Gaussian process regression as a predictive model for Quality-of-Service in Web service systems

Jakub M. Tomczak
Wrocław University of Technology
Wrocław, Poland
jakub.tomczak@pwr.wroc.pl

Jerzy Świątek
Wrocław University of Technology
Wrocław, Poland
jerzy.swiatek@pwr.wroc.pl

Krzysztof Latawiec
Opole University of Technology
Opole, Poland
k.latawiec@po.opole.pl

Abstract

In this paper, we present the *Gaussian process regression* as the predictive model for *Quality-of-Service* (QoS) attributes in Web service systems. The goal is to predict performance of the execution system expressed as QoS attributes given existing execution system, service repository, and inputs, e.g., streams of requests. In order to evaluate the performance of Gaussian process regression the simulation environment was developed. Two quality indexes were used, namely, Mean Absolute Error and Mean Squared Error. The results obtained within the experiment show that the Gaussian process performed the best with linear kernel and statistically significantly better comparing to *Classification and Regression Trees* (CART) method.

1 Introduction

Performance prediction in web service systems is one of the most important issues in modern computer networks which is still insufficiently solved by well-known methods because of the gap between theoretical considerations and applications. This research explores this issue. In general, web service systems consist of the following layers [17]: (i) an *execution layer* which controls the execution of composite Web services and manages dataflow between them, and (ii) an *application service layer* which delivers requested functionalities to clients. Web services are designed according to service oriented computing (SOC) paradigm [9] and represent encapsulated functionalities of applications.

In this paper, we focus on the execution layer only. Our goal is to predict performance of the execution system expressed as *Quality-of-Service* (QoS) attributes given existing execution system, service repository, and inputs, e.g., streams of requests. The predicted performance can be used not only for personalization of services [19] but most of all for service selection [10, 15] and resource allocation [13, 22]. For example, modelling dependency between QoS attributes and streams of requests allows to allocate computational resources in an optimal way. Otherwise other techniques are needed, e.g., change detection methods [14, 18, 20]. However, a predictive model for QoS attributes can be used as an objective function in an optimization task for resource allocation.

According to above facts the proposition of the predictive model becomes a crucial issue. It can be assumed that the execution system with fixed computational resources and given inputs performs roughly in a deterministic manner. Nevertheless, internal and unknown processes within the execu-

1

tion system introduce random noise and thus the QoS attributes are random variables as well. Hence, a probabilistic model seems to be the best suited in the considered application.

Recently, in the literature of machine learning, a non-parametric regression model called *Gaussian process* was introduced [3, 8, 12]. Gaussian processes are considered as one of the most successful regression models applied in many domains, e.g., biosystems [1], predictive control for chemical plants [7], hydraulic systems [6], learning inverted pendulum [5] and non-linear system identification [21].

The main contribution of this paper is twofold. First, an application of Gaussian process to predicting performance of the execution system in Web service system is presented. Second, a simulation environment for Web traffic is proposed.

The paper is organized as follows. In Sect. 2 the problem of performance prediction in Web service systems is stated. In Sect. 3 details about Gaussian processes models are outlined. In Sect. 3 the simulation environment is described and experiments are conducted. At the end conclusions are drawn.

2 Prediction of QoS in Web service systems

Let $\mathbf{x} \in \mathcal{X}$ denote a D-dimensional vector of input variables to the execution system. For example, inputs are total sizes of demands from D classes maintained in queues to the execution system, $\mathcal{X} = \mathbb{R}_+^D$. Outputs of the execution systems are denoted by $\mathbf{y} \in \mathcal{Y}$ and correspond to QoS attributes, e.g., time spent in the execution system (so called *latency*).[1]

Further, we assume that there exists a dependency between inputs and outputs. However, without knowing processes responsible for generating teletraffic and internal processes governing the execution systems we should consider noise in the model. The dependency between inputs and QoS attributes can be seen as a regression model, hence the target variable y is given by a function $f : \mathcal{X} \rightarrow \mathbb{R}$ and a Gaussian additive noise

$$y = f(\mathbf{x}) + \varepsilon, \tag{1}$$

where ε is a zero mean Gaussian random variable with precision β^{-1}, $\varepsilon \sim \mathcal{N}(\cdot|0, \beta^{-1})$.

The prediction task is to return an output $\hat{y}$ for given new inputs $\hat{\mathbf{x}}$ and N historical observations (data) $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$. Because we consider a probabilistic model (1), we need to calculate the following predictive distribution

$$p(\hat{y}|\hat{\mathbf{x}}, \mathcal{D}) = \int p(\hat{y}|\hat{\mathbf{x}}, f, \mathcal{D})p(f|\mathcal{D})\mathrm{d}f. \tag{2}$$

In order to calculate the predictive distribution we are supposed to give *a priori* distribution of dependencies f. As we will see shortly, this is analytically tractable as long as prior is Gaussian.

In the context of Web service systems, we want to predict QoS attributes, e.g., latency, for given inputs. Here we do not consider dynamics of input streams, thus the efficiency of our approach relies on proper formulation of input variables and calculation of target variable.

3 Gaussian process regression

3.1 The model

The idea of Gaussian processes is to put a prior distribution on function f and learn the dependencies basing on available data [3, 8, 12]. However, the Gaussian process is a non-parametric model and thus there is no need to formulate any fixed relationships between inputs and target variable.

---

[1]Multivariate regression model can be treated as a problem of several one-dimensional regression models [3] and thus, for further simplicity, we will consider only one output (target variable), i.e., y.

The non-linear regression model using Gaussian process (called *Gaussian process regression*) is as follows:

$$y = f(\mathbf{x}) + \varepsilon,$$
$$f \sim \mathcal{GP}(\cdot|0, k(\mathbf{x}, \mathbf{x}')),$$
$$\varepsilon \sim \mathcal{N}(\cdot|0, \beta^{-1}), \tag{3}$$

where $\mathcal{GP}$ denotes the Gaussian process, $k(\cdot, \cdot)$ is the covariance (kernel) function.

Now the predictive distribution (2) is analytically tractable because prior on f is $\mathcal{GP}$, *likelihood* is Gaussian, and *a posteriori* distribution on f is also a $\mathcal{GP}$.

Let $\mathbf{y}$ denote a column of output observations and $\mathbf{f}$ – a column of $f(\mathbf{x_n})$, $n = 1 \ldots N$. From the definition of Gaussian process the marginal distribution $p(\mathbf{f})$ is as follows:

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}), \tag{4}$$

where $\mathbf{0}$ – column of zeros, $\mathbf{K}$ – Gramm matrix, i.e., $K_{n,m} = k(\mathbf{x}_n, \mathbf{x}_m)$.

Similarly, the distribution of $\mathbf{y}$ conditioned by $\mathbf{f}$ is the following:

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \beta^{-1}\mathbf{I}_N), \tag{5}$$

where $\mathbf{I}_N$ is $N \times N$ unit matrix.

The marginal distribution $p(\mathbf{y})$ equals

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{C}), \tag{6}$$

where $\mathbf{C}$ is a matrix such that $C_{n,m} = K_{n,m} + \beta^{-1}\delta_{n,m}$, and $\delta_{n,m}$ is a Kronecker's delta.

Because all distributions are Gaussian, hence the predictive distribution[2] $p(\hat{y}|\hat{\mathbf{x}}, \mathbf{y})$ is Gaussian distribution with mean and covariance given by [3, 12]

$$m(\hat{\mathbf{x}}) = \mathbf{k}^\top \mathbf{C}^{-1} \mathbf{y}, \tag{7}$$
$$\sigma^2(\hat{\mathbf{x}}) = k(\hat{\mathbf{x}}, \hat{\mathbf{x}}) + \beta^{-1} - \mathbf{k}^\top \mathbf{C}^{-1} \mathbf{k}, \tag{8}$$

where $\mathbf{k}$ is a vector with elements $k(x_n, \hat{x})$, $n = 1 \ldots N$.

Finally, for given inputs to the execution systems $\hat{\mathbf{x}}$ we have calculated the predictive distribution with mean and variance defined as (7) and (8), respectively. Gaussian probability density function has one mode which is in the same time mean value, hence the mean value (7) is the most probable value for given inputs and the variance (8) determines its uncertainty.

3.2 Covariance function

Crucial step in modelling any phenomenon using Gaussian processes is the determination of the kernel function. There are many kernel functions described in literature (see [12] for further details), e.g., linear kernel

$$k_{lin}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{A}^{-2} \mathbf{x}' \tag{9}$$

where $\mathbf{A}^{-2}$ is $D \times D$ diagonal matrix, squared exponential kernel

$$k_{se}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\{(\mathbf{x} - \mathbf{x}')^\top \mathbf{A}^{-2}(\mathbf{x} - \mathbf{x}')\} \tag{10}$$

---

[2]Here we use shorthand notation in comparison to equation (2).

where σ_f^2 is a bias, and complex kernels, for example,

$$k(\mathbf{x}, \mathbf{x}') = k_{se}(\mathbf{x}, \mathbf{x}') + k_{lin}(\mathbf{x}, \mathbf{x}') + b, \tag{11}$$

where b is a bias parameter. Choosing specific kernel allows to reflect different similarities between points (see Fig. 1).
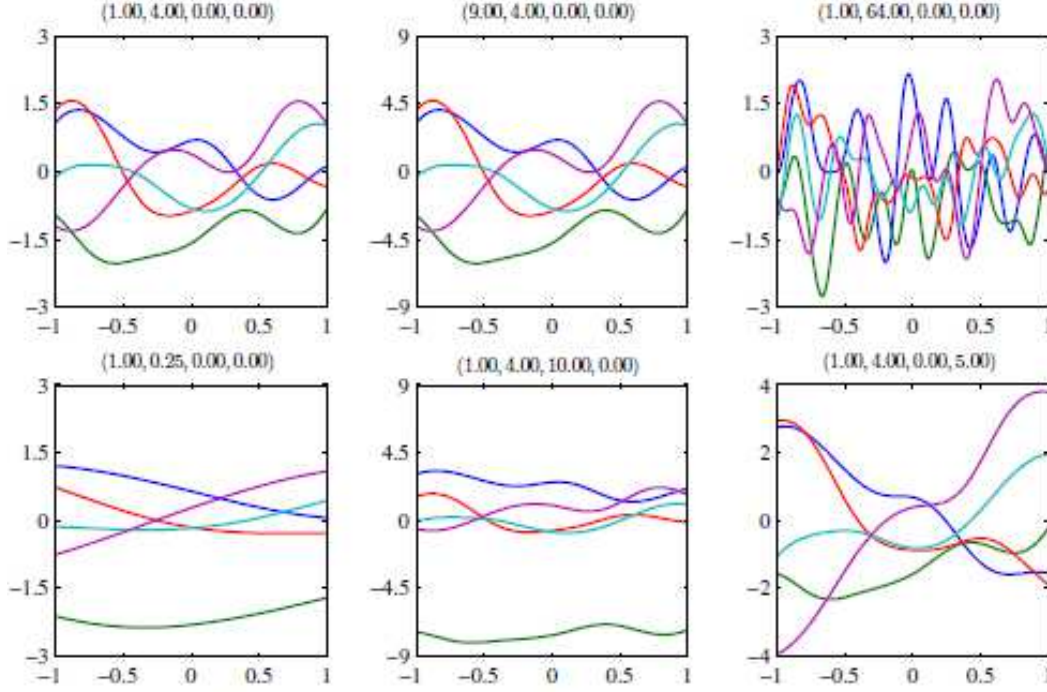

Figure 1: Examples from a Gaussian prior defined by the following complex covariance function: $k(\mathbf{x}, \mathbf{x}') = \theta_0 \exp\{-\frac{\theta_1}{2}\|\mathbf{x} - \mathbf{x}'\|^2)\} + \theta_2 + \theta_3 \mathbf{x}^\top \mathbf{x}'$. The title above each plot denotes $(\theta_0, \theta_1, \theta_2, \theta_3)$. Figure taken from [3].

3.3 Learning the hyperparameters

The predictions of a Gaussian process regression depend mainly on the choice of the covariance function. From the practical point of view it is more convenient to propose a parametric set of covariance functions than to fix the covariance function by hand. Then the inference of the values of hyperparameters can be conducted basing entirely on data.

In this paper we use the type 2 maximum likelihood procedure which allows to determine hyperparameters values by maximizing the log likelihood function ($\boldsymbol{\theta}$ denotes a vector of hyperparameters)

$$\ln p(\mathbf{y}|\boldsymbol{\theta}) = -\frac{1}{2}\ln|\mathbf{C}| - \frac{1}{2}\mathbf{y}^\top \mathbf{C}^{-1}\mathbf{y} - \frac{N}{2}\ln(2\pi). \tag{12}$$

If the evaluation of derivatives of $\mathbf{C}$ is straightforward we can easily calculate the following derivatives [3]

$$\frac{\partial}{\partial \theta_i}\ln p(\mathbf{y}|\boldsymbol{\theta}) = -\frac{1}{2}\mathrm{Tr}\left(\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\right) + \frac{1}{2}\mathbf{y}^\top \mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\mathbf{C}^{-1}\mathbf{y}. \tag{13}$$

4

4 Experiments

4.1 Preliminaries

The purpose of the experiment is to examine the prediction quality of the proposed approach. We take under consideration a Web service execution environment and average latency of services' responses in the system as an output QoS attribute. To reflect the nature of real Web service execution system we propose a simulation environment written in Matlab$^{®}$. The simulation model is presented in Fig. 2. The model consists of the following components: (i) *teletraffic generator* (TG), which imitates clients' behaviour by generating Web service requests, (ii) *scheduler*, which distributes service requests to proper queues, (iii) *queues*, which maintain demands and work in FIFO fashion,(iv) *Round Robin* (RR), which collects requests from queues in circular order [16], (v) *execution system*, which executes the services.
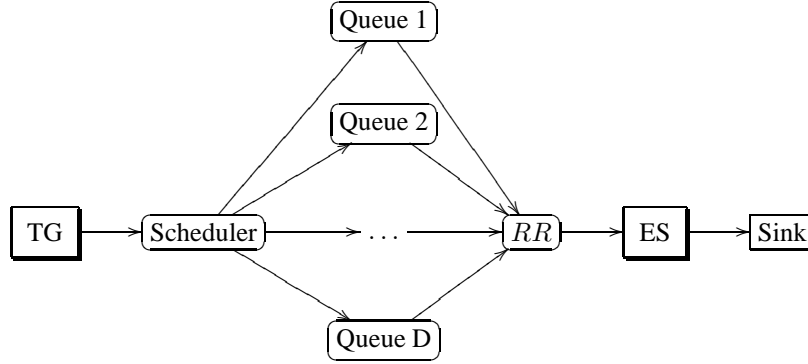
Figure 2: Schematic diagram of the simulation environment, TG – teletraffic generator, RR – Round Robin, ES – execution system.

The simulation environment provides teletraffic: inputs $\mathbf{x}$ to the execution system which are sizes of queues, and outputs of the execution systems which are average latencies y. Each quantity is calculated using T last observations. In the experiment we compare Gaussian process regression with well-known *Classification And Regression Trees* (CART) method [4] which is a baseline in the experiment. We use Matlab implementation of CART and a toolbox for Gaussian processes provided by Rasmussen and Nickisch [11].

4.2 Simulation details

4.2.1 Modelling teletraffic

We assume that each demand to the system can occur with a probability p. Then the class of the demand is generated with uniform probability and the size of the demand is drawn from lognormal distribution.[3] The process of generating a demand is as follows (assuming some universal time unit, e.g., one second):

1. Generate random number from interval $[0, 1]$. If it is greater than p, then go to step 2. Otherwise go to step 1.
2. Generate the demand class using uniform probability.
3. Generate the size of the demand using lognormal distribution. Go to step 1.

4.2.2 Modelling execution system

One demand arrives to the execution system according to the Round Robin scheduler. Execution of a demand takes as many universal time units as it is completely executed, i.e., the executed size

---

[3]According to [2], probability distribution function for the size of the file body is lognormal.

of the demand is zero. In the simulation environment we need to determine *execution sizes* per one universal time unit for each demand class.

4.2.3 Prediction and evaluation

In order to evaluate Gaussian process regression and CART we have to determine a number of training points (inputs with outputs) and a number of test points. We use the following quality indexes:

- Mean Absolute Error (MAE);
- Mean Squared Error (MSE).

We allow three covariance functions in the simulation environment, i.e., linear kernel (9), squared exponential kernel (10) and complex kernel expressed as in equation (11). Moreover, we use Gaussian likelihood with precision β^{-1} (called *noise* in the simulation environment), and type 2 maximum likelihood procedure for hyperparameters learning.
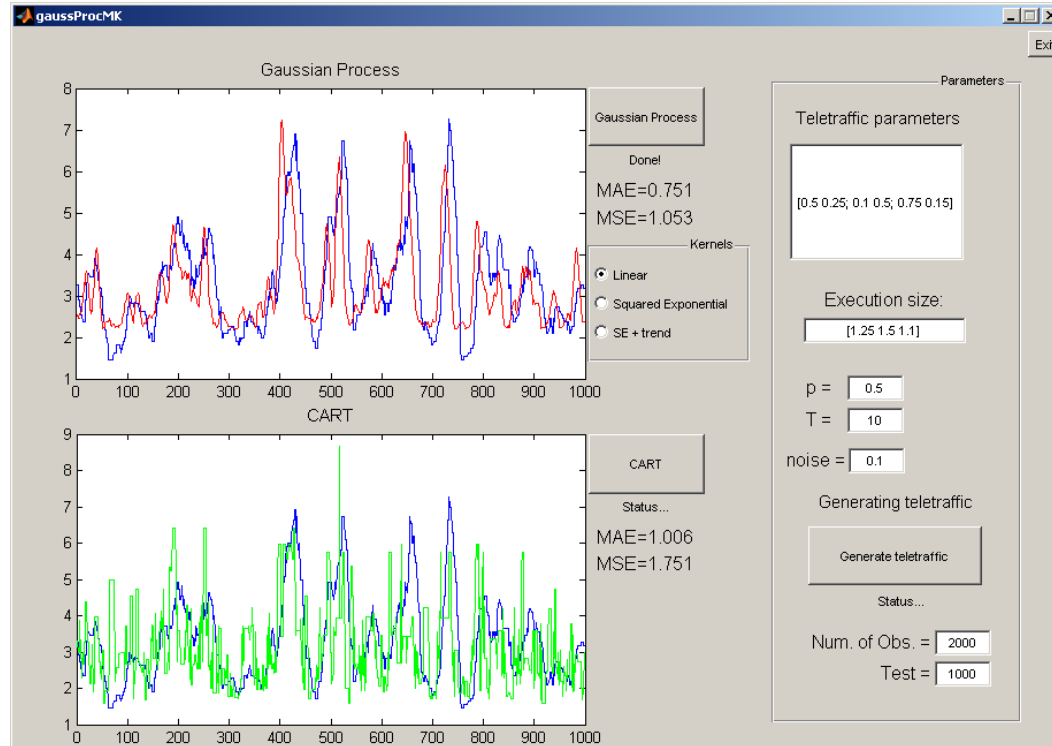

Figure 3: GUI of the simulation environment.

4.3 Results and discussion

The GUI of the simulation environment (see Fig. 3) allows to fix all parameters. In order to conduct experiments we generated 10 simulations of teletraffic with the following parameters:

- number of classes – 3;
- number of test points – 1000;
- number of training points – 1000;
- $p = 0.5$;
- $T = 10$;
- $\beta^{-1} = 0.1$;

6

- parameters of lognormal distribution – i) for class 1: $0.5, 0.25$, ii) for class 2: $0.1, 0.5$, iii) for class 3: $0.75, 0.15$;
- execution size per one universal time unit: i) for class 1: 1.25, ii) for class 2: 1.5, iii) for class 3: 1.1.

The results of 10 simulations are gathered and represented as a box-and-whisker plot in Fig 4 and 5 for MAE and MSE, respectively. The Gaussian process regression performed better in comparison to CART for any kernel function. However, the linear kernel appeared to be more proper to represent similarity between inputs than squared exponential or complex kernel. Especially, the complex kernel is a sum of linear and squared exponential kernels and that is why it performed slightly better than squared exponential.

In order to compare the results, we also performed two sample t-test for MAE at the 5% significance level between Gaussian process regression with linear kernel and CART. The null hypothesis, i.e., random samples share the same mean and equal but unknown variances, can be rejected with p-value equal 1.8406×10^{-5}. Similarly, for MSE, we can reject the null hypothesis with p-value equal 0.018. In other words, the Gaussian process regression performs statistically better than CART.
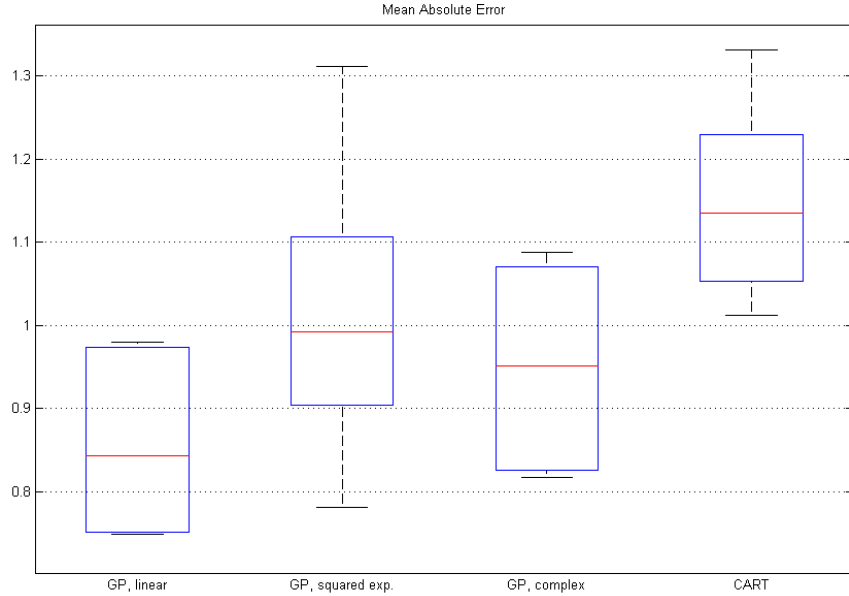

Figure 4: Box-and-whisker plot for MAE results. Red line represents median value, blue box – quantiles, and black lines – range of values.
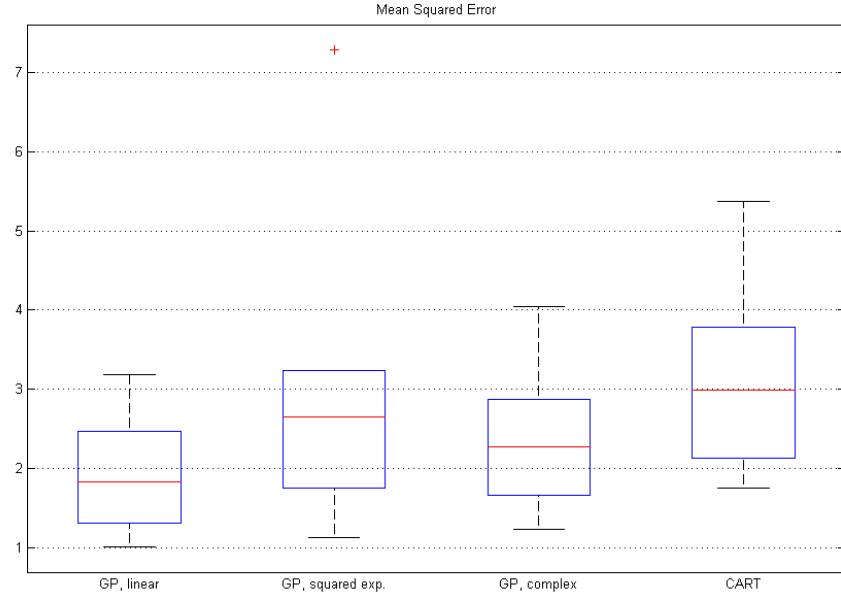
Figure 5: Box-and-whisker plot for MSE results. Red line represents median value, blue box – quantiles, black lines – range of values, and red cross – outlier.

5 Conclusions

In this paper, we have presented the Gaussian process regression as the predictive model for QoS attributes in Web service systems. The idea of Gaussian process regression is well-grounded in the field of machine learning but its application in Web service systems is novel. In order to evaluate the performance of Gaussian process regression the simulation environment was developed. Two quality indexes were used, namely, Mean Absolute Error and Mean Squared Error. The results show that the Gaussian process performed the best with linear kernel and statistically better comparing to CART method.

The proposed approach shows that application of machine learning methods can develop existing computer network systems. The results presented in the paper indicate high accuracy but further research and experiments, especially on existing systems, are necessary. Summing up, this paper tries to fill the gap between machine learning methods and computer network applications.

References

[1] K. Ažman and J. Kocijan. Application of Gaussian processes for black-box modelling of biosystems. *ISA Transactions*, 46:443–457, 2007.

[2] P. Barford and M. Crovella. Generating representative web workloads for network and server performance evaluation. *ACM SIGMETRICS Perform. Eval. Rev.*, 26(1):151–160, 1998.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, Singapore, 2006.

[4] L. Breiman, J. Friedman, R. Olshen, C. Stone, D. Steinberg, and P. Colla. CART: Classification and regression trees. *Wadsworth: Belmont, CA*, 1983.

[5] M. P. Deisenroth and C. E. Rasmussen. Efficient reinforcement learning for motor control. In *10th International PhD Workshop on Systems and Control*. Hluboká nad Vltavou, Czech Rep., 2009.

[6] G. Gregorčiča and G. Lightbody. Gaussian process approach for modelling of nonlinear systems. *Engineering Applications of Artificial Intelligence*, 22:522–533, 2009.

[7] B. Likar and J. Kocijan. Predictive control of a gas-liquid separation plant based on a Gaussian process model. *Computers and Chemical Engineering*, 31:142–152, 2007.

[8] D. J. C. MacKay. Introduction to Gaussian processes. *NATO ASI series. Series F: Computer and System Sciences*, 168:133–165, 1998.

[9] M. P. Papazoglou and D. Georgakopoulos. Service-Oriented Computing. *Communications of the ACM*, 46(10):25–28, 2003.

[10] A. Prusiewicz and M. Zieba. On some method for limited services selection. *International Journal of Intelligent Information and Database Systems*, 5(5):493–509, 2011.

[11] C. E. Rasmussen and H. Nickisch. Gaussian processes for machine learning (GPML) toolbox. *The Journal of Machine Learning Research*, 11:3011–3015, 2010.

[12] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, London, 2006.

[13] P. Rygielski and A. Gonczarek. Migration-aware optimization of virtualized computational resources allocation in complex systems. In *Systems Engineering (ICSEng), 2011 21st International Conference on*, pages 212–216. IEEE, 2011.

[14] P. Rygielski and J. M. Tomczak. Context change detection for resource allocation in service-oriented systems. *Lecture Notes in Computer Science*, 6882:591–600, 2011.

[15] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei. Personalized QoS prediction for Web services via collaborative filtering. In *IEEE International Conference on Web Services*, pages 439–446, 2007.

[16] A. Silberschatz, P. Galvin, and G. Gagne. *Operating system concepts*. Addison-Wesley, 1998.

[17] A. Tari, I. Elgedawy, and A. Dahmani. A dual-layered model for web services representation and composition. *J Intell Inf Syst*, 32:237–265, 2009.

[18] J. M. Tomczak. On-line change detection for resource allocation in Service-Oriented Systems. *Technological Innovation for Value Creation*, pages 51–58, 2012.

[19] J. M. Tomczak and J. Świątek. Personalisation in Service-Oriented Systems using Markov chain model and Bayesian inference. *Technological Innovation for Sustainability*, pages 91–98, 2011.

[20] J. M. Tomczak and M. Zięba. On-line Bayesian context change detection in Web service systems. *Knowledge and Information Systems*, 2012. (In Press).

[21] R. Turner, M. P. Deisenroth, and C. E. Rasmussen. System identification in Gaussian process dynamical systems. In D. Görür, editor, *NIPS Workshop on Nonparametric Bayes*. Whistler, Canada, 2009.

[22] M. Welsh, D. Culler, and E. Brewer. SEDA: An architecture for well-conditioned, scalable internet services. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 230–243, 2001.