ICIC Express Letters
Volume **14**, Number **8**, August **2020**

 ISSN 1881-803X
pp. **785–791**

HYPER-PARAMETER OPTIMIZATION OF GATED CNN WITH GAUSSIAN PROCESS REGRESSION

Hidekazu Yanagimoto[1], Makoto Okada[2] and Kiyota Hashimoto[3]

[1]College of Sustainable System Sciences
[2]College of Engineering
Osaka Prefecture University
1-1 Gakuen-cho, Naka-ku, Sakai, Osaka 599-8531, Japan
hidekazu@kis.osakafu-u.ac.jp; okada@cs.osakafu-u.ac.jp

[3]Faculty of Technology and Environment
Prince of Songkla University
80 Moo 1 Vichitsongkram Road, Kathu, Phuket 83120, Thailand
Kiyota.hashimoto@gmail.com

Received February 2020; accepted May 2020

Abstract. *We propose hyper-parameter optimization of a deep neural language model, which consists of a stacked gated convolutional neural network and a multi-layer neural network, with Gaussian process regression. A deep neural network has a very complicated architecture generally and includes many hyper-parameters. The architecture complexity makes hyper-parameter optimization difficult. So we realize the hyper-parameter optimization with Gaussian process regression. The proposed method automatically selects a better setting according to performance estimation scores. We confirmed the proposed method was effective by evaluation experience.*
Keywords: Natural language model, Deep learning, Gaussian process regression

1. **Introduction.** An enormous amount of data is created on a computer network and you need to analyze them for decision making. It is impossible to analyze them manually and you need to employ machine learning. Especially, deep learning is applied to various research areas, for example, image processing, signal processing, and natural language processing. It achieves state-of-the-art performance. So it is important to design a deep neural network with the optimal architecture.

A deep neural network with optimal architecture can achieve the highest performance. So a relation between the architecture and the final performance has to be defined to search the optimal architecture. In usual neural networks, the architecture is adjusted with some hyper-parameters, for example, the number of layers, and activation functions. Searching the optimal architecture is regarded as hyper-parameter optimization. However, it is not clear how to determine hyper-parameters and you usually have to employ brute-force approaches because of the relation between the hyper-parameters and the final performance. So hyper-parameter optimization is usually a very time-consuming task. Bengio and Bergstra [1,2] told that hyper-parameter optimization was executed with a random search because it is very difficult to explore hyper-parameters with grid search. Another hyper-parameter optimization approach is Bayesian optimization [3-5]. Bayesian optimization is one of blackbox optimization approaches and can optimize hyper-parameters in machine learning.

Our proposed method is one of the blackbox optimization approaches and employs Gaussian process regression [4] to define a relation between hyper-parameters and the final performance. Gaussian process regression estimates a mean and a variance with

DOI: 10.24507/icicel.14.08.785

respect to a hyper-parameter setting. So we determine the next candidate of the optimal hyper-parameter based on the mean and the variance and check the performance for the hyper-parameter. Especially, we apply the proposed method to natural language processing and discuss how effective Gaussian process regression is to optimize a neural language model. In evaluation experiments, we optimize a few hyper-parameters with the proposed method. This study contributes to the topics below.

- We apply Gaussian process regression to hyper-parameter optimization of a deep neural network. And we discuss how effective the proposed method is to optimize neural network architecture. This discussion helps designers of deep learning to determine some hyper-parameters.
- We apply Gaussian process regression to a natural language task. In this paper, we predict sentiment polarities for short texts, tweets. The sentiment polarity prediction system is Gated CNN, which is one of deep neural networks, and achieves good prediction accuracy.

This paper is constructed below. In Section 2, we explain the proposed method. Especially, we describe Gaussian process regression, Gated CNN [5], and the combination of Gaussian process regression and Gated CNN. In Section 3, we carry out evaluation experiments with twitter corpus. The corpus consists of short text and its sentiment polarity and is divided into training data and test data. We calculate the polarity prediction accuracy of test data and discuss the performance of the proposed method. In Section 4, we describe research achievement and future works.

2. Hyper-Parameter Optimization with Gaussian Process Regression.

In this section, we describe out proposed method, hyper-parameter optimization with Gaussian process regression. We explain a basic concept of Gaussian process regression and a gated convolutional neural network (Gated CNN).

2.1. Gaussian process regression.

In linear regression, we estimate an output as the weighted sum of input.

$$\hat{\boldsymbol{y}} = \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x})$$

where $\phi(\boldsymbol{x})$ is a basis function. The parameter $\boldsymbol{w}$ is determined with the least squared error.

$$C = \sum_i (\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i)^2$$

In this situation, we regard $\boldsymbol{w}$ is generated from Gaussian distribution.

$$\boldsymbol{w} \sim N\left(0, \lambda^2 I\right)$$

When we apply this probabilistic model to linear regression, we can obtain the following equation.

$$\boldsymbol{y} \sim N\left(\mu(\boldsymbol{x}), \lambda^2 \Phi\Phi^{\mathrm{T}}\right)$$

In this case, every $\boldsymbol{y}$ is generated from Gaussian distribution and this model is regarded as Gaussian process. In the distribution, $\mu(\boldsymbol{x})$ is determined from observations but $\Phi\Phi^{\mathrm{T}}$ is not determined from inputs directly. We define $\Phi\Phi^{\mathrm{T}}$ as a Gaussian kernel function. The Gaussian kernel function is defined below.

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \theta_1 \exp\left(-\frac{|\boldsymbol{x}_1 - \boldsymbol{x}_2|}{\theta_2}\right)$$

where θ_1 and θ_2 are predefined parameters.

When we regard $\boldsymbol{y}_i$ as $f(\boldsymbol{x}_i)$, which is the output of a function, we can construct a function $f(\boldsymbol{x})$ using $\boldsymbol{x}$ and $\boldsymbol{y}$. In this case, we call this model Gaussian process regression. Improving the simple Gaussian process regression, we can predict unobserved outputs of

the function, $f(\boldsymbol{x})$. Now observations are $(\boldsymbol{x}, \boldsymbol{y})$ and unobserved data are $(\boldsymbol{x}^*, \boldsymbol{y}^*)$. In this case, we model $(\boldsymbol{y}, \boldsymbol{y}^*)$ using the previous approach and we obtain the following model.

$$\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{y}^* \end{pmatrix} \sim N\left(\begin{pmatrix} f(\boldsymbol{x}) \\ f(\boldsymbol{x}^*) \end{pmatrix}, \begin{pmatrix} K_{NN} & K_{N*} \\ K_{*N} & K_{**} \end{pmatrix}\right)$$

where $K_{NN} = k(\boldsymbol{x}, \boldsymbol{x})$, $K_{N*} = k(\boldsymbol{x}, \boldsymbol{x}^*)$, and $K_{**} = k(\boldsymbol{x}^*, \boldsymbol{x}^*)$. Using the model, we calculate $p(\boldsymbol{y}^*|\boldsymbol{y})$.

$$p(\boldsymbol{y}^*|\boldsymbol{y}) \sim N\left(K_{*N}K_{NN}^{-1}f(\boldsymbol{x}), K_{**} - K_{*N}K_{NN}^{-1}K_{N*}\right)$$

Using the formulation, we can predict unobserved data with Gaussian process regression.

The Gaussian process regression can predict the output of a function and the variance of the output. So we can choose the next candidate based on the prediction and its variance. One of the selection strategies is the upper confidence bound. The upper confidence bound estimates outputs considering a mean and its variance. The upper confidence bound is defined below.

$$a_{UCB}(\mu, \sigma, N) = \mu + \sqrt{\frac{\log N}{N}}\sigma$$

The upper confidence bound is an optimistic estimation for prediction. So a candidate tends to be selected including a large variance and it means that uncertain parameters are selected.

2.2. **Gated convolutional neural network.** We explain a gated convolutional neural network (Gated CNN), which is an important part of the proposed method. In Figure 1, Gated CNN is applied to pairs of words in sentences repeatedly and generates semantic vectors. In natural language processing, LSTM is often used to construct a language model from a text corpus. LSTM has three gates to control the information flow: input, output, and forget gates. By preparing these three gates, LSTM enables learning to forget, or throw away, some learning results and to choose more appropriate learning results, which is desirable for language model construction. Gated CNN introduces this mechanism as a gated mechanism to CNN.

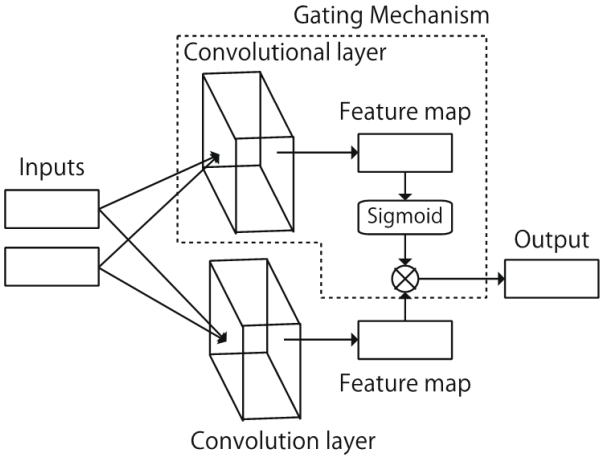

FIGURE 1. The architecture of gated convolutional neural network

As a general CNN accepts a matrix as an input, Gated CNN accepts a matrix that consists of multiple vectors. The input is transformed into two different vectors with two different kernels (filters). One vector includes semantic information from input data and the other vector describes the information to control gates in Gated CNN. The process is mathematically defined as follows:

$$H = \sigma(X * F_g) \otimes (X * F_c)$$

where $X \in \mathbb{R}^{N \times m}$ is the input of Gated CNN, $F_g \in \mathbb{R}^{m \times 1 \times n}$ and $F_c \in \mathbb{R}^{m \times 1 \times n}$ are kernels for 1D convolution operation, $\sigma(\cdot)$ is the sigmoid function, $*$ is the convolution operation, and $\otimes$ is the element-wise product between vectors. So, the output is $H \in \mathbb{R}^{(N-m+1) \times m}$, which is shorter than the length of the input X. $\sigma(X * F_g)$ is a scalar value between 1 and 0 and works like a gate. Whole parameters, F_g and F_c, are trained with training data. Especially, m and n denote embedding size and kernel size respectively and are hyper-parameters.

On the other hand, a sentiment analysis system consists of some Gated CNNs and a three-layer neural network in this paper. In Figure 2, the architecture of our proposed system is shown. The system includes Gated CNN to generate feature vectors from the input text. The three-layer neural network classified input text according to its sentiment polarity. To make more complex feature vectors, we stack more Gated CNNs in the proposed system. The stacking approach affects the final classification accuracy. The system includes some hyper-parameters: stacking size, hidden layer size in 3NN.

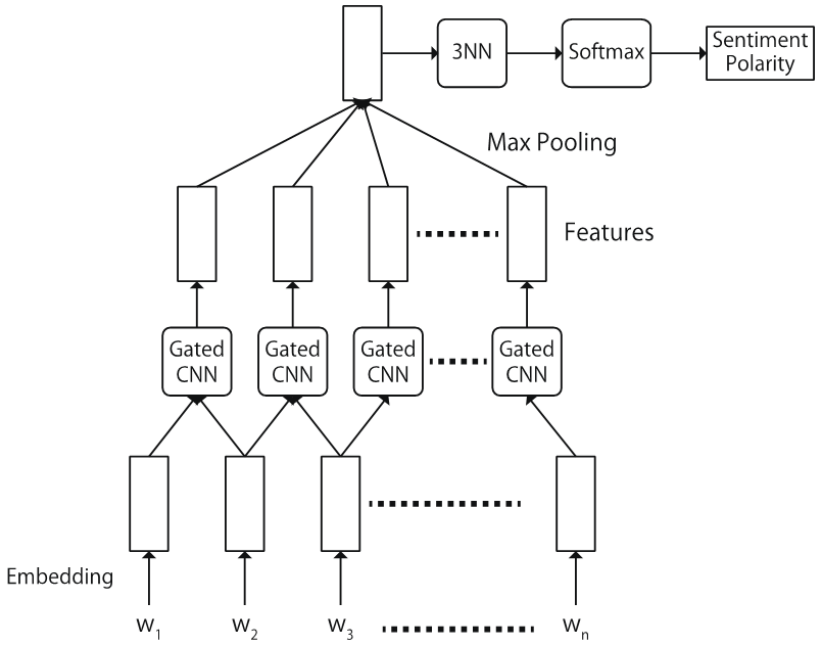

FIGURE 2. Sentiment analysis with gated convolutional neural network

2.3. **Hyper-parameter optimization with Gaussian process regression.** The sentiment analysis system includes some hyper-parameters and we can train the system after determining the hyper-parameters. In this study, we model a prediction accuracy, which the trained system achieves, with Gaussian process regression. After obtaining some observations, we can predict every accuracy which the system with every hyper-parameter

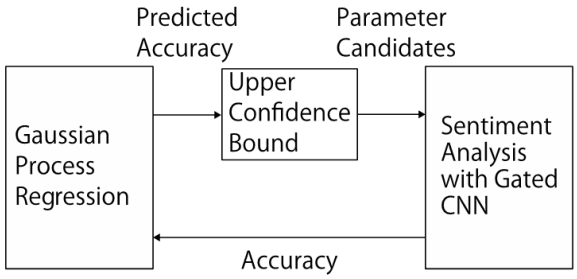

FIGURE 3. Hyper-parameter optimization with Gaussian process regression for sentiment analysis with Gated CNN

setting achieves. Using upper confidence bound, we select a hyper-parameter setting to train the system.

3. **Experiments.** In this section, we explain our evaluation experiments and discuss the performance of the proposed method. Firstly, we explain the dataset for the evaluation experiments. Next, we describe some hyper-parameters for the proposed method. Then, we tell a result of hyper-parameter optimization.

3.1. **Dataset.** We evaluate the proposed method with sentiment analysis corpus. The corpus consist of tweets and all tweets have sentiment polarity which is judged by human annotators. Especially, the corpus does not include neutral tweets which means no sentiment polarity. The contents of the corpus are shown in Table 1. The corpus is a balanced corpus and includes positive tweets and negative tweets equally. The average words in a tweet are about 15 words.

Table 1. Datasets for experiment

	Total	Training	Test
Positive tweets	15,972	12,778	3,194
Negative tweets	15,987	12,790	3,197
Total	31,959	25,568	6,391
The average number of words in tweets	15.1 words		

3.2. **Parameter setting.** The proposed method includes some predefined parameters. Especially, Gated CNN includes many predefined parameters: kernel size, stacking size, stride size, and so on. However, it is hard to optimize all hyper-parameters with the proposed method because we need so much optimization time. In this paper, we optimize kernel size and stacking size. Optimization of other hyper-parameters is one of the future works. Additionally, hyper-parameters in Gaussian process regression have to be defined previously. In this paper, we defined the hyper-parameters in Gaussian process regression manually. The optimal parameters for Gaussian process regression are one of future works.

All hyper-parameters except some parameters for optimization are shown in Table 2. The minimum word frequency is used to determine whether a word is registered in vocabulary or not. When a word is not registered in the vocabulary, the word is replaced as a special symbol, <UNK>, which means an out-of-vocabulary word.

Table 2. Parameter settings

Hyper-parameter	Setting
Minimum word frequency	3
Word embedding size	256
Stride size in Gated CNN	1
Hidden layer size in 3-NN	100
Output layer size in 3-NN	2
Activation function in 3-NN	Sigmoid function
Epoch for training	5
Minibatch size	100
Optimization algorithm	Adam
Estimation with Gaussian process regression	10
The number of stacked Gated CNNs	$[1, 2, 3, 4, 5]$
The width of kernel in Gated CNN	$[2, 3, 4, 5]$

Hyper-parameters in kernel function for Gaussian process regression are automatically optimized with training data. Speaking concretely, θ_1 and θ_2 are optimized.

For hyper-parameter optimization in Gaussian process, we execute hyper-parameter estimation with Gaussian process regression 10 times. Hence, we obtain 10 estimators according to the upper confidence bound criterion. In this setting, we have to check over 1,000 patterns combining stacking size and kernel size. So we check only 10 patterns by Gaussian process regression.

3.3. **Results.** For visualization, we discuss only two stacking Gated CNNs. In Figure 4 we show the accuracy rates for each hyper-parameter setting. In this case, neural networks with more complex architecture have less accuracy. So it is a good solution to search simple hyper-parameter settings. Table 3 shows how the proposed method searches hyper-parameter settings. The result shows the proposed method searches hyper-parameters from simple settings to complex ones.

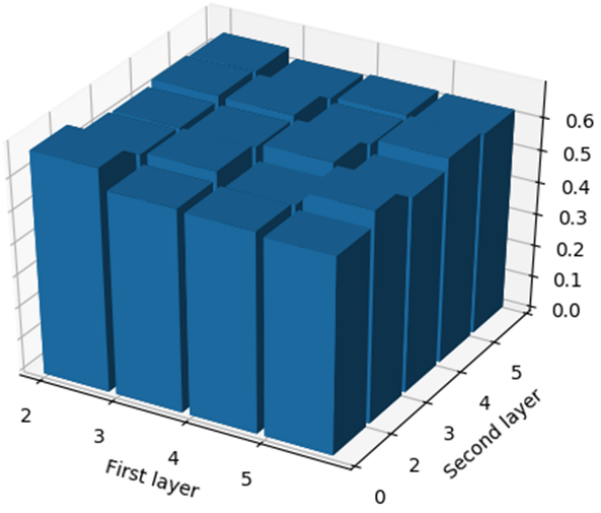

FIGURE 4. Accuracy rates for two stacking Gated CNNs

TABLE 3. Hyper-parameter setting exploration

Order	Hyper-parameter	Accuracy	Order	Hyper-parameter	Accuracy
1	$(2, 2)$	0.647	11	$(4, 2)$	0.614
2	$(2, 0)$	0.693	12	$(4, 3)$	0.653
3	$(3, 0)$	0.632	13	$(4, 4)$	0.622
4	$(3, 2)$	0.655	14	$(5, 3)$	0.607
5	$(3, 3)$	0.642	15	$(5, 4)$	0.638
6	$(2, 3)$	0.642	16	$(4, 5)$	0.627
7	$(2, 4)$	0.657	17	$(5, 5)$	0.624
8	$(3, 4)$	0.653	18	$(5, 2)$	0.658
9	$(3, 5)$	0.622	19	$(5, 0)$	0.615
10	$(2, 5)$	0.650	20	$(4, 0)$	0.626

We show the result of the proposed method for the original setting in Table 4. Using the hyper-parameter setting, the accuracy achieves about 72.0%. Especially, when we use a small stacking size, it takes about 30 minutes for training. However, when we use a large stacking size, it takes over 1 hour for training. It means that when we carry out a grid search, which evaluates all combinations of hyper-parameters, it takes so much time to find the optimal setting. On the other hand, Gaussian process regression reduces learning time to find the optimal hyper-parameter setting.

Table 4. Evaluation results

Estimated parameter	Accuracy
2-2	0.647
2	0.693
3	0.632
3-2	0.655
3-2-2	0.710
2-2-2	0.707
3-2-2-2	0.706
2-2-2-2	0.644
3-2-2-2-2	0.686
2-2-2-2-2	0.720

4. **Conclusions.** In this paper, we proposed a hyper-parameter optimization of neural network language model with Gaussian process regression. Basically, it takes much time to train a large neural network with large corpus and it is impossible to check various hyper-parameter settings. It means grid search is not an appropriate approach to optimize hyper-parameters. The proposed method can avoid the combination complexity of hyper-parameters and focus on promising settings. In the evaluation experiments, the proposed method can find a hyper-parameter setting automatically, which is similar to manually adjusted settings.

In this paper, many evaluations remain. In this paper, we restricted optimized hyper-parameters. For example, we adjust only stacking size and kernel size. In the future, we will optimize more hyper-parameters and discuss the performance of the proposed method. Moreover, training epoch of the neural network is only 5 because experiment time is short. In the future, we will increase training epoch and train a neural network more strictly.

REFERENCES

[1] Y. Bengio, Practical recommendations for gradient-based training of deep architectures, *arXiv:1206.5533v2*, 2012.

[2] J. Bergstra and Y. Bengio, Random search for hyper-parameter optimization, *JMLR*, vol.13, pp.281-305, 2012.

[3] J. Mockus, On Bayesian methods for seeking the extremum, *Proc. of IFIP Tech. Conf.*, pp.400-404, 1974.

[4] D. R. Jones, M. Schonlau and W. J. Welch, Efficient global optimization for expensive black-box functions, *Journal of Global Optimization*, vol.13, pp.455-492, 1998.

[5] J. Snoek, H. Larochelle and R. P. Adams, Practical Bayesian optimization of machine learning algorithm, *NIPS12*, pp.2951-2959, 2012.

[6] C. E. Rasmussen and C. K. I. Williams, *Gaussian Process for Machine Learning*, The MIT Press, 2006.

[7] Y. N. Dauphin, A. Fan, M. Auli and D. Grangier, Language modeling with gated convolutional neural network, *arXiv:1612.08083*, 2016.