. This model can also be used, replacing or sharing with the Dense layer classifier, if you are training on a larger set of data.

Training of all models was made on a laptop running the operating system Arch Linux x86_64 with an Intel (R) Core (TM) i5-5200U CPU with a frequency of 2.20GHz.

V. CONCLUSION

This approach is aimed at finding control actions from a finite set. Specifying the lengths of the sequences of characteristic values (both real and predictive) as hyperparameters, it is possible to determine the start and end time of the application of the control action.

But there are several important features that will need to be investigated in further work, namely:

- Working with a variable length signal is a very important feature for this system. Since the operation of real equipment cannot be described in constant-length data blocks. To solve this problem, we propose to consider the sliding window method.
- Check the developed method on the real data received from the equipment. Although synthetic tests show good results. When working with a real device everything will not be so perfect.

ACKNOWLEDGMENT

REFERENCES

[1] ISO/IEC/IEEE 15288:2015 Systems and software engineering - System life cycle processes, Technical Committee. ISO/IEC JTC 1/SC 7 Software and systems engineering, 2015.

[2] Arnott D., Pervan G. A critical analysis of decision support systems research revisited: the rise of design science. Journal of Information Technology. 2014. Vol 29(4), pp. 269–293.

[3] Engel Y., Etzion O. Towards Proactive Event-Driven Computing. Proceedings of the 5th ACM International Conference on Distributed Event-Based System. 2011, pp. 125–136.

[4] Tennenhouse D. Proactive computing, Communications of the ACM. 2000,vol. 43(5), pp. 43-50.

[5] Shcherbakov M.V., Brebels A., Shcherbakova N.L., Kamaev V.A., Gerget O.M., Devyatykh D. Outlier detection and classification in sensor data streams for proactive decision support systems. Journal of Physics: Conference Series. 2017. Vol. 803(1), https://doi.org/10.1088/1742-6596/803/1/012143

[6] Arnolds I.V., Gartner D. Improving hospital layout planning through clinical pathway mining. Annals of Operations Research. 2018, vol. 263(1), pp. 453-477.

[7] Saxena A., Goebel K. Turbofan Engine Degradation Simulation Data Set. NASA Ames Prognostics Data Repository. NASA Ames Research Center. Moffett Field, CA. 2008.

[8] Mathew V., Toby T., Singh V., Rao B.M., Kumar M.G. Prediction of Remaining Useful Lifetime (RUL) of turbofan engine using machine learning. Circuits and Systems (ICCS) 2017 IEEE International Conference. 2017. Pp. 306-311.

[9] Van Phu T., Shcherbakov M., Nguyen T A. EVGEN: A framework for event generator in proactive system design. 2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA). Chalkidiki. 2016, pp. 1-5.

Keras: The Python Deep Learning library https://keras.io/