DOI: 10.15514/ISPRAS-2025-37(5)-4

Round-Trip Time Prediction Using Machine Learning Methods

[1,2] *I.A. Stepanov, ORCID: 0009-0003-1964-5001 <ivan_mipt@ispras.ru>*
[1] *R.E. Ponomarenko, ORCID: 0009-0009-5741-3627 <rerandom@ispras.ru>*
[1] *D.R. Golovash, ORCID: 0009-0006-5552-4428 <golovash@ispras.ru>*
[1] *A.Y. Pokidko, ORCID: 0009-0008-8981-8429 <a.pokidko@ispras.ru>*
[1,2,3,4] *A.I. Getman, ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*

[2] *Moscow Institute of Physics and Technology (National Research University), 9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia.*

[3] *National Research University «Higher School of Economics», 20, Myasnitskaya ulitsa, Moscow, 101000, Russia.*

[4] *Lomonosov Moscow State University, 1, Leninskie Gory, Moscow, 119991, Russia.*

Abstract. The congestion control algorithms in the TCP protocol use RTT predictions indirectly or directly to determine congestion. The main algorithm for predicting RTT based on a weighted moving average is the Jacobson Algorithm. However, this algorithm may not work quite efficiently if the RTT is subject to a heavy-tailed distribution. In this paper, we propose an RTT prediction method based on supervised learning in both the offline and online cases. The results show improvement in the performance of algorithms based on supervised learning compared to the classical Jacobson algorithm in terms of MAPE, MAE, and MSE metrics. In addition, the high efficiency of online learning in comparison with offline learning in the case of data drift is shown.

Keywords: TCP; RTT prediction; online learning; Adaptive Random Forest regression.

For citation: Stepanov I.A., Ponomarenko R.E., Golovash D.R., Pokidko A.Y., Getman A.I. RTT prediction using offline and online learning. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 5, 2025, pp. 53-66. DOI: 10.15514/ISPRAS-2025-37(5)-4.

Предсказание времени приема-передачи с использованием методов машинного обучения

[1,2] *И.А. Степанов, ORCID: 0009-0003-1964-5001 <ivan_mipt@ispras.ru>*
[1] *Р.Е. Пономаренко, ORCID: 0009-0009-5741-3627 <rerandom@ispras.ru>*
[1] *Д.Р. Головаш, ORCID: 0009-0006-5552-4428 <golovash@ispras.ru>*
[1] *А.Ю. Покидько, ORCID: 0009-0008-8981-8429 <a.pokidko@ispras.ru>*
[1,2,3,4] *А.И. Гетьман, ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25.*

[2] *Московский физико-технический институт, 141700, Россия, Московская область, г. Долгопрудный, Институтский пер., 9.*

[3] *Национальный исследовательский университет «Высшая школа экономики», 101978, Россия, г. Москва, ул. Мясницкая, д. 20.*

[4] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, г. Москва, Ленинские горы, д. 1.*

Аннотация. Время приема-передачи (RTT, Round-Trip Time) – время, которое требуется для отправки пакета от отправителя к получателю и возврата подтверждения, что пакет был получен. Алгоритмы управления перегрузками в протоколе TCP косвенно или напрямую используют предсказанные значения RTT для определения перегрузки сети. Основным алгоритмом для прогнозирования RTT на основе взвешенного скользящего среднего является алгоритм Джейкобсона. Однако этот алгоритм может работать не совсем эффективно, если RTT имеет распределение с тяжёлым хвостом, т.е. существуют редкие, но очень большие значения RTT. В этой статье мы предлагаем метод прогнозирования RTT, основанный на обучении с учителем, который может работать как в оффлайн режиме (с заранее собранной обучающей выборкой), так и в онлайн режиме (с поступлением данных в реальном времени и их последовательной обработкой). Полученные результаты показывают улучшение алгоритмов, основанных на машинном обучении, по сравнению с классическим алгоритмом Джейкобсона с точки зрения показателей MAPE, MAE и MSE. Кроме того, показана высокая эффективность онлайн обучения по сравнению с оффлайн обучением в случае дрейфа концепции или дрейфа данных.

Ключевые слова: транспортный протокол TCP; прогнозирование времени приема-передачи (RTT); онлайн-обучение; адаптивная регрессия случайного леса.

Для цитирования: Степанов И.А., Пономаренко Р.Е., Головаш Д.Р., Покидько А.Ю., Гетьман А.И. Предсказание RTT с использованием оффлайн и онлайн обучения. Труды ИСП РАН, том 37, вып. 5, 2025 г., стр. 53–66 (на английском языке). DOI: 10.15514/ISPRAS–2025–37(5)–4.

1. Introduction and Motivation

RTT (round-trip time) is the time required to send a data packet from the source to the recipient and back to the source. This is an important parameter in network performance. In addition, the retransmission timer (RTO) has an important role in the TCP protocol. This timer is set when sending a segment and its expiration serves as a congestion signal. The problem of choosing this timer is related to the fact that the RTT has a high variance from the point of view of a random variable, which significantly complicates the prediction of this value.

The prediction of RTT is an important component of congestion control algorithms (CCA). Packet loss-based CCAs such as TCP Reno and TCP Cubic indirectly use RTT information to determine congestion. In addition to loss-based CCA, there are CCAs that detect congestion directly from RTT: TCP Vegas, TCP Vegas-A. Therefore, for such methods, it is extremely important to accurately predict RTT one step ahead. Also, multipath technology has recently become very popular, allowing

the client to transfer data over multiple network paths. The scheduler, which determines the path to send the packet, makes decisions based on certain metrics, one of which is RTT. In this case, RTT prediction can also be very important.

They are usually based on the Jacobson algorithm, which predicts RTT using the moving average method. However, as some researchers have noted, the moving average method may not work well for values from distributions with a heavy tail, which may well include RTT. Therefore, a number of papers have been proposed that predict RTT using recurrent neural networks. Since recurrent neural networks require a large training dataset, its collection is an important component of the RTT prediction task. However, models of this class can often work inefficiently in terms of decision-making time, which can be critical in terms of congestion control.

In addition, due to the high variability of RTT, a model trained in one network environment (with a low RTT value) may be less effective in another network environment (with a high RTT value). This behaviour is due in part to data drift.

In order to avoid a drop in predictive ability during the transition from one environment to another, it makes sense to detect drift during model runtime, and in case of drift, online learning it based on new data.

Therefore, in this paper there is propose an online machine learning method with drift detection. The results obtained show an improvement in RTT prediction using this method compared to the Jacobson algorithm. At the same time, an improved prediction is observed in various network scenarios, in terms of the RTT value.

The rest of the article is structured as follows. Section II provides information on the structure of RTT and the classical methods of its measurement. Section III describes RTT prediction methods that use both probability distributions and machine learning. Section IV contains a statement of the problem of online learning and drift detection. Section V provides a description of our method. Section VI contains comparisons of the method implemented in this paper with the Jacobson algorithm.

2. Background

Using different concepts of RTT, it can be stated that:

$$RTT \approx delay_{propagation} + delay_{transmission} + delay_{queueing} + delay_{processing}$$

- $delay_{propagation}$ – the propagation delay is the time it takes for a signal to move from the sender to the receiver through physical media (such as cables or radio waves). It depends on the distance between the nodes and the speed of signal propagation in the environment.
- $delay_{transmission}$ – the transmission delay is the time required to transmit a data packet over a communication channel. The transmission delay depends on the packet size and bandwidth of the communication channel.
- $delay_{queueing}$ – the queue delay is the time during which a data packet is queued on the forwarding devices, waiting for the next packets to be transmitted.
- $delay_{processing}$ – the processing delay is the time required for packet processing on routers and end nodes. It includes the time required to process headers, check for errors, and perform other operations related to packet routing and processing.

It makes sense to consider RTT between sender and recipient as the sum of two main components: the constant component, which includes propagation delay and transmission delay, and the variable component, which includes queuing delay and processing delay. Queuing delay and processing delay are the main source of uncertainty in the prediction of RTT, as they depend on various components.

2.1 RTT measurement methods

To predict RTT using machine learning models, it is necessary to collect a dataset containing information about the RTT sequence. There are two ways to do this with ready-made tools.

The first is to use the ping command and send ICMP packets. However, ping does not always measure an accurate RTT. For example, when routers process ICMP packets during congestion, certain application flows may be prioritized. Thus, ICMP packets will generate RTTs that do not reflect the RTT that the priority traffic is encountering. In addition, some networks may block ICMP traffic, which also complicates the data collection process. The second way is to use the Wireshark tool: *tcp.analysis.rtt*. The third way to get an RTT value is by using TCP packet parameters such as Tsecr and Texp. However, in this case, the RTT accuracy will be limited to milliseconds.

2.2 The Jacobson algorithm

The first classical RTT prediction algorithm was the Jacobson algorithm, introduced in TCP Reno [1]. In this algorithm, the predicted RTT is subsequently used to calculate the RTO in the following form:

$$ERR = |(RTT_n - SRTT_{n-1})|$$
$$SRTT_n = \frac{7}{8}SRTT_{n-1} + \frac{1}{8}RTT_n$$
$$VAR_n = \frac{3}{4}VAR_{n-1} + \frac{1}{4}ERR$$
$$RTO = SRTT_n + 4VAR_n$$

Based on the moving average formula, we can see that:

$$SRTT_n = \frac{1}{8}RTT_n + \left(\frac{7}{8}\right)\cdot\frac{1}{8}RTT_{n-1} + \left(\frac{7}{8}\right)^2\cdot\frac{1}{8}RTT_{n-2} + \ldots$$

The usual estimate proposed by Jacobson works well in Gaussian distributed delay environments. However, as some researchers have noted, this algorithm may be inaccurate in environments with a different RTT distribution.

3. Related Work

There are two areas of work on RTT prediction: based on probability distributions and based on machine learning.

In several papers, RTT and, as a result, RTO are predicted based on the assumption that they are subject to a certain distribution. Thus, in [2], a method for approximate estimate of RTT was proposed based on the assumption that RTT is subject to the Weibull distribution. In [3], a method was proposed for a more detailed assessment of RTT based on the assumption that RTT is subject to a normal distribution. In [4], the authors proposed a method based on the calculation that the difference between neighboring values of RTT is subject to the Cauchy distribution. Using this assumption and Chebyshev's inequality, the authors can obtain the following estimate for the RTO:

$$RTO(k) = RTT(k-1) + \sqrt{\left(\frac{2\gamma\epsilon}{tan(\pi\phi)}\right) + \epsilon^2 - \gamma^2}$$

- γ– jitter dispersion
- ϕ– defined quality of service (QoS) parameter, which indicates the minimum fraction of time during which the prediction error is below the acceptable error ϵ.

These methods rely on assumptions about the distribution of RTT. However, the dynamic variability of RTT negatively affects the ability to accurately predict RTT in these methods, because the distribution of RTT can vary depending on the network environment.

A hybrid RTT prediction method based on geographical distance was proposed in [5]. The RTT prediction algorithm consisted of several stages. The first is an estimate of the distance between two IP addresses (sender and recipient). If the distance is less than 120 km, the RTT value was determined based on the database. If the distance is greater than 120 km, the RTT value was determined based on the trained model. The trained model was based on a decision tree that contained three features: Internet service provider, geographical distance between pairs of IP addresses and time of day. It is worth noting that distance is not always an informative feature, as it can change rapidly due to dynamic changes in network routes.

Recurrent neural networks have shown good predictive ability for predicting time series. As a result, a number of papers have appeared that predicted RTT based on previous values of RTT. The algorithm proposed in [6] has the following form:

$$ERR = |(RTT_n - SRTT_{n-1})|$$
$$SRTT_n = F(RTT_1, RTT_2 \dots RTT_{188})$$
$$VAR_n = \frac{3}{4} VAR_{n-1} + \frac{1}{4} ERR$$
$$RTO = SRTT_n + 4VAR_n$$

Here, F is a function implemented by a recurrent neural network. In [7], an RTT prediction method was proposed based on passive measurements collected at an intermediate node. The recurrent neural network (LSTM) was chosen as the prediction model. In [8], a lightweight version of the recurrent neural network GRU was proposed.

However, neural networks can require high computational costs, which is critical in the context of RTT prediction. Therefore, it makes sense to consider classical machine learning models (Random Forest, Linear regression).

It is worth noting that the RTT prediction study in the above papers was given only for the offline case. However, the efficiency of the algorithm in the offline and online case may vary greatly. Therefore, both offline and online scenarios will be considered in this paper.

4. Online learning

The task of online learning can be formulated as follows. Let's give a sequence of features and target values $(x_i, y_i)_{i=1}^n$. $a(x, w)$ - parametric model, $L(w, y)$ – loss function. At each step i, the following set of actions is performed:

- getting object features x_i
- the prediction is made based on the received object $a(x_i, w_{i-1})$
- getting y_i
- calculation of the loss function $L\big(y_i, a(x_i, w_{i-1})\big)$
- updating the weights of the model based on the loss function w_i

It is worth noting that incremental learning, unlike online learning, works with batches, while online learning uses only one object at each step. Otherwise, the two approaches are very similar in the context of the task under consideration.

4.1 Drift detection

Data drift is a phenomenon in which the statistical properties of the data used to train a machine learning model change over time. This means that the distribution of the input data in the real world

no longer corresponds to the distribution of the data on which the model was trained. Ultimately, due to this phenomenon, the accuracy of the model degrades.

More strictly, let there be some target variable and a set of features defining this variable. The drift is understood as a change in the distribution of the input data $P(X)$, the target variable $P(Y)$, or the relationship between them $P(Y|X)$ over time. It is worth noting that there are several types of drift detection in research.

Input data drift: let's give the initial distribution of input data (features) $P_0(X)$ and some distribution of data $P_t(X)$ at time t. It is said that there is a drift in the input data if:

$$P_0(X) \neq P_t(X)$$

Drift of the target variable (Label Drift): let's give the initial distribution of the target variable $P_0(Y)$and some target variable $P_t(Y)$ at time t. To say that there is a drift in the label data if:

$$P_0(Y) \neq P_t(Y)$$

Concept Drift: let's give the initial dependence distribution $P_0(Y|X)$ and $P_t(Y|X)$ at time t. To say that there is a concept drift in the data if:

$$P_0(Y|X) \neq P_t(Y|X)$$

There are a large number of ways to detect drift. These include statistical methods: the Kolmogorov—Smirnov test [9], the Chi-square test [10], the Darling-Anderson test [11], methods based on autoencoders [12], as well as methods based on the ARIMA model [13].

4.2 ADWIN

The ADWIN (Adaptive Windowing) [14] algorithm is a method that solves the problem of detecting changes in statistical characteristics of data, such as mean or variance. ADWIN uses the hypothesis of equality of the averages between different parts of the data window. If these hypotheses are rejected, it means that data drift has occurred.

The algorithm divides the window Winto two sub-parts: $W_0$and W_1. Then, for each part, the following are calculated: n_0, n_1- size of window $W_0$and $W_1\mu_0, \mu_1$- average values $W_0$and W_1. If the difference between the observed mean values $|(\mu_0 - \mu_1)|$exceeds ϵ_{cut}, the algorithm considers that the distributions in $W_0$and $W_1$are different, and deletes the old part $W_0$of the window. In this case, $\epsilon_{cut}$is calculated as follows:

$$m = \frac{n_0 \cdot n_1}{n_0 + n_1}$$

$$\delta' = \frac{\delta}{n}$$

$$\epsilon_{cut} = \sqrt{\left(\frac{1}{2m}\right) \cdot ln\left(\frac{4}{\delta'}\right)}$$

4.3 Adaptive Random Forest regressor

Adaptive Random Forest (ARF) [15-16] is an online learning algorithm that adapts to concept drift. The main idea of the algorithm is to have an ADWIN-based drift detector for each tree of a Random Forest. If the detector detects a change, the corresponding one is removed and retrained on the new dataset. Thus, the ensemble of trees adapts to the new distribution.

4.4 Online learning and drift detection

The general scheme of online learning used in this work is shown in Fig. 1.

The online learning process consists of several important parts: the main dataset, an Adaptive Random Forest, and a drift detector based on the ADWIN method. In the process of online learning, new objects are received at the input of the algorithm. The drift detector checks for drift between new objects and the main dataset, which is constantly being updated. If drift is detected, the Adaptive Random Forest is updated based on new data; if not, the Adaptive Random Forest remains unchanged. Thus, the model's stability to changing environmental conditions is achieved.

In this case, the ADWIN algorithm determines the drift for the normalized value: $|(y_{true} - y_{predict})|$. Thus, if the distribution of $|(y_{true} - y_{predict})|$changes significantly, the ADWIN algorithm detects the drift.

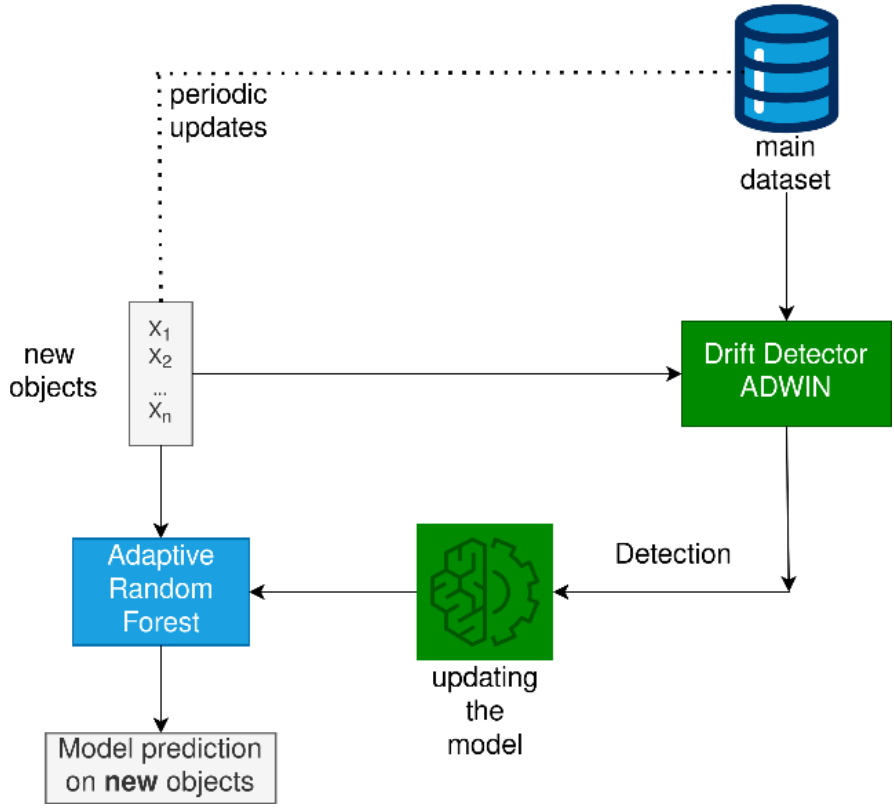

Fig. 1. Online learning and drift detection scheme.

5. Implementation

5.1 Problem formulation

From the point of view of supervised learning, the RTT prediction task is a regression task.

$$f: X \to Y$$

- X– features object
- Y- predicted RTT value

It makes sense to consider the following characteristics of a TCP stream as features.

RTT: In most studies, it is proposed to use sequential RTT values as features. This paper also examines these values for RTT prediction.

TTL: As noted earlier, the geographical distance between two hosts can change dynamically and is not always an informative feature in the RTT prediction task. However, it makes sense to use the TTL parameter, which is the IPv4 field of the packet header that specifies the maximum number of routers (hop count) through which the packet can pass before it is dropped. Each time a packet passes through the router, the TTL value decreases by one. Therefore, this parameter can be used as features for RTT prediction. In IPv6, the Hop Limit field is an analogue of the TTL parameter from IPv4. From the point of view of the problem under consideration, the Hop Limit and TTL are equivalent parameters.

Bytes in Flight: This value indicates how much data (in bytes) have been sent from the sender, but have not yet been confirmed by the recipient. The congestion control algorithm strives to maximize the use of the transmission channel so that the number of bytes in flight is approximately equal to BDP (Bandwidth-delay Product). Therefore, it can be stated that there is some connection between RTT and the number of bytes in the flight and use this feature in the task under consideration.

Thus, the following features vector is used for prediction RTT_n:

$$RTT_{n-1}\dots RTT_{n-k}, bytes_{n-1}\dots bytes_{n-k}, TTL_{n-1}\dots TTL_{n-k}$$

In this formula, k is a parameter that indicates the number of previous values used for prediction. The search for the optimal value of k, which preserves the high performance of the algorithm, will be described later in the paper.

5.2 Machine learning models

In this paper, the following machine learning algorithms were investigated within the framework of the problem under consideration: Linear regression with regularization, Decision Tree and Random Forest, as well as recurrent neural networks.

After examining the collected dataset, a high linear relationship was found between the predicted RTT and the features discussed above. This behavior motivates the use of linear regression with *L1* and *L2* regularization in the context of the task.

Random Forest is an algorithm that has proven itself well in working with data containing a large number of noises. In terms of RTT prediction, an abnormal value of this value caused by some external factors can be considered noise. Therefore, it makes sense to consider this model in the context of the task under consideration.

Recurrent neural networks have proven themselves well in the context of sequence prediction. Classical RNNs can have gradient attenuation problems when the network needs to remember information from the distant past: RTT prediction based on a large number of previous values. LSTM has a better ability to store information over long time intervals, but this model has a complex structure and may require high computing resources. The GRU model is simpler in terms of structure. In this paper, all three models for solving the RTT prediction problem will be investigated for a detailed analysis.

5.3 Training data

In the training process, the following dataset was collected, simulating 3 situations: a user is uploading files, an online game, and a regular web interaction.

For a wide variety of data and, consequently, for more efficient prediction of the model in a variety of network scenarios, a dataset was collected in a wide range of network characteristics: bandwidth from 10 Mbps to 100 Mbps, distance between sender and receiver from 100 km to 1200 km, as well as the use of both IPv4 and IPv6 protocols.

The characteristics of the collected dataset are shown in Table 1.

Table 1. Training Data.

Scenario	Number of objects	μ RTT, ms	σ^2 RTT, ms^2
Uploading files	922063	60.37	1344
Uploading files	1791633	9.89	144
Uploading files	1731034	40.75	58
Online game	30012	31.85	738
Web interaction	38250	10.10	140

- μ – average of RTT value
- σ^2 – variance of RTT value

It should be noted that the number of objects in the case of loading is significantly higher than in the other two scenarios. This is due to the fact that the complexity of obtaining objects in an online game scenario and in a web interaction scenario is much more complicated than in an upload scenario. However, section VI explores this problem for both balanced dataset and unbalanced dataset (number of objects in the loading scenario).

As noted above, a dataset with a true RTT value is needed to train the model. Experiments have shown that using the TCP packet option to measure accurate RTT does not provide significant advantages over *tcp.analysis.rtt* in the context of the task under consideration. Therefore, in this work, a *tcp.analysis.rtt* was used to obtain the correct RTT values.

6. Evaluation

This section presents the main results of the implemented algorithms in both offline and online scenarios. In addition, a comparison of machine learning algorithms with the classical Jacobson algorithm is presented.

6.1 Offline scenario balanced dataset

In this subsection, the considered models are trained on a balanced dataset containing objects from the upload scenario, the online game scenario, and the web interaction scenario. The models are tested on a dataset that also contains an equal proportion of objects in all three scenarios. The results obtained are presented in Table 2.

From the results obtained, it can be stated that in this scenario, a Random Forest shows the best result in terms of all metrics. It can also be noted that neural networks do not provide significant improvements compared to simpler algorithms in the context of the problem under consideration. The best value of k is understood as the smallest k, with an increase in which the error decreases slightly.

6.2 Offline scenario unbalanced dataset

In this subsection, the considered models are trained on an unbalanced dataset containing objects only from the loading scenario. The models are tested on a dataset that contains an equal proportion of objects in all three scenarios. The results obtained are presented in Table 3.

The results show that from the point of view of RTT prediction, the scenarios of different network situations do not differ.

Table 2. Offline scenario (balanced dataset).

Algorithm	best k	MSE	MAE	R^2	MAPE
ElasticNet	7	45.65	2.10	0.94	8.43
Random Forest	**4**	**40.36**	**1.80**	**0.94**	**6.75**
RNN	14	48.33	2.21	0.92	12.91
LSTM	14	47.98	2.13	0.93	11.29
GRU	14	48.50	2.20	0.92	12.58
The Jacobson algorithm	-	59.50	2.45	-	22.31

Table 3. Offline scenario (unbalanced dataset).

Algorithm	best k	MSE	MAE	R^2	MAPE
ElasticNet	8	43.37	2.00	0.94	7.97
Random Forest	**4**	**38.32**	**1.70**	**0.95**	**6.40**
RNN	11	48.24	2.19	0.92	12.84
LSTM	11	47.84	2.11	0.93	11.17
GRU	11	48.03	2.17	0.92	12.44
The Jacobson algorithm	-	59.30	2.67	-	23.24

From the results obtained, it can be argued that in this scenario, a Random Forest shows the best result in terms of all metrics. It can also be noted that neural networks do not provide significant improvements compared to simpler algorithms in the context of the problem under consideration. The best value of *k* is understood to be the smallest value of *k*, with an increase in which the error decreases slightly.

6.3 Online scenario

Due to the high dynamism of network environments, machine learning models trained on one traffic may not work well enough in traffic with other characteristics.

In this subsection, an online learning method based on an Adaptive Regression Forest with drift detection using the ADWIN method is proposed. The code implementing this training uses the River library [17].

The first dataset was collected in a low RTT network environment $objects: 60000, \mu = 8.67ms, \sigma = 1.24ms^2$, while the second dataset was collected in an environment with high RTT $objects: 20000, \mu = 73.93ms, \sigma = 2759.20ms^2$ and RTT have distribution with a heavy tail (Fig. 2).

A Random Forest was trained based on 50,000 objects in the first dataset, and then Random Forest tested on 10,000 objects of the first dataset and 20,000 objects of the second dataset. Thus, the case was considered when a model trained on a dataset with certain network characteristics was tested on a dataset with other network characteristics.

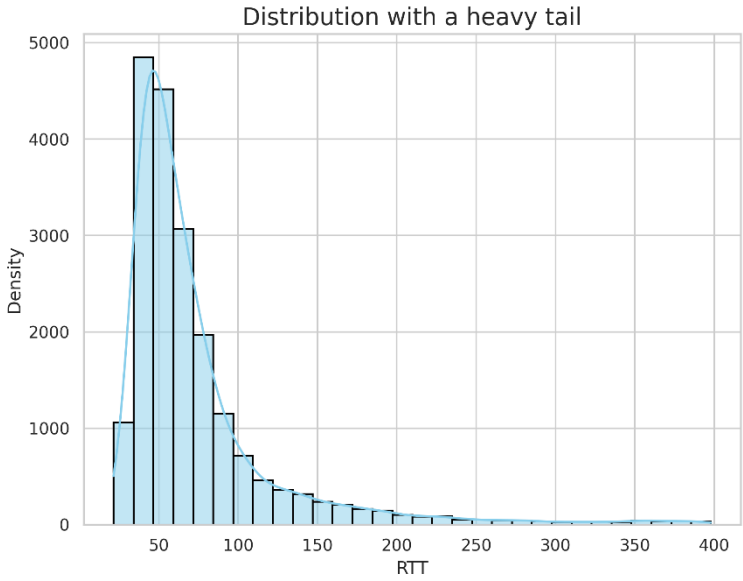

Fig. 2. Distribution with a heavy tail.

In the second experiment, the Adaptive Random Forest was trained on the 50000 objects of first dataset. On the second dataset, the algorithm was trained online using drift detection. This detection was used to more accurately train a Random Forest, in which the forest trees that solved the problem were most poorly replaced by new trees. The results obtained are shown in Fig. 3. The results obtained show the effectiveness of online learning in this task: the overall value of the MAPE metric does not deteriorating as critically as in the case of offline learning.

In the second pair of experiments, the dataset with a higher RTT $objects: 60000, \mu = 68.69ms, \sigma^2 = 2039.15ms^2$ value was the first, while the dataset with a lower RTT $objects: 20000, \mu = 8.70ms, \sigma = 1.22ms^2$ value was the second. The results obtained are shown in Fig. 4.

In this case, online learning also shows improvement. At the same time, offline learning shows very bad results.

7. Conclusions

In this paper, algorithms for RTT prediction using offline and online learning were presented. The results show that the algorithms based on learning works better in terms of MAPE, MSE, and MAE metrics in network environments with a wide range of network characteristics than the classic Jacobson algorithm.

However, as shown in this article, offline learning can be ineffective in dynamically changing network environments. To solve this problem, an online learning method with drift detection was proposed. The results show that in the case of online learning, the prediction efficiency does not deteriorate or deteriorates slightly when the environment changes.

The results obtained allow us to identify the following areas of future work:

- integration of online learning algorithms implemented in this paper into classical congestion control algorithms (TCP Reno, TCP CUBIC),
- study of the performance of classical congestion control algorithms using RTT prediction using online learning.

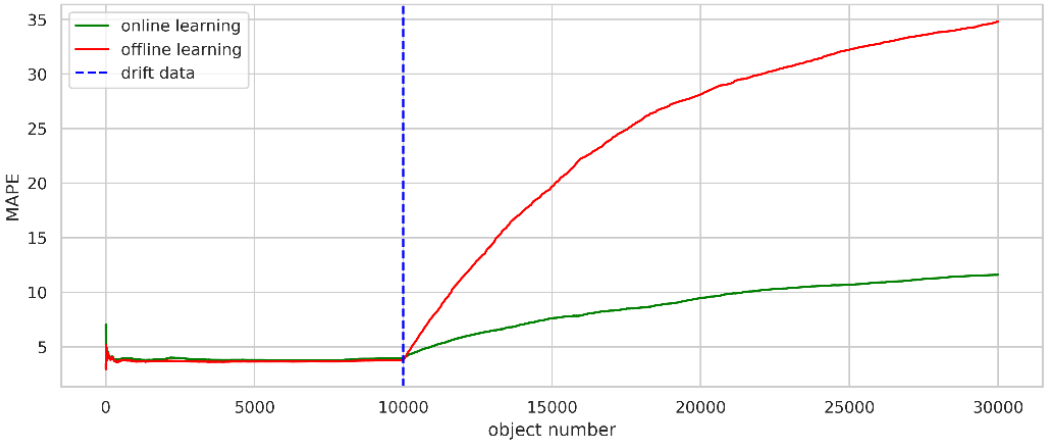

Fig. 3. Offline and Online learning (the first pair of experiments).

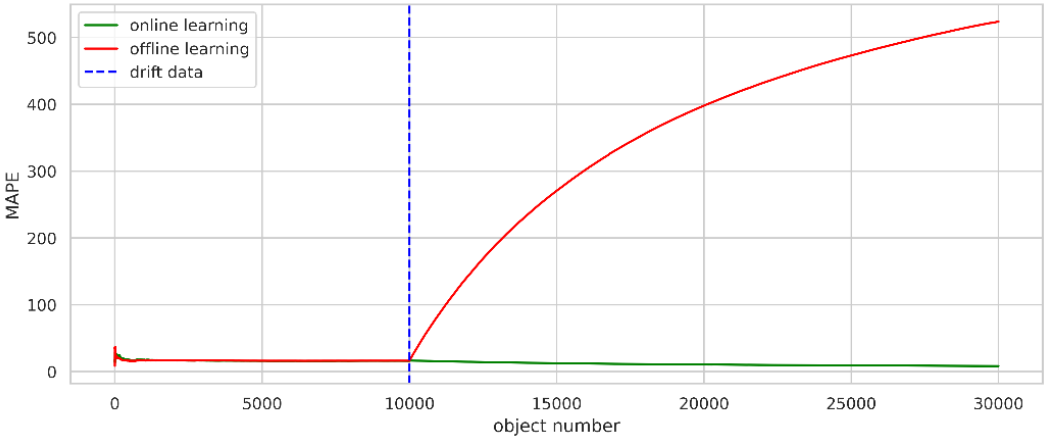

Fig. 4. Offline and Online learning (the second pair of experiments).

Список литературы / References

[1]. Mo, J., La, R. J., Anantharam, V., and Walrand, J. (1999, March). Analysis and comparison of TCP Reno and Vegas. In IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320) (Vol. 3, pp. 1556-1563). IEEE.

[2]. Hernández, José-Alberto, and Iain W. Phillips. "Weibull mixture model to characterise end-to-end Internet delay at coarse time-scales." IEE Proceedings-Communications 153.2 (2006): 295-304.

[3]. Cerroni, W., Foschini, L., Grabarnik, G. Y., Shwartz, L., and Tortonesi, M. (2017). Estimating delay times between cloud datacenters: A pragmatic modeling approach. IEEE Communications Letters, 22(3), 526-529.

[4]. Rizo-Dominguez, L., Munoz-Rodriguez, D., Vargas-Rosales, C., Torres-Roman, D., and Ramirez-Pacheco, J. (2014). RTT prediction in heavy tailed networks. IEEE Communications Letters, 18(4), 700-703.

[5]. Hu, Wen, Zhi Wang, and Lifeng Sun. "Guyot: a hybrid learning-and model-based RTT predictive approach." 2015 IEEE International Conference on Communications (ICC). IEEE, 2015.

[6]. Dasgupta, B., D. Valles, and S. McClellan. "Estimating TCP RTT with LSTM Neural Networks." Proceedings on the International Conference on Artificial Intelligence (ICAI). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp). 2019.

[7]. Hagos, D. H., Engelstad, P. E., Yazid, A., and Griwodz, C. (2019, October). A deep learning approach to dynamic passive RTT prediction model for TCP. In 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC) (pp. 1-10). IEEE.
[8]. Dong, Ai, Zhijiang Du, and Zhiyuan Yan. "Round trip time prediction using recurrent neural networks with minimal gated unit." IEEE Communications Letters 23.4 (2019): 584-587.
[9]. Berger, Vance W., and YanYan Zhou. "Kolmogorov–smirnov test: Overview." Wiley statsref: Statistics reference online (2014).
[10]. Berkson, Joseph. "Some difficulties of interpretation encountered in the application of the chi-square test." Journal of the American Statistical Association 33.203 (1938): 526-536.
[11]. Engmann, Sonja, and Denis Cousineau. "Comparing distributions: the two-sample anderson-darling test as an alternative to the kolmogorov-smirnoff test." Journal of applied quantitative methods 6.3 (2011).
[12]. Jaworski, Maciej, Leszek Rutkowski, and Plamen Angelov. "Concept drift detection using autoencoders in data streams processing." International Conference on Artificial Intelligence and Soft Computing. Cham: Springer International Publishing, 2020.
[13]. Nau, Robert. "The mathematical structure of ARIMA models." Duke University Online Article 1.1 (2014): 1-8.
[14]. Bifet, Albert, and Ricard Gavalda. "Learning from time-changing data with adaptive windowing." Proceedings of the 2007 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, 2007.
[15]. Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., ... and Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. Machine Learning, 106, 1469-1495.
[16]. Gomes, H. M., Barddal, J. P., Ferreira, L. E. B., and Bifet, A. (2018, April). Adaptive random forests for data stream regression. In ESANN.
[17]. Montiel, J., Halford, M., Mastelini, S. M., Bolmier, G., Sourty, R., Vaysse, R., ... and Bifet, A. (2021). River: machine learning for streaming data in python. Journal of Machine Learning Research, 22(110), 1-8.

Информация об авторах / Information about authors

Иван Александрович СТЕПАНОВ – аспирант, стажёр-исследователь ИСП РАН, ассистент кафедры информатики и вычислительной математики МФТИ. Сфера научных интересов: анализ сетевого трафика с помощью машинного обучения.

Ivan Alexandrovich STEPANOV – postgraduate student of the ISP RAS, intern researcher at ISP RAS, an assistant at the Department of Computer Science and Computational Mathematics at MIPT. Research interests: network traffic analysis using machine learning.

Роман Евгеньевич ПОНОМАРЕНКО – младший научный сотрудник ИСП РАН. Научные интересы: архитектура программного обеспечения, оптимизация программ, глубокий анализ сетевого трафика.

Roman Evgenevich PONOMARENKO – junior researcher at ISP RAS. Research interests: software architecture, program optimization, deep packet inspection.

Денис Ростиславович ГОЛОВАШ — лаборант ИСП РАН, студент ВМК МГУ. Сфера научных интересов: анализ сетевого трафика с помощью машинного обучения.

Denis Rostislavovich GOLOVASH is a laboratory assistant at the ISP RAS, a student at the Moscow State University. Research interests: network traffic analysis using machine learning.

Антон Юрьевич ПОКИДЬКО – стажер-исследователь отдела компиляторных технологий ИСП РАН. Научные интересы: дрейф в машинном обучении и нейронных сетях, трансферное обучение, федеративное обучение, онлайн обучение, анализ сетевого трафика.

Anton Yurevich POKIDKO – research intern at Compiler Technology department of ISP RAS. Research interests: drift in machine learning and neural networks, transfer learning, federated learning, online learning, network traffic analysis.

Александр Игоревич ГЕТЬМАН – кандидат физико-математических наук, старший научный сотрудник ИСП РАН, ассистент ВМК МГУ и МФТИ, доцент ВШЭ. Сфера научных интересов: анализ бинарного кода, восстановление форматов данных, анализ и классификация сетевого трафика.