Sparsity Regression Model-Based Relearning Architecture for Shortening Learning Time in Traffic Prediction

Takahiro HIRAYAMA (a), Takaya MIYAZAWA (b), Masahiro JIBIKI (c), and Ved P. KAFLE (d), Members

SUMMARY Network function virtualization (NFV) enables network operators to flexibly provide diverse virtualized functions for services such as Internet of things (IoT) and mobile applications. To meet multiple quality of service (QoS) requirements against time-varying network environments, infrastructure providers must dynamically adjust the amount of computational resources, such as CPU, assigned to virtual network functions (VNFs). To provide agile resource control and adaptiveness, predicting the virtual server load via machine learning technologies is an effective approach to the proactive control of network systems. In this paper, we propose an adjustment mechanism for regressors based on forgetting and dynamic ensemble executed in a shorter time than that of our previous work. The framework includes a reducing training data method based on sparse model regression. By making a short list of training data derived from the sparse regression model, the relearning time can be reduced to about 57% without degrading provisioning accuracy.

key words: network function virtualization (NFV), service function chaining (SFC), machine learning, dynamic resource arbitration

1. Introduction

Network function virtualization (NFV) techniques enable us to implement the softwarized functions of network equipment, known as virtual network functions (VNFs), on generic servers [1]. If the virtualized network services are created manually according to a legacy procedure, it may take approximately two weeks for the successful construction, from receiving a construction request to providing the service to the customer [2]. To continue satisfying the diverse quality-of-service (QoS) requirements despite time-varying network user behaviors and traffic volumes, the automation of the deployment and periodical adjustments of computational resources for each VNF, in particular, would help in realizing efficient and stable service provisioning to customers in time-varying network conditions (e.g., resource utilization and failure occurrences).

For the autonomic management of computational resources in an service function chaining (SFC) platform, i.e., an NFV platform specialized to provide service function chains, our previous work [3] proposed a framework for dynamic resource adjustments. Note that, service function chain contains a series of VNFs, such as load balancer, firewall, and so on [4]. Our framework consists of the following three steps as shown in Fig. 1: 1) resource arbitration between various network functions deployed in a server node, 2) network function migration by keeping the communication path unchanged, and 3) network function migration by changing the communication path. The evaluation results [3] demonstrated that the autonomic resource adjustment methods at the first and second steps can reduce the occurrence of CPU saturation by more than 90%. A prediction-based proactive control mechanism is expected to further enhance the adaptiveness to traffic variation.

However, with the rapid prevalence of Internet of things (IoT) applications and mobile network services, network traffic volumes and patterns are varying rapidly. This has led to the degradation of prediction accuracy. To ensure performance consistency despite the changes in traffic volumes and patterns, relearning the predictors has a key role. For efficient (re)learning process to keep accuracy in spite of traffic trend changes, we should pay attention to 1) eliminating insufficient variables from the training dataset to shorten learning time, and also, 2) executing relearning process in various time scales to tune predictors finely. In this paper, we propose a sparse regression model-based relearning architecture to tackle the above two challenges.

In general, machine learning models are based on adding and multiplying matrices (or tensors) having large amounts of parameters, such as backpropagation methods in deep neural networks (DNN). So, as the size of input data (the size of input layers in DNN) becomes larger, learning process takes tremendous time. Therefore, reducing data is one of the most common challenges in machine learning. We firstly present our least absolute shrinkage and se-
lection operator (LASSO) based sparse regression model to predict a virtual server load [5]. In our previous work [5], we dealt with the sparsity of real-life dataset of number of access to an operational Web server which is targeted as a virtual server load, and evaluated the effectiveness of our model from the perspective of prediction error.

Secondly, we propose a relearning architecture for traffic prediction framework based on the two-stage relearning processes to adapt to trend changes in short- and long-term scales. Our previous evaluation [6] revealed that the proposed framework can reduce the frequencies of over- and under-provisioning by more than 45% in comparison to RNNs and autoregressive moving average (ARMA). However, the relearning process takes a much time especially when it trains multiple regressors to keep up with trend changes in long-term.

This paper is extended version of our previous work presented in IEEE Netsoft’20 [6]. We lastly show that combination of the sparse model with the relearning architecture makes the relearning time about 57% shorter than that of previous work due to simplifying the explanatory variables. Note that the provisioning accuracy is not degraded due to simplification.

This paper is organized as follows. In Sect. 2, we introduce the related work. We explain our sparse regression model utilizing LASSO to predict the average number of access to the Web server in Sect. 3. We then evaluate the model quantitatively by means of MATLAB calculations in Sect. 4. In Sect. 5, we present the traffic prediction framework based on ensemble learning and its relearning architecture. The performance results are discussed in Sect. 6. The performance of a combination of the sparse regression model and relearning architecture is explained in Sect. 7. Finally, we conclude this work in Sect. 8.

2. Related Work

2.1 Resource Adjustments

Previous studies [7]–[9] investigated machine learning-based traffic prediction for realizing proactive control. The authors in [7] utilized artificial neural networks, while those in [8] utilized deep learning, which is a type of neural network. However, a neural network-based computational process generally requires a large amount of training data and its learning procedure is time consuming owing to its complex learning model. It might not be suitable for agile and dynamic resource adjustments when the network environment (such as traffic demands and failure occurrences) is rapidly varying (e.g., in the order of seconds or minutes). The authors in [9] used support vector regression for traffic prediction. Support vector regression is generally effective at improving the prediction accuracy.

2.2 Network Traffic Prediction Using Machine Learning

The authors in [7]–[14] investigate machine-learning based traffic prediction for realizing proactive control. The authors in [7], [10], [12] utilize artificial neural networks, while those in [8], [11] utilize deep learning. However, its model is so complicated, and thus, makes it difficult for humans to perform posterior analysis after learning processes.

In addition to deep learning techniques, many researchers have considered to introduce regression models for traffic prediction. Autoregressive models, such as ARMA, were used because of their simplicity [15]. To adapt to long term traffic transitions, the regression models based on support vector machine regressor, gradient boosting and Gaussian process were also considered. For example, Qian et al. proposed a prediction framework based on support vector machine (SVM) regression [16]. They first preprocessed traffic data by using empirical mode decomposition denoising and then used the preprocessed data for training. As a result, the prediction performance of SVM regressor improved because of denoising. Xia et al. proposed the traffic prediction framework based on filtering by random forest and regression in ensemble learning (gradient boosting) [17]. Bayati et al. proposed traffic prediction based on Gaussian process regression [18]. They focused on self-similarity of network traffic and claimed that Gaussian process regressor including self-similar covariance functions can provide better accuracy among various time-scale prediction targets, from every 5 minutes to 120 minutes. However, they mainly focused on discovering more suitable learning models for traffic prediction, and did not consider updating learning models in response to traffic trend changes in a long-term scale, what is called relearning mechanism, which is generally known to be effective at maintaining the prediction accuracy.

In this paper, we focus on the traffic prediction for the agile control of computational and communication resources allocated to the VNF platform. We aim at achieving shorter learning time and maintaining the prediction accuracy high despite the fluctuation of traffic in long- or short-term basis. We propose the ensemble learning based prediction framework and its relearning mechanism. Our ensemble learning framework is inspired by the stacking architecture, which is based on the combination of two or more regression models. We mainly evaluate the advantageous effects in terms of the reduction of learning time by extracting important variables from the datasets and the improvement of prediction accuracy by the relearning approach. Therefore, we only use RNN, elastic net, random forest, and Gaussian process as the components of the ensemble learning framework although our framework can easily support other learning models, including the methods discussed in [15]–[18].

When the data has sparsity, sparse modeling such as LASSO [19]–[22] is effective to simplify the model. Here, sparsity means that, even though the data itself is massive, only a small part of data is meaningful for an intended analysis such as prediction, detection, and so on. LASSO is a kind of regression analysis originally proposed in Year 1996 [19], and its model and/or application have been widely stud-
ied by many researchers on machine learning. LASSO can extract really-important data effectively from a large-size and complicated original data. LASSO also has characteristics that it achieves agile prediction processing and improvement of memory consumption by making a short list of explanatory variables. Therefore, our approach is to use LASSO [5] to reduce learning data and shorten learning time of machine learning.

3. Sparse Regression Model

As discussed above, to shorten the computation time in re-learning process in our prediction framework, sparse regression analysis plays a critical role because it determines the important explanatory variables and the time period targeted for explanatory variables. So, we firstly present our previous work of sparse regression analysis.

3.1 LASSO-Based Regression Analysis Model

In [5], we formulate a LASSO based regression analysis model. Our model regards the values of server loads monitored every one minute in the time period from $t_1$ to $t_2$ as explanatory variables, and predicts the average value of server loads in the period from $t_1$ to $t_2$.

We define $E$ as the least mean square error (LMSE) in prediction of the average value of server loads, which can be formulated as

$$E = \sum_{\mu=1}^{D} (Z_\mu - z_\mu)^2 + \lambda \sum_{i=1}^{M} |w(i)|$$

(1)

where $Z_\mu$ is the actual observed value of objective variable, $\lambda$ is a regularization parameter, $z_\mu$ is the predicted value of objective variable which is expressed as

$$z_\mu = \sum_{i=1}^{M} w(i)x_\mu(i) + b$$

(2)

where $x_\mu(i)$ and $w(i)$ are the actual observed value and weight of $i$ th explanatory variable in $\mu$ th data ($\mu \in \{1, 2, \ldots, D\}$), respectively. $b$ is an intercept coefficient. $M$ is the number of explanatory variables, which is expressed as $M = t_2 - t_1 + 1$. The LASSO sets up the values of $w(i)$, $b$ and $\lambda$ so that the value of $E$ can be minimum. Generally, the second term in Eq. (1) is called a regularization term. If $\lambda = 0$, this term does not affect the prediction error and the model is the same as the least square method (LSM). In other words, the LASSO strives to avoid overfitting of learning by adding the regularization term to the equation of prediction error [19]–[22]. LASSO simplifies the mathematical model by making a short list of explanatory variables. Note that we define important explanatory variables as a small number of variables selected for prediction of server loads by LASSO.

3.2 Algorithm for Determining the Time Period Targeted for Explanatory Variables

To reduce the errors as much as possible, trying all possibilities of time periods requires too much learning time. Thus, previously, we proposed an effective algorithm to determine the time period targeted for explanatory variables, of which the detail is described in [5]. Our algorithm determines the time period targeted for explanatory variables by iterative learning processes. In the first cycle of iterative learning, the LASSO-based regression analysis is performed by use of the initial setting values of $t_1$ and $t_2$. Then, our algorithm examines if removing one slot is effective at reducing the prediction error. Here is an issue of how to decide a removed slot in each cycle of iterative learning. We adopt two approaches: (i) preferentially removing the earliest slot in the current time period and (ii) preferentially removing the latest slot in the current time period. Our algorithm prioritizes method (i) over (ii). In every cycle of iterative learning process, if method (i) can reduce or does not affect the prediction error by LASSO-based regression analysis, the algorithm removes the earliest slot out of the current set of slots.

Figure 2 (a) illustrates state transition steps to determine the optimal time period targeted for explanatory variables, and Fig. 2 (b) shows an example of removing one slot in each cycle of iterative learning. For example, the initial state denotes the case when $(t_1, t_2) = (7:00, 11:29)$, and the lower state is a narrower time period than the upper state. If the algorithm removes the earliest time slot from the current time period, the state is changed from the current state to the
left lower one. On the other hand, if the algorithm removes the latest time slot, the state is changed to the right lower one. Figure 2(b) illustrates three patterns of state transition. The first pattern (i.e. left upper) indicates the case in which the transition from State \( x \) (ex. \( (t_1, t_2) = (9:00, 11:29) \)) to the left lower State \( y \), (ex. \( (9:29, 11:29) \)) does not degrade the prediction error. That is, the LASSO regressor trained with the data gathered within the time period at State \( y \) can predict a server load more correctly than the one trained with the data gathered within the time period at State \( x \). In this case, State \( y \) becomes the current state. The second pattern (i.e. right upper) indicates the case in which the transition from State \( x \) to the left lower state (State \( y \)) degrades the prediction error, and thus the state is going back to the previous state (State \( x \)). Besides, since the transition from State \( x \) to the right lower state (State \( z \)) does not degrade the prediction error, State \( z \) becomes the current state. The third pattern (i.e. lower) indicates the case in which both transitions from State \( x \) to the left/right lower states degrade the prediction error. In this case, the algorithm does not execute state transition any longer, and State \( x \) becomes the eventual state. In this way, the targeted time period in State \( x \) is determined as the optimal one, and the algorithm finishes the iterative learning processes of LASSO-based regression analysis.

4. Evaluations of Sparse-Regression Analysis

To evaluate our algorithm mentioned in Sect. 3.2, in [5], we used real-life dataset (i.e. the number of access) obtained from the Web server of our research institute the National Institute of Information and Communications Technology (NICT). We presuppose that the longest time period targeted for explanatory variables is the period from \( t_1 = 7:00 \) to \( t_2 = 11:29 \), in the initial state. This means that the number of explanatory variables, \( M = 270 \) in the initial state. At the each of state transitions, the size of the monitoring window, \( M \), is decreased by 30. The objective variable is the average number of access to the Web server in the time period from 12:00 to 12:59. Our algorithm determines the time period targeted for explanatory variables consisting of one or multiple continuous slots by iterative learning processes explained in [5].

As we explained in Sect. 3.1, the LASSO sets up the values of \( w(t) \), \( b \) and \( \lambda \) so that the value of \( E \) (i.e. prediction error) can be minimum in Eq. (1). In the MATLAB calculations, the value of \( \lambda \) is selected within the range from 0 to 200. Then, the LASSO examines 1000 values of \( \lambda \) with the step size of 0.2, and finds out the optimal value of \( \lambda \) that gives minimum value of \( E \).

4.1 Effect of Our Algorithm

Figure 3 illustrates the advantageous effect of our algorithm for determining the time period targeted for explanatory variables [5]. As we explained in Sect. 3.2, the objective is to find out the optimal time period for explanatory variables, which is the optimal combination of \( (t_1, t_2) \) in view of the LMSE of prediction. If we try all 9 combinations of \( (t_1, t_2) \) in the range from 7:00 to 11:29 by 30 minute intervals, i.e., \( (t_1, t_2) = (7:00, 7:29), (7:30, 7:59), \ldots, (11:00, 11:29) \), the total number of trials is \( 511 \), which is too large, and thus, it takes a long time for the learning processes. Since one trial takes around 10 sec at minimum (and around 20 sec at maximum), the total time required for learning processes is much more than 5210 sec, which is around 1 and a half hours. In order to achieve agile and dynamic server resource adjustments utilizing the sparse regression analysis results, we need to drastically reduce the time required for learning processes.

As we stated in the above, \( (t_1, t_2) \) are initially set to \( (7:00, 11:29) \). Our algorithm preferentially removes the earliest slot in the current time period in each cycle of iterative learning in accordance with method (i). It continues until the prediction error degrades by removing a slot. In this case, the performance degrades when the values of \( (t_1, t_2) \) changes from \( (9:30, 11:29) \) to \( (10:00, 11:29) \). Hence, the time period is reverted to the previous state (i.e. period from 9:30 to 11:29), and our algorithm tries to remove the latest slot ranging from 11:00 to 11:29 in accordance with method (ii). The algorithm finishes the iterative LASSO-based learning processes and finally determines the time period targeted for explanatory variables as the range from 10:00 to 10:59. Note that “Not degrade” or “Degrade” described in Fig. 3 in each state transition corresponds to the variations of average value of LMSE. As a result, our algorithm only requires 11 iterative learning processes (trials). In other words, we can reduce the number of trials (i.e. the time required to find out the optimal time period targeted for explanatory variables) by around 98 %.

4.2 Degree of Making a Short List of Explanatory Variables

Next, we evaluate the degree of making a short list of explanatory variables by our LASSO-based sparse regression model [5]. Figure 4 shows the number of explanatory variables selected by LASSO and regarded as important ones by us (i.e. important explanatory variables; Left vertical scale) and the total number of explanatory variables (Right vertical scale) versus the starting time of period targeted for explanatory variables, where the former results are shown by the bar graphs and the latter ones are shown by the line graphs. In this paper, we extract the important explanatory variables via the \( k \)-fold cross validation method. In the \( k \)-fold cross
validation, we randomly separate the training data into unordered $k$ groups. We then train a LASSO-based regressor with the $k - 1$ groups with the remaining one group. We repeat the training and testing cycles $k$ times by taking one group at each time as the test data and the remaining $k - 1$ groups as training data. By doing so, $k$ trials (from $i = 1$ to $k$) are executed to tune the hyper parameters. Hyper parameters of the regressor are defined as the average values the results of the above $k$ trials. As a result of trials and errors, we set the value of $k = 7$ in this paper. That is, we define the important explanatory variable as the one which has four or more non-zero weights in the 7-fold cross validation of LASSO. There are several cases that the number of important explanatory variables increases even though the value of $M$ decreases. Though it is difficult to perceive the detailed behavior of LASSO-based regression analysis, we observed that several explanatory variables newly become important ones even though they had not been extracted in previous states. In any cases, Fig. 4 indicates that the number of important explanatory variables falls within the range from 9 to 21. In the optimal targeted time period ranging from 10:00 to 10:59, the value of $M$ is 60 and the MATLAB calculation result shows that the important explanatory variables include 14 clock times, which are 10:01, 10:04, 10:07, 10:08, 10:15, 10:22, 10:27, 10:32, 10:33, 10:40, 10:41, 10:50, 10:51, 10:53. That is, the number of explanatory variables reduced to 23%.

5. **Ensemble Learning and Two-Stage Relearning Architecture**

5.1 Overview of Our Prediction Framework

We assume that physical server machines where multiple VMs can be created. Additionally, there is a controller, which consistently monitors the VM workload and utilization of server resources. It migrates some VMs from a resource-saturated server to those that have enough available resources. Figure 5 shows the architecture of our machine learning framework. Our prediction framework is based on one of ensemble learning models, stacking architecture [23]. The stacking architecture is consisted of multiple weak regressors and one strong regressor. At the first step, weak regressors predict traffic volumes by using explanatory variables. Next, the strong regressor predicts traffic by using output values from weak regressors as its explanatory variables. By using multiple learning models combined as parts of the ensemble learning architecture, we attempted to achieve better prediction performance than that of each separate model. For ensemble learning, we created multiple weak regressors and one strong regressor by training with the datasets $D_w$ and $D_s$, respectively. To prepare for the changes in traffic trends in future, we proposed a two-stage relearning mechanism based on dynamic ensemble and forgetting. In the dynamic ensemble process, the strong regressor was retrained with the data collected over past few hours or days. This process was designed for the adjustment of strong regressor to adapt to the changes in trend in short time scales. Moreover, in the forgetting process, the weak regressors were retrained with the data gathered during the past few weeks or months. This process was aimed to stay up-to-date with the changes in trends in long time scales. By virtue of the above relearning mechanism, our prediction framework can keep up with both the short- and long-term changes in trends.

5.2 Training Dataset

We used time-series data that represented the number of accesses to a web server per minute. Because the number of accesses to a web server is related to CPU utilization, this type of data is useful for the prediction of future workload and dynamic adjustments to the server resources. We targeted and analyzed real-life data collected from the web server of our institute. Figure 6 shows a sample data collected over one day (from 10:50 to 13:00 on March 1, 2018). The vertical axis represents the number of accesses to the web server every minute and the horizontal axis is the time of the day. Using a regression analysis model, we predicted the values of the objective variable (i.e., an unknown future value) based on the values of explanatory variables (i.e., known given data) in the monitoring window between time $t_1$ and $t_2$. Here, the objective variable is the predicted average number of accesses to the server in the period from $t_3$ to $t_4$, while the time period targeted for explanatory variables (i.e., past data) is moved according to the objective. We set the values of $t_1$ and $t_2$ as $(M + I)_{\min}$ earlier than $t_3$ and $t_4$, respectively, that is, $t_1 = t_3 - (M + I)$ and $t_2 = t_4 - I$. The term $(M + I)_{\min}$ is derived from the sum of the monitoring range ($M_{\min}$) and time margin for NFV management, such as arbitration or migration ($I_{\min}$). Note that we set $M = 60$, $I = 10$, $t_1 = 10:50$, $t_2 = 11:49$, $t_3 = 12:00$, and $t_4 = 12 : 19$ in this case. The time from 11:50 to 11:59 denotes an interval of $I (= 10)$ min. We used sample data from 30 and 60 days as the training and testing data, respectively, from Jan-

---

**Fig. 4** The number of important explanatory variables (Left vertical scale) and the total number of explanatory variables (Right vertical scale) versus the starting time of period targeted for explanatory variables.
January 29, 2018 to April 29, 2018. In this paper, the prediction intervals were set at every 20 min, from 12:00 to 25:59 (i.e., \((t_1, t_2) = (12:00, 12:19), (12:20, 12:39), \ldots, (25:40, 25:59)\)). Accordingly, there were 42 prediction intervals each day. In our dataset, the average numbers of web requests in the morning (from 02:00 to 12:00) and afternoon (from 12:00 to 25:59) are about 417.5 and 1134.1 requests per minute, respectively. That is, the mean value of the number of requests in the afternoon is 2.7 times larger than that in the morning. In addition, the maximum number of arrivals per minute in the afternoon was 1559, which was two times larger than that the morning value of 719. Therefore, in this work, we mainly focus on the traffic prediction during the afternoon because the traffic fluctuation is more drastic than that in the morning.

5.3 Weak Regressors for Ensemble Learning

In the first step, we created some weak regressors trained by the subsets of the training dataset, \(D_w\), collected within \(\{\widetilde{D}_w\} (= 30)\) days, to obtain a wider variance in the models, considering the bagging approach. We separated the training dataset per day, \(D_w\), into \(N_w\) and \(N_r\) subsets according to the random-based and \(k\)-means-based rules, respectively. The real traffic data represent a type of time-series data. By using the \(k\)-means method, the training dataset is expected to be separated into groups that have similar traffic trends, such as weekday and holiday traffic. We packed the daily traffic data into \(N_r\) subsets with the time series \(k\)-means method [24]. Note that we selected this method to compare the results of the \(k\)-means and \(k\)-shape methods. We also created weak regressors trained using all training datasets. As a result of training, \(N_w = (m(N_r + N_k + 1))\) weak regressors were created, where \(m\) denotes the number of learning models. The \(i\)-th regressor \((i \in 1, 2, \ldots, N_w)\) predicts the average traffic volume in the range \([t_1, t_4]\) according to the explanatory variables at every minute from \(t_1\) to \(t_2\), i.e.,

\[
p_{w,i}(t_3, t_4) = f_{w,i}(v(t_1), v(t_1 + 1), \ldots, v(t_2 - 1), v(t_2))
\]  

where \(p_{w,i}(t_3, t_4)\) denotes the predicted value, \(f_{w,i}(\ldots)\) denotes the prediction function obtained by training with a subset, and \(v(t)\) denotes the traffic volume at time \(t\).

We used the following three \((m = 3)\) learning models for weak regressors: RNN, elastic net, and random forest. We selected these models because they represent three different types of models, namely a typical neural network, statistical learning model, and ensemble learning models, respectively. RNN regressors have the typical RNN structure composed of a long short-term memory (LSTM) unit [25]. The RNN regressors read and memorize the time-series data whose length is \(M\) sequentially. Accordingly, they predict the average traffic volume in the range \([t_3, t_4]\).

Elastic net regularization [26] is the linear combination of LASSO regression and ridge regression. It includes L1 and L2 penalties and it can be considered as an extension of the LASSO model because it overcomes the constraint of the number of explanatory variables of LASSO. In elastic net, the balance between L1 and L2 penalties can be coordinated by a tunable parameter \(R_{L1} \ (0 < R_{L1} < 1)\). When \(R_{L1} = 1\), the elastic net regressor is the same as that of LASSO regression, and when \(R_{L1} = 0\), it is the same as that of ridge regression.

Random forests [27] are a type of ensemble learning method with multiple decision trees. Decision trees predict the traffic volume by checking whether each explanatory variable exceeds the threshold. Each decision tree is deeply trained by bagged training data; consequently, it tends to be overfitted. Random forest regressors output the average value of the predictions of all trees as the final prediction.

Training these weak regressors is time-consuming be-
cause the dataset is large (i.e., $|D_w| = 30$ days). It may require specific computational resources, such as CUDA-compatible GPUs for RNN. Therefore, in the proposed architecture, we assumed that such heavy training for weak regressors is executed on high power controller machines.

5.4 Strong Regressors for Ensemble Learning

Weak regressors trained on the controllers are distributed to each VNF server. In the second step, we created strong regressors trained by the dataset $D_s$, acquired within the past $|D_s|$ (= 7) days. The strong regressors predict the average traffic in the range from $t_3$ to $t_4$ according to $N_w$ explanatory variables acquired by the weak regressors, i.e.,

$$p_s(t_3, t_4) = f_s(p_{w,1}(t_3, t_4), p_{w,2}(t_3, t_4), \ldots, p_{w,N_w}(t_3, t_4))$$  \hspace{1cm} (4)

where $p_s(t_3, t_4)$ denotes the predicted values and $f_s(\ldots)$ denotes the prediction function obtained by the following learning process for the strong regressors.

1. Predict the objective value by using $N_w$ weak regressors. Consequently, $N_s$ predicted values are obtained.
2. Train the strong regressors with average traffic volume in the prediction target by considering $N_w$ predicted values as explanatory variables.

In the above training process, the training regressors do not consume a significant amount of computation time because the training dataset is much smaller than the datasets in the case of training weak regressors, as described in the previous subsection.

We used Gaussian process regression as the learning model for strong regressors. It showed better performance than the case elastic net is used as the strong regressor. In the Gaussian process, the hyper-parameters are determined via maximum likelihood estimation. The Gaussian process can be used like deep neural networks [28]. However, these parameters need not be tuned because the Gaussian process is a non-parametric regression model.

5.5 Relearning Methods for Strong and Weak Regressors

Because the traffic trends keep changing over time, it is difficult to predict the traffic volume based on the training data acquired in the past, such as more than one month ago. A simple countermeasure to fill the gap between the current traffic patterns and training datasets is training each regressor with the newly obtained traffic data. In our architecture, weak regressors were re-trained periodically (every $T_R$ (= 15 days)) with new traffic data. This process is based on the so-called “forgetting” approach.

As previously described, training weak regressors requires a significant amount of computation time and specific hardware resources. Note that the forgetting process (including relearning new weak regressors) required several tens of minutes, up to several hours, with Intel Core i7-8700 CPU and GeForce RTX 2080 GPU. The frequent execution of the forgetting process, i.e., hourly or daily, is not desirable. Therefore, we introduced a light-weight re-training method that can be executed on each VNF server. Servers always monitor the prediction performance of strong regressors by comparing their predicted values and real traffic volume. When the frequency of prediction errors of a strong regressor exceeds a certain threshold (say $\text{Threshold}_{D_s} = 3$), the server re-trains the regressor with the most recent $|D_s|$-day traffic data. We assumed that a prediction error occurs when the provided CPU resource, according to the prediction, becomes 1.5 times larger than the real traffic or less than the real one. Moreover, we set the amount of CPU resources to be 1.2 times larger than the predicted value. This process is based on a “dynamic ensemble” approach. As previously mentioned, training strong regressors does not require a significant amount of computation time. Thus, this re-training process can be executed on VNF servers in a short period (less than 100 s in our case).

6. Evaluation of Relearning Architecture

6.1 Training Method

Dynamic ensemble improves the prediction accuracy by selecting adequate weak regressors from several weak regressors. Therefore, the number of weak regressors should be sufficiently large. To increase the number of weak regressors, we separated the training dataset (i.e., 30-day traffic data) into daytime and nighttime data. Consequently, we used three datasets, i.e., datasets for the daytime (from 12:00 to 19:59), nighttime (from 18:00 to 25:59), and all of them (from 12:00 to 25:59). Each dataset was separated into two groups by random clustering and $k$-means clustering, that is, $N_r = N_k = 2$. The number of weak regressors, $N_w$, was $m(N_r + N_k + 1) \times 3 = 45$.

We implemented RNN regressors by using PyTorch, while the other regressors were implemented using scikit-learn. The regressor hyper-parameters were tuned via grid search and cross-validation techniques. For example, the values of $R_{t_1}$ in the elastic net were selected from $[0.1, 0.5, 0.7, 0.9, 0.95, 0.99, 1]$. The kernel set used in the Gaussian process was selected from 11 patterns of combinations of linear, RBF, and Matérn kernels.

We set the values of $M$, $T_R$, $|D_w|$, and $|D_s|$ as 60 [min], 15 [days], 30 [days], and 7 [days], respectively. Evidently, these values strongly relate with the training time and performance. We set these values based on trial and error. For training and regression, traffic volume is preprocessed as following function: $z = (x - u)/s$ where $x$ is traffic volume, $u$ and $s$ are the mean value and the standard deviation of the training datasets.

6.2 Prediction Performance

We evaluated the prediction accuracy within the period from March 1, 2018 to April 29, 2018. We also evaluated the ARMA model as the baseline of prediction. The ARMA
model is trained with the same dataset as the initial state of our ensemble-learning-based prediction framework. That is, it is trained with the 30-day traffic dataset from January 30, 2018 to February 28, 2018. It predicts the traffic volume according to the recent 6 h traffic data. Figure 7 shows the typical difference between ensemble learning with and without the relearning mechanism. This figure shows the traffic data from March 15 to March 19. The forgetting process is already executed once because the first forgetting process occurs on March 15, i.e., more than $T_R$ (= 15) days after the start date. The dynamic ensemble process is executed three times. The predictions made by the ensemble learning process without the relearning mechanism (denoted by “Ensemble” in Fig. 7) is far from the real traffic (dashed line in the figure). The predicted traffic volume is smaller than the real one on March 19. However, the ensemble learning process with the relearning mechanism (denoted by “DE+FG” in Fig. 7) predicts the traffic volume that is consistent with the real traffic. We also evaluated the R$^2$ score (not shown in the figure) to estimate the prediction accuracy of the ensemble learning process with and without forgetting and/or dynamic ensemble and each of the weak regressors. Consequently, the predicted values obtained by the ensemble learning architecture including the relearning framework based on DE (dynamic ensemble only) and DE+FG yielded the best R$^2$ scores (about 0.78 and 0.77, respectively), while the R$^2$ score of the ARMA model was 0.73.

We evaluated the prediction performance for provisioning resources to each VM according to the prediction. The results shown that the total provisioning error is approximately 45% smaller than that of the ARMA model and RNN [6].

7. Evaluation of the Combination of Sparse Regression and Relearning Architecture

In previous two sections, we show our relearning architecture keeps prediction and provision accuracy. However, it takes a much time to learning. As a result of evaluations, we reveal one forgetting process for 45 weak regressors with 30-day traffic data of only one www service takes more than 20 minutes. To earn the scalability to the number of services and regressors, (re)learning process should be shortened. For that purpose, we attempt to reduce the number of explanatory variables by the sparse regression analysis as described in Sect. 3.

To predict traffic arrivals from $t_3$ to $t_4$, our ensemble learning framework uses $M$ explanatory variables ($t_1, t_1 + 1, t_1 + 2, \ldots, t_1 + M - 2, t_1 + M - 1 (= t_2)$). By extracting the important variables from the $M$ explanatory variables on the basis of the sparse regression analysis described in Sect. 3.2, the learning time can be reduced. Hereinafter, we set the size of interval $I$ as a constant value to locate the monitoring window nearby the prediction objective (i.e., $I = 10$) to enhance the prediction accuracy against the multiple prediction objective ranged from 12:00 to 25:59. Meanwhile, the appropriate size of monitoring window $M$ is determined by the algorithm described in Sect. 3.2. We assume that the monitoring window size can be stretched until the number of important explanatory variables extracted by the sparse modeling exceeds 60. We have fixed the monitoring window size to 60 in the previous evaluation, because our purpose in this section is to reduce training time and maintain the prediction accuracy. To avoid falling into a local minimum value, we started the process to determine a suitable value of $M$ with the cases when initial $M$ values are 60, 120, 180 and 240. As a result of that, the best values of the monitoring window sizes are $M = 50$ in the second relearning process and $M = 60$ in the other relearning processes. As a result, the number of explanatory variables for the first learning process at 2018/3/1 with the data gathered from Jan. 30, 2018 to Feb. 28, 2018 was reduced to 30 as shown in Fig. 8. The number of important variables is larger than 6 in the case of Sect. 4.2 due to the multiple time periods of prediction objectives. In addition, the numbers of explanatory variables for the second relearning process at Mar. 16, 2018, the third relearning process at Mar. 31, 2018, and the fourth relearning process at Apr. 15, 2018 are reduced to 28, 48, and 33, respectively. In summary, the average number of explanatory variables for each relearning process is approximately 45% smaller than that of the ARMA model and RNN [6].
of important explanatory variables is 34.75, in other words, the number of explanatory variables is decreased about 42%. As a result, (re)learning time in the case of extracted important variables becomes about 57% shorter than the case of $M$ variables. As described in Sect. 3.1, important variables are selected from LASSO with L1 regularization. In the ensemble learning, we also include L1 regularization in the weak regressors based on the elastic net. Therefore, the extraction of important variables is useful for the regressors based on the elastic net due to the consistency among LASSO and the elastic net as those include L1 regularization. There is inconsistency among the sparse modeling and the other regression models, RNN, and elastic net. However, our evaluation results have revealed that extracting important variables is also helpful to improve the performance accuracy of other regressors based on RNN and random forest, in spite of inconsistency of models.

We evaluate the prediction and provision accuracy. Figure 9 shows the traffic prediction from Mar. 15 to Mar. 19, 2018. In spite of the reduced variables, the predicted traffic by extracted variables ("Sparse +Relearning" in the figure) shows the similar pattern to the result of $M$ variables ("Relearning (DE+FG)"). We also evaluate the $R^2$ score via 60-day traffic prediction from Mar. 1 to Apr. 29, 2018. The value = 0.74 is slightly reduced from the result of all $M$ variables, 0.78 (DE) and 0.77 (DE+FG).

As shown in Fig. 9, our prediction framework predicts traffic arrivals. The figure shows that the predicted traffic exhibited a little delay behind the real traffic when the volume of traffic fluctuated. We have to evaluate our framework to know if it is better than the case when we use the regressor predicts traffic by just copying the amount of traffic arrivals at the previous time slot. For this purpose, we label the reference method as “just repeating method” and define it as a prediction method based on just copying the traffic data of the previous time slot. In this method, the regressor defines the predicted traffic volume as the mean value of traffic arrived from $t_{\text{head}}$ to $t_{\text{tail}}$. As a result of trials and errors, this method exhibited the best $R^2$ score (about 0.80) when we set $t_{\text{head}} = t_3 - 65$ and $t_{\text{tail}} = t_3 - 5$. Figure 9 also shows the predicted traffic by the just repeating method (labelled as JustRepeat in the figure). The just repeating method shows the best $R^2$ score among the other regressors used in this work. However, to consider not only reproducibility of the models but also their usefulness for resource provisioning, we evaluated the prediction performance for provisioning resources to each VM according to the prediction. So, we evaluated the prediction performance for provisioning resources to each VM according to the prediction. We assumed that each VNF server set the amount of CPU resources (vCPU) for example, vCPU quota in KVM). In this evaluation, we assumed that each VNF server set the amount of CPU resources to be 1.2 times larger than the predicted traffic. We assumed that the prediction error occurred in the case when the provided CPU resources, according to the prediction, exceeded the range of being 1 to 1.5 times larger than the real traffic. When the CPU resources are less than those required to process the real traffic, the case is considered to be under provisioned. Similarly, when the CPU resources are 1.5 times larger than the required quantity, the case is considered to be over provisioned.

Figure 10 shows the mean values of occurrence of prediction errors for each day. We evaluated the mean values of occurrence of prediction errors for each day, defined as the mean values of (frequency of errors in a day)/(number of prediction targets per day (= 42)). The 95% confidence intervals are also plotted. As shown in Fig. 10(b), making a short list of explanatory variables more often causes under provisioning. However, the quantities of over- or under-provisioned resources at the worst cases still stay in the similar range to the case when all $M$ variables are used. We investigated the cause of the gap in the cases of under-provisioning. We confirmed that the performance of the combination of sparse modeling and relearning extremely decreases in the range from 41st day (Apr. 9) to 45th (Apr. 14) day. In fact, the $R^2$ score reaches 0.77 (it equals to the result of all M variables) and the ratio of under-provisioning error reduces when the results the above duration are excepted, as shown in Fig. 11 as “SP+RL (1-40, 46-60).” This phenomenon may be induced by the drastic trend changes from March to April (this is the end of fiscal year in Japan) that cannot be compensated by our sparse modeling. The performance gap between relearning and sparse+relearning...
is disappeared at the next forgetting process executed at Apr. 15th data. To overcome it, one of simple measures is the manual execution of forgetting process when a drastic performance degradation occurs.

In Fig. 10, the results of the just repeating method are also shown. This method slightly suppresses the frequency of under-provisioning in comparison with our prediction framework. However, the frequency of over-provisioning in the method is two times larger than those in our framework. Clearly, our prediction framework outperforms the just repeating method in terms of the prediction error rate.

Figure 11 shows the letter-value plot of the difference between the provisioned resources and real traffic. For instance, when the quantity of provided resources fits to accommodate the amount of arrived traffic without excess or deficiency, the difference is 0. The bar width represents the density of the values existing around each range of difference. The outliers existing at 5% areas of the top or bottom are plotted as points. Figure 11 reveals that the results of provisioning with extracted variables show the almost the same performance as provisioning with all M variables. The performance of the just repeating method is also shown in the figure. The size of extremely over-provisioned resources (in the top 5%) is larger than that of our framework. In summary, extracting the important variables based on the sparse regression analysis can reduce the (re)learning time of our ensemble-learning-based prediction framework without drastic performance degradation.

8. Conclusions

In a network function virtualization environment, prediction results of function load in a specific time period are necessary as triggers for agile and dynamic server resource adjustments. In this paper, we firstly proposed the algorithm for extracting the important explanatory variables within the monitoring time period, which can reduce the number of explanatory variables to about 23%.

We presented a two-stage ensemble learning architecture, where the stages were executed on the controller and resource managers of the VNF servers, respectively. We also presented a relearning mechanism with the dynamic ensemble and forgetting approaches to keep up with the changes in traffic trends. The performance evaluation results showed that our prediction framework obtained the best prediction accuracy among the weak regressors and ARMA. The above sparse regression analysis contributes to reduce the learning (relearning) time of the relearning architecture without performance degradation by extracting important variables within the monitoring window. In future work, we will improve the prediction accuracy to mitigate resource shortages induced by sudden increases in the traffic.

Acknowledgments

This work was conducted as part of the project entitled “Research and development for innovative AI network integrated infrastructure technologies (JPMI00316)” supported by the Ministry of Internal Affairs and Communications, Japan.

References

[1] B. Han, V. Gopalkrishman, L. Ji, and S. Lee, “Network Function Virtualization: Challenges and Opportunities for Innovations,” IEEE Commun. Mag., vol.53, no.2, pp.90–97, Feb. 2015.
[2] K. Katsuura et al., “IaaS Automated Operations Management Solutions That Improve Virtual Environment Efficiency,” NEC Technical Journal, vol.8, pp.29–32, April 2014.
[3] T. Miyazawa, M. Jibiki, V.P. Kafle, and H. Harai, “Autonomic Resource Arbitration and Service-Continuable Network Function Migration along Service Function Chains,” IEEE/IFIP Network Operations and Management Symposium (NOMS), April 2018.
[4] A.M. Medhat, T. Taleb, A. Elmangoush, G.A. Carella, S. Covaci, and T. Magedanz, “Service Function Chaining in Next Generation Networks: State of the Art and Research Challenges,” IEEE Commun. Mag., vol.55, no.2, pp.216–223, Feb. 2017.
[5] T. Miyazawa, H. Harai, Y. Yokota, and Y. Naruse, “Sparse Regression Model to Predict a Server Load for Dynamic Adjustments of Server Resources,” Proc. IEEE ICIN, Feb. 2019.
[6] T. Hirayama, M. Jibiki, and V.P. Kafle, “Regressor Relearning Architecture Adapting to Traffic Trend Changes in NFV Platforms,” Proc. IEEE Conference on Network Softwarization (Netsoft), pp.272–276, June 2020.
[7] Y. Li, H. Liu, W. Yang, D. Hu, and W. Xu, “Inter-Data-Center Network Traffic Prediction with Elephant Flows,” Proc. IEEE/IFIP NOMS, April 2016.
[8] W. Wang, Y. Bai, C. Yu, Y. Gu, P. Peng, X. Wang, and R. Wang, “A Network Traffic Flow Prediction with Deep Learning Approach for Large-Scale Metropolitan Area Network,” Proc. IEEE/IFIP NOMS, April 2018.
[9] S.E. Hammami, H. Afifi, M. Marot, and V. Gauthier, “Network Planning Tool Based on Network Classification and Load Prediction,” Proc. IEEE WCNC, April 2016.
[10] F. Morales, M. Ruiz, L. Gifre, L.M. Contreras, V. L ´opez, and L. Velasco, “Virtual Network Topology Adaptability Based on Data Analytics for Traffic Prediction,” IEEE J. Opt. Commun. Netw., vol.9, no.1, pp.A35–A45, Jan. 2017.
[11] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, “Spatiotemporal Modeling and Prediction in Cellular Networks: A Big Data Enabled Deep Learning Approach,” Proc. IEEE Conference on Computer Communications (INFOCOM), pp.1–9, May 2017.
[12] A.Y. Nikravesh, S.A. Ajila, C.-H. Jung, and W. Ding, “Mobile Network Traffic Prediction Using MLP, MLPWD, and SVM,” Proc. IEEE International Congress on Big Data (BigData 216),
[13] M. Brehm and R. Prakash, “Proactive Resource Allocation Optimization in LTE with Inter-Cell Interference Coordination,” Springer Wireless Network, vol.20, pp.945–960, July 2014.

[14] P. Martinez-Julia, V.P. Kafe, and H. Harai, “Anticipating Minumum Resources Needed to Avoid Service Disruption of Emergency Support Systems,” Proc. IEEE International Conference on Innovation in Clouds, Internet and Networks (ICIN), Feb. 2018.

[15] A. Adas, “Traffic Models in Broadband Networks,” IEEE Commun. Mag., vol.35, no.7, pp.82–89, July 1997.

[16] Y. Qian, J. Xia, K. Fu, and R. Zhang, “Network Traffic Forecasting by Support Vector Machines based on Empirical Mode Decomposition Denoising,” Proc. International Conference on Consumer Electronics, Communications and Networks (CECNet), April 2012.

[17] H. Xia, X. Wei, Y. Gao, and H. Lv, “Traffic Prediction Based on Ensemble Machine Learning Strategies with Bagging and LightGBM,” Proc. IEEE International Conference on Communications Workshops (ICC Workshops), May 2019.

[18] A. Bayati, V. Asghari, K. Nguyen, and M. Cheriet, “Gaussian Process Regression Based Traffic Modeling and Prediction in High-Speed Networks,” Proc. IEEE GLOBECOM, Dec. 2016.

[19] R. Tibshirani, “Regression Shrinkage and Selection via the LASSO,” Journal of the Royal Statistical Society Series B (Methodological), vol.58, no.1, pp.267–288, 1996.

[20] T. Hastie, R. Tibshirani, and M. Wainwright, “Statistical Learning with Sparsity: the Lasso and Generalizations,” CRC press, May 2015.

[21] Y. Yokota, Y. Igarashi, M. Okada, and Y. Naruse, “Estimation of the Reaction Times in Tasks of Varying Difficulty from the Phase Coherence of the Auditory Steady-State Response Using the Least Absolute Shrinkage and Selection Operator Analysis,” Proc. International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp.6662–6666, Aug. 2015.

[22] S. Panicacci, M. Donati, L. Fanucci, I. Bellin, F. Profili, and P. Francesconi, “Population Health Management Exploiting Machine Learning Algorithms to Identify High-Risk Patients,” Proc. IEEE International Symposium on Computer-Based Medical Systems (CBMS), pp.298–303, June 2018.

[23] D.H. Walpert, “Stacked Generalization,” Neural Networks, vol.5, no.2, pp.241–259, 1992.

[24] X. Huang, Y. Ye, L. Xiong, R.Y.K. Lau, N. Jiang, and S. Wang, “Time series k-means: A new k-means type smooth subspace clustering for time series data,” Information Sciences, vol.367-368, pp.1–13, Nov. 2016.

[25] S. Hochreiter and J. Schmidhuber, “Long Short-term Memory,” Neural Computation, vol.9, no.8, pp.1735–1780, Dec. 1997.

[26] H. Zou and T. Hastie, “Regularization and Variable Selection via the Elastic Net,” Journal of Royal Statistical Society Series B, vol.67, no.2, pp.301–320, March 2005.

[27] L. Breiman, “Random Forests,” Machine Learning, vol.45, pp.5–32, Oct. 2001.

[28] J. Lee et al., “Deep Neural Networks as Gaussian Processes,” Proc. ICLR, May 2018.

Takahiro Hirayama received a M.S. Degree and Ph.D. Degree from Osaka University in 2010, 2013, respectively. In April 2013, he joined National Institute of Information and Communications Technology (NICT) and is now a researcher of Network System Institute in NICT. He received the IEICE Communications Society Best Paper Award in 2019. His research interests are in optical networks, software defined networking (SDN), network function virtualization (NFV), and machine learning (ML) for network management. He is a member of IEICE and IEEE.

Takaya Miyazawa received an M.E. and a Ph.D. in Information and Computer Science from Keio University, Yokohama, Japan, in 2004 and 2006, respectively. From April 2006 to March 2007, he was a visiting researcher at the University of California, Davis, USA. Since April 2007, he has been working in NICT, Tokyo. From April 2019 to July 2020, he had been transferred to Ministry of Internal Affairs and Communications (MIC), Japan, as a deputy director, and returned to NICT in August 2020. He served as a secretary of IEEE ComSoc Asia-Pacific Board from 2016 to 2017. He is a recipient of the 2007 Hiroshi Ando Memorial Young Engineer Award, the 2010 Funai Young Researcher Award, and Best Paper Award at the ITU Kaleidoscope Academic Conference in 2018. His research interests include network control, management and virtualization technologies. He is a member of IEICE.

Masahiro Jibiki received the Ph.D. degree in systems management from the University of Tsukuba, Japan, in 2003. In 1992, he joined NEC Corporation and was a researcher in the Central Research Laboratories until 2011. From 2006 to 2009, he was also a visiting professor at the University of Wakayama, Japan. Currently, he is a senior researcher in the National Institute of Information and Communications Technology. He received the IEICE Communications Society Best Paper Award in 2010 and 2019, respectively. His research interests include networking, software science, and mathematical model.

Ved P. Kafe received a B.E. in Electronics and Electrical Communications from Punjab Engineering College, Chandigarh, India, an M.S. in Computer Science and Engineering from Seoul National University, South Korea, and a Ph.D. in Informatics from the Graduate University for Advanced Studies, Japan. He is currently a research manager at NICT, Tokyo, and concurrently holds a visiting associate professor’s position at the University of Electro-Communications, Tokyo. He has been serving as a Co-rapporteur of ITU-T Study Group 13 since 2014. His research interests include new network architectures, network resource control automation, Internet of things, application of artificial intelligence and machine learning for the automation of 5G and beyond communication networks. He received the ITU Association of Japan’s Encouragement Award and Accomplishment Award in 2009 and 2017, respectively, and four Best Paper Awards at the ITU Kaleidoscope Academic Conferences in 2009, 2014, 2018 and 2020. He is an ITU-T SG13 Fellow, IEEE senior member and IEICE member.