AutoML from Service Provider’s Perspective: Multi-device, Multi-tenant Model Selection with GP-EI

Chen Yu
Department of Computer Science, University of Rochester

Bojan Karlaš
Department of Computer Science, ETH Zurich

Jie Zhong
Department of Mathematics, California State University Los Angeles

Ce Zhang
Department of Computer Science, ETH Zurich

Ji Liu
Department of Computer Science, University of Rochester

Abstract

AutoML has become a popular service that is provided by most leading cloud service providers today. In this paper, we focus on the AutoML problem from the service provider’s perspective, motivated by the following practical consideration: When an AutoML service needs to serve multiple users with multiple devices at the same time, how can we allocate these devices to users in an efficient way? We focus on GP-EI, one of the most popular algorithms for automatic model selection and hyperparameter tuning, used by systems such as Google Vizer. The technical contribution of this paper is the first multi-device, multi-tenant algorithm for GP-EI that is aware of multiple computation devices and multiple users sharing the same set of computation devices. Theoretically, given $N$ users and $M$ devices, we obtain a regret bound of $O((\text{MIU}(T, K) + M)\frac{N^2}{M})$, where $\text{MIU}(T, K)$ refers to the maximal incremental uncertainty up to time $T$ for the covariance matrix $K$. Empirically, we evaluate our algorithm on two applications of automatic model selection, and show that our algorithm significantly outperforms the strategy of serving users independently. Moreover, when multiple computation devices are available, we achieve near-linear speedup when the number of users is much larger than the number of devices.

1 INTRODUCTION

One of the next frontiers of machine learning research is its accessibility — How can we make machine learning systems easy to use such that users do not need to worry about decisions such as model selection and hyperparameter tuning as much as today? The industry’s answer to this question seems to be making AutoML services available on the cloud, and prominent examples include Google Cloud AutoML and Microsoft Cognitive Services. In these services, users are provided with a single interface for uploading the data, automatically dealing with hyperparameter tuning and/or model selection, and returning a model directly without any user intervention as shown in Figure 1. Bayesian optimization is one of the core techniques that make AutoML possible by strategically planning a series of models and hyperparameter configurations to tune and try.

From the service provider point of view, resource allocation is the problem coming together with the emerging popularity of such a service — When an online AutoML service needs to serve multiple users with limited number of devices, what is the most cost efficient way of allocating these devices to different users? During our conversation with multiple cloud service providers, many of them believe that an effective and cost efficient resource sharing could be of great practical interests.

---

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).
and is a natural technical question to ask.

In this paper, we focus on GP-EI, one of the most popular algorithms for AutoML that is used in systems such as Google Vizier [2017] and Spearmit [2017]. Specifically, we are interested in the scenarios that each user runs her own GP-EI instance on a different machine learning task, and there are multiple devices, each of which can only serve one user at the same time. How to allocate resources? What are the theoretical properties of such an algorithm?

The result of this paper is the first multi-device, multi-tenant, cost sensitive GP-EI algorithm that aims at optimizing for the “global happiness” of all users given limited resources. In order to analyze its performance, we introduce a new notation of Maximum Incremental Uncertainty (MIU) to measure dependence among all models. Given $N$ users and $M$ devices, the upper bound of cumulative regret is $O((\text{MIU}(T,K)+M)^{\frac{2}{\gamma}T})$, where $\text{MIU}(T,K)$ will be specified in Section 5.2. This bound converges to optimum and is nearly linear speedup when more devices are employed, which will be discussed also in Section 5.2.

We evaluate our algorithms on two real-world datasets: (1) model selection for image classification tasks that contains 22 datasets (users) and 8 different neural network architectures; and (2) model selection for models available on Microsoft Azure Machine Learning Studio that contains 17 datasets and 8 different machine learning models. We find that, multi-tenant GP-EI outperforms standard GP-EI serving users randomly or in a round robin fashion, sometimes by up to 5× in terms of the time it needs to reach the same “global happiness” of all users. When multiple devices are available, it can provide near linear speedups to the performance of our algorithm.

2 RELATED WORK

Multi-armed bandit problem There are lots of research on the multi-armed bandit problem. Some early work such as [Lai and Robbins, 1985] provided the lower bound of stochastic multi-armed bandit scenario and designed an algorithm to attain this lower bound. Then many research focused on improving constants of the bound and designing distribution-free-bound algorithms such as upper confidence bound (UCB) [Auer et al., 2002] and Minimax Optimal Strategy in the Stochastic case (MOSS) [Audibert and Bubeck, 2009]. UCB is now becoming a very important algorithm in bandit problem. Lots of algorithms are based on UCB, such as LUCB [Kalyanakrishnan et al., 2012], GP-UCB [Srinivas et al., 2012]. The UCB algorithm constructs the upper confident bound for each arm in every iteration, and chooses the arm with largest bound as the next arm to observe. UCB is very efficient by effectively balancing exploration and exploitation, admitting the regret upper bound $O(\sqrt{T/|L| \log T})$ [Bubeck et al., 2012], where $T$ is running time, $L$ is the set of arms. There also exist some variations of the bandit problem other than stochastic bandit problem, such as contextual bandit problem [Deshmukh et al., 2017] and bandit optimization problem [Arora et al., 2012; Hazan et al., 2016]. We recommend readers to refer to the book by [Bubeck et al., 2012]. Regret is a common metric of algorithms above, while another important metric is the sample complexity [Gabillon et al., 2012; Kalyanakrishnan et al., 2012]. Recently there are also some research to combine bandit algorithms and Monte Carlo tree methods to find out an optimal path in a tree with random leaf nodes, that corresponds to finding the optimal strategy in a game. Representative algorithms include UCT [Kocsis et al., 2006], UGapE-MCTS [Kaufmann and Koolen, 2017], and LUCB-micro [Huang et al., 2017].

Expected improvement methods The expected improvement method dates back to 1970s, when J. Mockus and Zilinskas [1978] proposed to use the expected improvement function to decide which arm to choose in each iteration, that is, the arm is chosen with the maximal expected improvement. The advantage of this method is that the expected improvement can be computed analytically [J. Mockus and Zilinskas, 1978; Jones et al., 1998]. Recently, Snoek et al. [2012] extended the idea of expected improvement to the time sensitive case by evaluating the expected improvement per second to make selection. We also adopt this concept in this paper, namely, Elrate. There are also some works on analyzing the asymptotically convergence of EI method [Ryzhov, 2016] analyzed the hit number of each non-optimal arm and the work by [Bull, 2011] provides the lower and upper bound of instantaneous regret when values of all arms are in Reproducing-Kernel Hilbert Space. Expected improvement methods have many application scenarios, see [Malkomes et al., 2016].

GP-UCB GP-UCB is a type of approaches considering the correlation among all arms, while the standard UCB [Auer et al., 2002] does not consider the correlation. GP-UCB chooses the arm with the largest UCB value in each iteration where the UCB value uses the correlation information. The proven regret achieves the rate $O(\sqrt{T \log T \log (|L| T)})$ where $T$ is running time, $L$ is the set of arms, and $\gamma_T$ is the maximum information gain at time $T$. Some variants of GP-UCB include Branch and Bound algorithm [de Freitas et al., 2012], EGP algorithm [Rana et al., 2017], distributed...
batch GP-UCB [Daxberger and Low 2017], MF-GP-UCB [Kandasamy et al. 2016], BOCA [Kandasamy et al. 2017], GP-(UCB/EST)-DPP-(MAX/SAMPLE) [Kathuria et al. 2016], to name a few.

**Parallelization bandit algorithms** To improve the efficiency of bandit algorithms, multiple agents can be employed and they can perform simultaneous investigation. Zhong et al. [2017] designed an asynchronous parallel bandit algorithm that allows multiple agents working in parallel without waiting for each other. Both theoretical analysis and empirical studies validate that the nearly linear speedup can be achieved. Kandasamy et al. [2018] designed an asynchronous version of Thompson sampling algorithm to solve parallelization Bayesian bandit optimization problem. While they consider the single user scenario, our work extends the setup to the multi-user case, which leads to our new notation to reflect the global happiness and needs some new technique in theoretical analysis.

**AutoML** Closely related to this work is the emerging trend of AutoML system and services. Model selection and hyperparameter tuning is the key technique behind such services. Prominent systems include Spark TuPAQ [Sparks et al.], Auto-WEKA [Kotthoff et al.], Google Vizier [Golovin et al.], Spearmint [Snoek et al. 2012], GPyOpt [GPyOpt 2016], and Auto scikit-learn [Feurer et al.] and prominent online services include Google Cloud AutoML and Microsoft Cognitive Services. Most of these systems focus on a single-tenant setting. Recently, Li et al. [2018] describes a system for what the authors call “multi-tenant model selection”. Our paper is motivated by this multi-tenant setting, however, we focus on a much more realistic choice of algorithm, GP-EI, that is actually used by real-world AutoML services.

### 3 MATHEMATICAL PROBLEM STATEMENT

In this section, we give the mathematical expression of the problem. We first introduce the multi-device, multi-tenant (MDMT) AutoML problem, which is a more general scenario of single-device, single-tenant (SDST) AutoML problem. Then in Section 3.1, we propose a new notion – time sensitive hierarchical bandit (TSHB) problem to abstract the MDMT AutoML problem. At last in Section 3.2, we define a new metric to quantify the goal.

**Single-device, single-tenant (SDST) AutoML** AutoML often refers to the single-device, single-tenant scenario, in which a single user aims finding the best model (or hyper parameter) for his or her individual dataset as soon as possible. Here the device is an abstract concept, it can refer to a server, or a CPU, GPU. The problem is usually formulated into a Bayesian optimization problem or a bandit problem using Gaussian process to characterize the connection among different models (or hyper parameters).

**Multi-device, multi-tenant (MDMT) AutoML** The MDMT AutoML considers the scenario where there are multiple devices available and multiple tenants who seek the best model for each individual dataset (one tenant corresponds to one dataset). While the objective of the SDST AutoML is purely from the perspective of a customer, the objective of MDMT AutoML is from the service provider; because it is generally impossible to optimize performance for each single one, given the limited computing resource. There are two fundamental challenges: 1) When there are multiple devices are available, how to coordinate all computing resources to maximize the efficiency? Simply extending the algorithms in SDST AutoML often let multiple devices run the same model on the same dataset, which apparently wastes the computing resource; 2) To serve multiple customer, we need to find a global metric to guide us to specify the most appropriate customer to serve besides of choosing the most promising the model for his / her dataset for each time. The overall problem is how to utilize all devices to achieve a certain global happiness. Each device is considered to be atomic, that is, each device can only run one algorithm (model) on one dataset at the same time.

### 3.1 Time sensitive hierarchical bandit (TSHB) problem

To find a systematic solution to the MDMT AutoML problem, we develop a new notion – time sensitive hierarchical bandit (TSHB) problem to formulate the MDMT AutoML problem.

**Definition of TSHB problem** Now we formally propose the **time sensitive hierarchical bandit problem** to abstract the multi-device, multi-tenant AutoML framework. Suppose that there are \( N \) users (or datasets in autoML framework) and \( M \) devices. Each user has a candidate set of models (or algorithms in autoML framework) he or she is interested, specifically, \( \mathcal{L}_i \) is the candidate model set for user \( i \in [N] \). Here we consider a more general situation, that is we do not assume that \( \mathcal{L}_i \) and \( \mathcal{L}_j \) are disjoint for \( i, j \in [N] \). Denote the set of all models by \( \mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \cdots \cup \mathcal{L}_N \). Running a model \( x \in \mathcal{L} \) on a device will take \( c(x) \) units of time. One model can only be run on one device at the same time and one device can only run one model at the same time. Since one model has been assigned to an idle device, \( c(x) \) units time later, the performance of
model $x$ will be observed, denoted by $z(x)$. For example, the performance could be the accuracy of the model. W.L.O.G., we assume that the larger the value of $z(x)$, the better. Roughly saying, the overall goal is to utilize $M$ devices to help $N$ users to find out each individual optimal model from the corresponding candidate set as soon as possible. More technically, the goal (from the perspective of service provider) is to maximize the cumulative global happiness over all users. We will define a regret to reflect this metric in the next subsection.

**Remark 1.** Here we simply assume $c(x)$ to be known beforehand. Although it is usually unknown beforehand, it is easy to estimate an approximate (but high accurate) value by giving the dataset set size, the computational hardware parameters, historical data, and other information. Therefore, for simplicity in analysis, we just use estimated value so that we can assume the runtime value by giving the dataset set size, the computational rate ($\text{EIrate}$ ($x$)), and any user who has this model ($z(x)$) to measure the quality of each model in the set of models that have not yet been selected before (selected models include the ones whose performance have been observed or that are under test currently). The $\text{EIrate}$ value for model $x$ depends on two factors: the running time of model $x$ and its expected improvement (EI) value (defined as Expected Improvement Function in Section 4.1).

$$\text{EIrate}(x) := \text{EI}(x)/c(x).$$

This measures the averaged expected improvement. This concept also appeared in Snoek et al. [2012].

### 3.2 Regret definition for cumulative global happiness

To quantify the goal – cumulative global happiness, we define the corresponding regret. Let us first introduce some more notations and definitions:

- $L(t)$: the set of models whose performances have been observed up to time $t$;
- $x_i^*$: the best model for user $i$, that is, $x_i^* := \underset{x \in \mathcal{L}_i}{\arg\max} z(x)$;
- $x_i^*(t)$: the best model for user $i$ observed up to time $t$, that is, $x_i^*(t) := \underset{x \in L(t) \cap \mathcal{L}_i}{\arg\max} z(x)$.

We define the individual regret (or negative individual happiness) of user $i$ at time $t$ by the gap between the currently best and the optimal, i.e., $z(x_i^*) - z(x_i^*(t))$.

In most AutoML systems, the user experience goes beyond the regret at a single time point for a single user. Instead, the regret is defined by the integration over time and the sum over all users’ regrets. It is worth noting that the regret for each user is not the same as the one in the SDST scenario since even a user is not served currently, he or she still receives the penalty (measured by the gap between the optimal model’s performance and the currently best performance). More formally, the regret at time $T$ is defined by

$$\text{Regret}_T = \sum_{i=1}^{N} \int_0^T (z(x_i^*) - z(x_i^*(t))) \, dt.$$
we have
\[ \text{EI}_t(x) = E\left[ \max \{ z(x) - z(x^*_i(t)) \} \right]. \]  
(3)

Here E means taking expectation of posterior distribution of \( z(x) \) at time \( t \).

Then, we sum this value over all users who have model \( x \) to represent the total expected improvement \( \text{EI}_t(x) \) if model \( x \) is observed. Formally, we have
\[ \text{EI}_t(x) = \sum_{i=1}^{N} I(x \in \mathcal{L}_i)\text{EI}_i(x), \]  
(4)

where \( I(A) = 1 \) if \( A \) happens, and \( I(A) = 0 \) if \( A \) does not happen. At last, we define the \( \text{EI} \) rate value of \( x \) at time \( t \) as follows:
\[ \text{EI} \text{Rate}_t(x) = \frac{\text{EI}_t(x)}{c(x)}. \]  
(5)

Now, we can choose the model with the max value of \( \text{EI} \) as the next one to run at time \( t \):
\[ x_{\text{next}} \text{ to run at time } t = \arg \max_{x \in \mathcal{L} \setminus \mathcal{L}(t)} \text{EI} \text{Rate}_t(x). \]  
(6)

Algorithm 1 MM-GP-EI Algorithm

**Input:** \( \mu(x), k(x, x'), c(x), \mathcal{L}, \{\mathcal{L}_i\}_{i=1}^{N} \) and the total time budget \( T \).

1: \( x_{\text{initial}} = \arg \max_{x \in \mathcal{L}} \mu(x), \forall i \in \mathbb{N} \).
2: \( \mathcal{L}_{ob} = \{x_{\text{initial}}\}_{i=1}^{N} \)
3: while there is a device available and the elapsed time \( t \) is less than \( T \) do
4: Refresh \( \mathcal{L}_{ob} \) to include all observed models at present
5: Update posterior mean \( \mu(\cdot) \), posterior covariance \( k(\cdot, \cdot) \) of \( z(x) \) given \( \{z(x)\}_{x \in \mathcal{L}_{ob}} \)
6: Update \( x^*_i(t) = \arg \max_{x \in \mathcal{L}_{ob} \cap \mathcal{L}_i} z(x), \forall i \in \mathbb{N} \)
7: \( \text{EI}(x) = \sum_{i=1}^{N} \sum_{x \in \mathcal{L}_i \setminus \mathcal{L}_{ob}} \sigma_i(x) \tau \left( \frac{\mu(x) - z(x^*_i(t))}{\sigma_i(x)} \right), \forall x \in \mathcal{L} \)
8: Run \( x_{\text{next}} = \arg \max_{x \in \mathcal{L} \setminus \mathcal{L}_{ob}} \text{EI}_t(x) \) on this free device
9: end while

**Output:** \( x^*_1(T), x^*_2(T), \ldots, x^*_N(T) \).

4.2 Choosing Prior: Gaussian Process

Next, we must choose a suitable prior of \( z(x) \) to estimate \( \text{EI} \) function in (4). Here, we choose Gaussian Process (GP) as the prior like many other Bayesian optimization algorithms [Bull [2011], Srinivas et al. [2012]], mainly because of its convenience of computing posterior distribution and \( \text{EI} \) function.

A Gaussian Process \( GP(\mu(x), k(x, x')) \) is determined by its mean function \( \mu(x) \) and covariance function \( k(x, x') \). If \( z(x) \) has a GP prior \( GP(\mu(x), k(x, x')) \), then after observing models in \( \mathcal{L}(t) \) at time \( t \), the posterior distribution of \( z(x) \) given \( \{z(x)\}_{x \in \mathcal{L}(t)} \) is also a Gaussian Process \( GP(\mu(x), k(x, x')) \). Here, posterior mean \( \mu(x) \) and variance \( k(x, x') \) can be computed analytically. We give the formulas in Supplemental Materials (Section A).

\( \text{EI} \) function can also be computed analytically if \( z(x) \) obeys Gaussian Process (whatever prior or posterior). The following lemma gives the expression.

**Lemma 1.** Let \( \Phi(x) \) denote cumulative distribution function (CDF) of standard normal distribution and \( \phi(x) \) denote probability density function (PDF) of standard normal distribution. Also, let \( \tau(x) = x\Phi(x) + \phi(x) \).

Then, if \( X \sim N(\mu, \sigma^2) \), and \( a \in \mathbb{R} \) is a constant, we have
\[ \mathbb{E}\left[ \max \{X - a, 0\} \right] = \sigma \tau \left( \frac{\mu - a}{\sigma} \right). \]

This section ends by the detailed description of the proposed MM-GP-EI algorithm in Algorithm 1.

**Discussion: How to Choose Prior Mean \( \mu(x) \) and Prior Covariance \( k(x, x') \)**

Prior mean \( \mu(x) \) and prior covariance \( k(x, x') \) are chosen according to the specific problem. They often characterize some properties of models in the problem, such as expected value of models and correlations among different models. In our multi-device multi-tenant example, the parameters of Gaussian process can be obtained from historical experiences and the correlation depends on two factors: the similarity of algorithms and the similarity of users’ datasets. We following standard AutoML practice (used in Google Vizier or ease.ml) to construct the kernel matrix from historical runs.

5 MAIN RESULT

Before we introduce the main theoretical result, let us propose a new notation Maximum Incremental Uncertainty (MIU), which plays a key role in our theory.

5.1 Maximum Incremental Uncertainty

Suppose that \( K \) is the kernel matrix, that is, \( K := [k(x, x')]_{x, x' \in \mathcal{L}} \), where \( k(x, x') \) is the kernel function and \( \mathcal{L} \) is the set of all models as defined in Section 3.1. So, \( K \) is a \( |\mathcal{L}| \times |\mathcal{L}| \) positive semi-definite (covariance) matrix. Suppose \( S \) is a subset of \( |\mathcal{L}| := \{1, 2, \ldots, |\mathcal{L}|\} \). Let \( K_S \) be a submatrix of \( K \) with columns and rows indexed by \( S \).

We define the s-MIU score of matrix \( K \) (\( 1 \leq s \leq |\mathcal{L}| \)) by
Let us understand the meaning of the notation MIU. Given $|\mathcal{L}|$ Gaussian random variables with covariance matrix $K \in \mathbb{R}^{|\mathcal{L}| \times |\mathcal{L}|}$, $\det(K_S')$ denotes the total quantity of uncertainty for all random variables in $S' \subseteq \mathcal{L}$. $\det(K_S')/\det(K_S)$ denotes the incremental quantity of uncertainty by adding one more random variable into $S'$ to form $S$. If the added random variable can be linearly represented by random variables in $S'$, the incremental uncertainty is zero. If the added random variable is independent to all variables in $S$, the incremental uncertainty is the variance of the added variable. Therefore, $\text{MIU}_s(K)$ measures the largest incremental quantity of uncertainty from $s-1$ random variables to $s$ random variables in $\mathcal{L}$.

Remark 2. Why do not use Information Gain? People who are familiar with the concept of information gain (IG) may ask “IG is a commonly used metric to measure how much uncertainty reduced after observed a sample. Why not use it here?” Although IG and MIU essentially follow the same spirit, the IG metric is not suitable in our setup. Based on the mathematical definition of IG (see Lemma 5.3 in Srinivas et al. [2012]), it requires the observation noise of the sample to be nonzero to make it valid (otherwise it is infinity), which makes it inappropriate in the main target scenario in this paper. In our motivating example – cloud computing platform, the observation noise is usually considered to be zero, since no people run the same experiment twice. That motivates us to define a slightly different metric, (i.e., Maximum Incremental Uncertainty), to fix the non-observation-noise issue.

5.2 Main Theorem
To simplify the analysis and result, we make the following assumption commonly used for analyzing EI [Bull 2011] and GP-UCB [Srinivas et al. 2012].

Assumption 1. Assume that

- there exists a constant $R$ such that: $|z(x) - \mu_t(x)| \leq \sigma_t(x)R$, for any model $x \in \mathcal{L}$ and any $t \geq 0$;
- $\sigma(x) \leq 1$.

Now we are ready to provide the upper bound for the regret defined in $\text{Regret}_T$.

Theorem 2. Let $\text{MIU}(T, K) := \sum_{s=2}^{\lfloor \mathcal{L}(t) \rfloor} \text{MIU}_s(K)$. Under Assumption 1, the regret of the output of Algorithm 1 up to time $T$ admits the following upper bound

$$\text{Regret}_T \lesssim (\text{MIU}(T, K) + M) \frac{N^2}{M} \bar{c},$$

where $\bar{c} := \frac{1}{N} \sum_{i=1}^{N} c(x_i^*)$ is the average time cost of all optimal models, and $\lesssim$ means “less than equal to” up to a constant multiplier.

The proof of Theorem 2 can be found in the Supplemental Materials. To the best of our knowledge, this is the first bound for time sensitive regret. We offer the following general observations:

- (convergence to optimum) If the growth of $\text{MIU}(T, K)$ with respect to $T$ is $o(T)$, then average regret converges to zero, that is,

$$\frac{1}{T} \text{Regret}_T \to 0.$$

In other words, the service provider will find the optimal model for each individual user.

- (nearly linear speedup) When more and more devices are employed, that is, increasing $M$, then the regret will decrease roughly by a factor $M$ as long as $M$ is dominated by $\text{MIU}(T, K)$.

Convergence Rate of the Average Regret. We consider the scenario where $M \ll \text{MIU}(T, K)$ and $\mathcal{L}(t)$ increases linearly with respect to $T$. Then the growth of $\text{MIU}(T, K)$ will dominate the convergence rate of the average regret. Note that $\text{MIU}(T, K)$ is bounded by

$$\text{MIU}(T, K) \leq \sum_{i \in \text{top } |\mathcal{L}(t)|} \sqrt{\det(K(i, i))}.$$

Discussion: Special Cases. Consider the following special cases:

- $O(1/T)$ rate) The convergence rate for $\frac{1}{T} \text{Regret}_T$ achieves $O(1/T)$, if $\text{MIU}(T, K)$ is bounded, for example, all random variables (models) are linearly combination of a finite number of hidden Gaussian random variables.

- (not converge) If all models are independent, then $K$ is a diagonal matrix, and $\text{MIU}_s(K)$ is a constant, which means $\text{MIU}(T, K)$ is linearly increased of $T$. In such a case, the regret is of order $T$, which implies no convergence for the average regret. This is plausible in that the algorithm gains no information from previous observations to decide next because of independence.

- $O(1/T^{1-\alpha})$ rate with $\alpha \in (0, 1))$ This rate can be achieved if $\text{MIU}(T, K)$ grows with the rate $O(T^\alpha)$.\n
\[\text{MIU}_s(K) := \max_{S' \subseteq \mathcal{L}, \sum_{s' \neq s} \delta_{s'} = s-1} \left\{ \begin{array}{ll} \sqrt{\det(K_{S'})/\det(K_{S})}, & \text{if } \det(K_{S'}) \neq 0; \\ 0, & \text{otherwise,} \end{array} \right. \]
6 EXPERIMENTS

We validate the effectiveness of the multi-device, multi-tenant GP-EI algorithm.

6.1 Data Sets and Protocol

We use two datasets for our experiments, namely (1) DeepLearning and (2) Azure. Both datasets are from the ease.ml paper [Li et al., 2018] in which the authors evaluate their single-device, multi-tenant GP-UCB algorithm. The DeepLearning dataset is collected from 22 users, each runs an image classification task. The system needs to select from 8 deep learning models, including NIN, GoogLeNet, ResNet-50, AlexNet, BNAlexNet, ResNet-18, VGG-16, and SqueezeNet. The Azure dataset is collected from 17 users, each runs a Kaggle competition. The system needs to select from 8 binary classifiers, including Averaged Perceptron, Bayes Point Machine, Boosted De-
We validate the hypothesis that, given a single device, we test different model selection algorithms using the expected improvement, the global happiness of all users can increase faster than treating all users equally. On the other hand, for DeepLearning has more users than Azure (14 vs. 9), we see that the speedup is larger on DeepLearning. The significant speedup of reaching instantaneous regret of 0.03 for Azure is most probably due to the small number of users compared to the number of devices (9 vs. 8).

6.3 Multiple device experiments

We validate the hypothesis that using multiple devices speeds up our multi-device, multi-tenant model selection algorithm. Figure 3 shows the result of using multiple devices for our algorithm. We see that the more devices we use, the faster the instantaneous regret drops. In terms of speedup, since DeepLearning has been sampled.

We now compare our approach against GP-EI-Round-Robin and GP-EI-Random when there are multiple devices available. Figure 4 shows the result. We see that, up to 4 devices (9 users in total), our approach outperforms round robin significantly on Azure. When we use 8 devices for Azure, because there are only 9 users, both our approach and round robin achieve almost the same performance.

We also conduct an experiment using a synthetic dataset with 50 users and 50 models. We model the performance as a Gaussian Process and generate random samples independently for each user. The Gaussian Process has zero mean and a covariance matrix derived from the Matérn kernel with $\nu = 5/2$. Each generated sample is upwards in order to be non-negative. We run our approach on the same dataset while varying the number of devices. For each device count we repeat the experiment 5 times. To quantify speed gains we measure the average time it takes the instantaneous regret to hit a cutoff point of 0.01. We can observe that adding more devices makes the convergence time drop at a near-linear rate.

7 CONCLUSION

In this paper, we introduced a novel multi-device, multi-tenant algorithm using GP-EI to maximize the “global happiness” for all users, who share the same set of computing resources. We formulated the “global happiness” in terms of a cumulative regret and first time provided a theoretical upper bound for the time sensitive regret in the GP-EI framework. We evaluated our algorithm on two real-world datasets, which significantly outperforms the standard GP-EI serving users randomly or in a round robin fashion. Both our theoretical results and experiments show that our algorithm can provide near linear speedups when multiple devices are available.
Acknowledgements

Chen Yu and Ji Liu are in part supported by NSF CCF1718513, IBM faculty award, and NEC fellowship. Ce Zhang and the DS3Lab gratefully acknowledge the support from Mercedes-Benz Research & Development North America, MeteoSwiss, Oracle Labs, Swiss Data Science Center, Swisscom, Zurich Insurance, Chinese Scholarship Council, and the Department of Computer Science at ETH Zurich.

References


Supplemental Materials

A Gaussian Process and Posterior Formulas

In this section we give the posterior formulas of Gaussian Process. Most formulas in this section come from Srinivas et al. [2012], but we modify them to fit our settings.

In Section 4.2, we choose the Gaussian Process \( GP(\mu(x), k(x, x')) \) as the prior of \( z(x) \), and point out that at any time \( t \), the posterior of \( z(x) \) given \( \{z(x)\}_{x \in \mathcal{L}} \) is also a Gaussian Process \( GP(\mu_t(x), k_t(x, x')) \). Now we give the formulas to compute \( \mu_t(x) \) and \( k_t(x, x') \).

Suppose at time \( t \), the observed models are \( x^{(1)}, x^{(2)}, \cdots, x^{(\mathcal{L}(t))} \), then we have the following:

\[
\mu_t(x) = v_t(x)^T K_t^{-1} (z_t - w_t) + \mu(x),
\]

\[
k_t(x, x') = k(x, x') - v_t(x)^T K_t^{-1} v_t(x'),
\]

where

\[
v_t(x) : = \left[k(x^{(1)}, x), k(x^{(2)}, x), \cdots, k(x^{(\mathcal{L}(t))}, x)\right]^T,
\]

\[
K_t : = \left[k(x^{(i)}, x^{(j)})\right]_{i, j \in \{1, 2, \cdots, \mathcal{L}(t)\}},
\]

\[
z_t : = \left[z^{(1)}, z^{(2)}, \cdots, z^{(\mathcal{L}(t))}\right],
\]

\[
w_t : = \left[\mu(x^{(1)}), \mu(x^{(2)}), \cdots, \mu(x^{(\mathcal{L}(t))})\right].
\]

In this supplemental material, we give the proof of our main theorem. Before that, let us show Lemma 1.

B Proof of Lemma 1

Proof. First, because \( \tau'(x) = \Phi(x) \) and \( \tau(-\infty) = 0 \), we have \( \tau(x) = \int_{-\infty}^x \Phi(t)dt \). Define \( Y = \frac{X - \mu}{\sigma} \), then \( X = \sigma Y + \mu \) and \( Y \sim \mathcal{N}(0, 1) \). Let \( S(x) = 1 - \Phi(x) \), for any \( b \in \mathbb{R} \), we have

\[
E\left[\max\{Y - b, 0\}\right] = \int_b^{+\infty} (y - b) \phi(y) dy
\]

\[
= \int_b^{+\infty} (y - b) d\left(-S(y)\right)
\]

\[
= -(y - b)S(y)|_b^{+\infty} + \int_b^{+\infty} S(y) dy (y - b)
\]

\[
= \int_b^{+\infty} S(y) dy = \int_b^{+\infty} \left(1 - \Phi(y)\right) dy = \int_b^{+\infty} \Phi(-y) dy
\]

\[
= \int_{-\infty}^{-b} \Phi(y) dy
\]

\[
= \tau(-b).
\]

Applying this into \( X \), we have

\[
E\left[\max\{X - a, 0\}\right] = E\left[\max\{\sigma Y + \mu - a, 0\}\right]
\]

\[
= \sigma E\left[\max\{Y - \frac{a - \mu}{\sigma}, 0\}\right]
\]

\[
= \sigma \tau\left(\frac{\mu - a}{\sigma}\right).
\]
C Proof of Theorem 2

We will prove the main theorem by the following steps:

- We will give a bound for EI in Lemma 3.
- Using the bound of EI, we will bound Regret by the sum of variance of posterior distribution. This will be presented in Lemma 4.
- Using Lemma 5, we will convert the sum of variance into MIU, and then prove the main theorem.

Let \( \hat{t}(x) \) denote the time when model \( x \) began to be tested. Since every model will just be tested once, this is well-defined. Define \( \hat{\sigma}(x) = \sigma_{\hat{t}(x)}(x) \). And let \( \hat{x}^{(1)}_{\text{test}}, \hat{x}^{(2)}_{\text{test}}, \cdots, \hat{x}^{(p(T))}_{\text{test}} \) denote all models that have been tested or are being tested up to time \( T \) by the order of their start-testing time.

It follows from Assumption 1 that we have \( |z(x) - \mu_{0}(x)| \leq \sigma_{0}(x)R \). So, the reward of all models has a universal bound \( C_R \), which means \( |z(x)| \leq C_R \) for every model \( x \).

**Lemma 3.** Suppose Assumption holds. For each user \( i \), let \( x \) be a model belonging to user \( i \). Let \( (z(x) - z(x^\ast(t)))^{+} = \max\{z(x) - z(x^\ast(t)), 0\} \). Then for every \( t \geq 0 \), we have

\[
\frac{\tau(-R)}{\tau(R)} \left( z(x) - z(x^\ast(t)) \right)^{+} \leq EI_{i,t}(x) \leq \left( z(x) - z(x^\ast(t)) \right)^{+} + (R + 1)\sigma_{t}(x).
\]

**Proof.** If \( \sigma_{t}(x) = 0 \), then \( |z(x) - \mu_{t}(x)| \leq 0 \), which means \( z(x) = \mu_{t}(x) \) is a constant for fixed \( x \) and \( t \). Then from (3), we have \( EI_{i,t}(x) = \left( z(x) - z(x^\ast(t)) \right)^{+} \), and the result is trivial.

Suppose \( \sigma_{t}(x) > 0 \). From (3) and Lemma 1, we have \( EI_{i,t}(x) = \sigma_{t}(x)\tau\left( \frac{\mu_{t}(x) - z(x^\ast(t))}{\sigma_{t}(x)} \right) \). Also, since \( |z(x) - \mu_{t}(x)| \leq R\sigma_{t}(x) \), we have

\[
\left| \frac{\mu_{t}(x) - z(x^\ast(t))}{\sigma_{t}(x)} - \frac{z(x) - z(x^\ast(t))}{\sigma_{t}(x)} \right| \leq R,
\]

which implies

\[
\frac{\mu_{t}(x) - z(x^\ast(t))}{\sigma_{t}(x)} \leq \frac{z(x) - z(x^\ast(t))}{\sigma_{t}(x)} + R \leq \left( \frac{z(x) - z(x^\ast(t))}{\sigma_{t}(x)} \right)^{+} + R.
\]

Also, since \( \tau'(y) = \Phi(y) \in [0, 1] \), \( \tau \) is non-decreasing, and \( \tau(y) \leq 1 + y \) for \( y \geq 0 \). Therefore,

\[
EI_{i,t}(x) = \sigma_{t}(x)\tau\left( \frac{\mu_{t}(x) - z(x^\ast(t))}{\sigma_{t}(x)} \right) \\
\leq \sigma_{t}(x)\tau\left( \frac{z(x) - z(x^\ast(t))}{\sigma_{t}(x)} + R \right) \\
\leq \sigma_{t}(x)\left( \frac{z(x) - z(x^\ast(t))}{\sigma_{t}(x)} + R + 1 \right) \\
= \left( z(x) - z(x^\ast(t)) \right)^{+} + (R + 1)\sigma_{t}(x),
\]

and the upper bound is proved for the case that \( \sigma_{t}(x) > 0 \).

If \( (z(x) - z(x^\ast(t)))^{+} = 0 \), then the left side of the inequality is 0. From (3), we have \( EI_{i,t}(x) \geq 0 \). Therefore, the lower bound is trivial.
Now suppose \((z(x) - \hat{z}(x_i^*(t)))^+ > 0\), which means \((z(x) - \hat{z}(x_i^*(t)))^+ = z(x) - \hat{z}(x_i^*(t))\).

From (8), we have \(\frac{\mu_t(x) - \hat{z}(x_i^*(t))}{\sigma_t(x)} \geq \frac{z(x) - \hat{z}(x_i^*(t))}{\sigma_t(x)} - R\). Also, we have \(\tau(y) = y + \tau(-y) \geq y\). Thus

\[
\text{EI}_{i,t}(x) = \sigma_t(x) \tau \left( \frac{\mu_t(x) - \hat{z}(x_i^*(t))}{\sigma_t(x)} \right)
\]

\[
\geq \sigma_t(x) \tau \left( \frac{z(x) - \hat{z}(x_i^*(t))}{\sigma_t(x)} - R \right)
\]

\[
\geq \sigma_t(x) \left( \frac{z(x) - \hat{z}(x_i^*(t))}{\sigma_t(x)} - R \right)
\]

\[
= z(x) - \hat{z}(x_i^*(t)) - R \sigma_t(x).
\]

If \(z(x) - \hat{z}(x_i^*(t)) - R \sigma_t(x) \geq \frac{\tau(-R)}{\tau(R)} \left( z(x) - \hat{z}(x_i^*(t)) \right)\), we conclude the proof. Otherwise, we have

\[
R \sigma_t(x) > \left( 1 - \frac{\tau(-R)}{\tau(R)} \right) \left( z(x) - \hat{z}(x_i^*(t)) \right)
\]

\[
= \frac{\tau(R) - \tau(-R)}{\tau(R)} \left( z(x) - \hat{z}(x_i^*(t)) \right)
\]

\[
= \frac{R}{\tau(R)} \left( z(x) - \hat{z}(x_i^*(t)) \right),
\]

which implies \(1 > \frac{z(x) - \hat{z}(x_i^*(t))}{\sigma_t(x)}\). Also, it follows from (8) and the assumption \(z(x) - \hat{z}(x_i^*(t)) > 0\) that

\[
\frac{\mu_t(x) - \hat{z}(x_i^*(t))}{\sigma_t(x)} + R \geq \frac{z(x) - \hat{z}(x_i^*(t))}{\sigma_t(x)} > 0,
\]

and thus \(\frac{\mu_t(x) - \hat{z}(x_i^*(t))}{\sigma_t(x)} \geq -R\). So

\[
\text{EI}_{i,t}(x) = \sigma_t(x) \tau \left( \frac{\mu_t(x) - \hat{z}(x_i^*(t))}{\sigma_t(x)} \right)
\]

\[
\geq \sigma_t(x) \tau(-R) \geq \sigma_t(x) \tau(-R) \frac{z(x) - \hat{z}(x_i^*(t))}{\sigma_t(x) \tau(R)}
\]

\[
= \frac{\tau(-R)}{\sigma_t(x) \tau(R)} \left( z(x) - \hat{z}(x_i^*(t)) \right),
\]

which also concludes the proof.

\[\square\]

**Lemma 4.** Under Assumption 1, we have

\[
\text{Regret}_T \leq \left( \frac{\tau(R)N(R+1)}{\tau(-R)M} \sum_{k=1}^{\bar{p}(\mathcal{X})} \tilde{\sigma}(x_{\tilde{c}(k)}) + C_R + \frac{\tau(R)}{\tau(-R) NC_R} \right) \sum_{i=1}^{N} c(x_i^*).
\]

**Proof.** Let \(\hat{x}_j(t)\) denote the model that device \(j\) is testing at time \(t\). For each device \(j\), define a function \(f_j(t)\) as follows:

\[
f_j(t) = \sum_{i=1}^{N} \max \left\{ z(\hat{x}_j(t)) - \hat{z}(x_i^*(\hat{x}_j(t))), 0 \right\} + (R+1)\tilde{\sigma}(\hat{x}_j(t))
\]

For each user \(i\), define a function \(g_i(t)\) as follows:

\[
g_i(t) = \left\{ \begin{array}{ll} 
0, & \text{if the actual optimal model } x_i^* \text{ is being tested} \\
\hat{z}(x_i^*(t)), & \text{otherwise}. 
\end{array} \right.
\]

\[
\]
Applying Lemmas 3 and 4, we have
\[ g_i(t) \leq \frac{\tau(R)}{\tau(-R)} c(x_i^*) f_j(t). \quad (9) \]

At time \( t \), for any device \( j \) and any user \( i \), if model \( x_i^* \) is being tested, then \( g_i(t) = 0 \). Thus the inequality (9) is obvious. If the test of model \( x_i^* \) has been finished, then \( x_i^*(t) = x_i^* \), we also have \( g_i(t) = 0 \), which implies (9).

Therefore, we only consider the situation that \( x_i^* \) has not been tested up to time \( t \) yet. Then by Algorithm 1 we have \( \text{EI}_{i,\hat{x}_j(t)}(x_i^*) \leq \text{EI}_{i,\hat{x}_j(t)}(\hat{x}_j(t)) \), or equivalently
\[ \frac{\text{EI}_{i,\hat{x}_j(t)}(x_i^*)}{c(x_i^*)} \leq \frac{\text{EI}_{i,\hat{x}_j(t)}(\hat{x}_j(t))}{c(\hat{x}_j(t))}. \]

Applying Lemmas 3 and 4, we have
\[
\frac{\tau(-R)}{\tau(R)} \left( z(x_i^*) - z(\hat{x}_j(t)) \right) \leq \frac{\text{EI}_{i,\hat{x}_j(t)}(x_i^*)}{c(x_i^*)} \leq \frac{\text{EI}_{i,\hat{x}_j(t)}(\hat{x}_j(t))}{c(\hat{x}_j(t))} = \sum_{i=1}^{N} \frac{1(\hat{x}_j(t) \in \mathcal{L}_i) \text{EI}_{i,\hat{x}_j(t)}(\hat{x}_j(t))}{c(\hat{x}_j(t))} \leq f_j(t).
\]

Noticing that \( z(x_i^*(t)) \) is a non-decreasing function of variable \( t \) and \( \hat{t}(\hat{x}_j(t)) \leq t \), we have \( z(x_i^*(\hat{t}(\hat{x}_j(t)))) \leq z(x_i^*(t)) \), then we get (9).

Recall that \( c(x_i^*) \) is the time required to finish testing the model \( x_i^* \), so \( g_i(t) \neq z(x_i^*) - z(x_i^*(t)) \) only holds on a set with measure at most \( c(x_i^*) \). From Assumption 1, \( z(x_i^*) - z(x_i^*(t)) \leq C_R \). Therefore, for any \( j = 1, 2, \cdots, M \), we have
\[
\int_0^T \left( z(x_i^*) - z(x_i^*(t)) \right) dt \leq \int_0^T g_i(t) dt + c(x_i^*) C_R \leq \frac{\tau(R)}{\tau(-R)} c(x_i^*) \int_0^T f_j(t) dt + c(x_i^*) C_R,
\]
which implies
\[
\int_0^T \left( z(x_i^*) - z(x_i^*(t)) \right) dt \leq \frac{\tau(R)}{\tau(-R)M} \sum_{j=1}^{M} \int_0^T f_j(t) dt + c(x_i^*) C_R. \quad (10)
\]

Now, we fix \( j = 1 \). In fact, \( f_1(t) \) is a step function, that is, when device 1 is testing a model, \( f_1(t) \) remains the same. Let \( x_i^{(1)}, x_i^{(2)}, x_i^{(3)}, \cdots \) denote the models tested on device \( i \) by order. Let \( x_i^{(p_1(T))} \) denote the model that is being tested on device \( i \) at time \( T \). Then we have
\[
\int_0^T f_1(t) dt = \sum_{k=1}^{p_1(T)-1} \sum_{i=1}^{N} 1(x_1^{(k)} \in \mathcal{L}_i) \left( \max \left\{ z(x_1^{(k)}) - z(x_1^{(k)}(\hat{t}(x_1^{(k)}))), 0 \right\} + (R + 1) \hat{\sigma}(x_1^{(k)}) \right) \\
+ \int_{\hat{x}_1^{(p_1(T))}}^T f_1(t) dt \\
\leq \sum_{k=1}^{p_1(T)} \sum_{i=1}^{N} 1(x_1^{(k)} \in \mathcal{L}_i) \left( \max \left\{ z(x_1^{(k)}) - z(x_1^{(k)}(\hat{t}(x_1^{(k)}))), 0 \right\} + (R + 1) \hat{\sigma}(x_1^{(k)}) \right) \\
= \sum_{i=1}^{N} \sum_{k=1}^{p_1(T)} 1(x_1^{(k)} \in \mathcal{L}_i) \max \left\{ z(x_1^{(k)}) - z(x_1^{(k)}(\hat{t}(x_1^{(k)}))), 0 \right\} + N(R + 1) \sum_{k=1}^{p_1(T)} \hat{\sigma}(x_1^{(k)}). \quad (11)
\]
Let $x_{1, i}^{(1)}, x_{1, i}^{(2)}, \ldots, x_{1, i}^{(p_{1, i}(T))}$ denote the models from user $i$ tested on device 1 by order. Then, we have

\[
\sum_{k=1}^{p_{1, i}(T)} \mathbb{I}(x_{1, i}^{(k)} \in \mathcal{L}_i) \max \left\{ z(x_{1, i}^{(k)}) - z(x_{1, i}^{*}(\hat{t}(x_{1, i}^{(k)}))), 0 \right\} = \sum_{k=1}^{p_{1, i}(T)} \max \left\{ z(x_{1, i}^{(k)}) - z(x_{1, i}^{*}(\hat{t}(x_{1, i}^{(k)}))), 0 \right\} = \sum_{k=1}^{p_{1, i}(T)} \left( \max \left\{ z(x_{1, i}^{(k)}), z(x_{1, i}^{*}(\hat{t}(x_{1, i}^{(k)}))) \right\} - z(x_{1, i}^{*}(\hat{t}(x_{1, i}^{(k)}))) \right).
\]

When $x_{1, i}^{(k+1)}$ begins to test, $x_{1, i}^{(k)}$ of course has finished testing, so

\[
z(x_{1, i}^{*}(\hat{t}(x_{1, i}^{(k+1)}))) \geq \max \left\{ z(x_{1, i}^{(k)}), z(x_{1, i}^{*}(\hat{t}(x_{1, i}^{(k)}))) \right\} \geq z(x_{1, i}^{*}(\hat{t}(x_{1, i}^{(k)}))).
\]

Therefore,

\[
\sum_{k=1}^{p_{1, i}(T)} \mathbb{I}(x_{1, i}^{(k)} \in \mathcal{L}_i) \max \left\{ z(x_{1, i}^{(k)}) - z(x_{1, i}^{*}(\hat{t}(x_{1, i}^{(k)}))), 0 \right\} \leq \max \left\{ z(x_{1, i}^{(p_{1, i}(T))}), z(x_{1, i}^{*}(\hat{t}(x_{1, i}^{(p_{1, i}(T))}))) \right\} - z(x_{1, i}^{*}(\hat{t}(x_{1, i}^{(1)}))) \leq C_R. \tag{12}
\]

Applying (12) into (11), we get

\[
\int_0^T f_1(t) dt \leq NC_R + N(R + 1) \sum_{k=1}^{p_{1, i}(T)} \hat{\sigma}(x_{1, i}^{(k)}).
\]

Similarly, for any $j \in \{1, 2, \ldots, M\}$, we also have

\[
\int_0^T f_j(t) dt \leq NC_R + N(R + 1) \sum_{k=1}^{p_{1, j}(T)} \hat{\sigma}(x_{j, i}^{(k)}).
\]

Applying these to (10), we have

\[
\int_0^T \left( z(x_{i}^{*}) - z(x_{i}^{*}(t)) \right) dt \leq \frac{\tau(R)}{\tau(-R)} c(x_{i}^{*}) NC_R + c(x_{i}^{*}) C_R + \frac{\tau(R) c(x_{i}^{*}) N(R + 1)}{\tau(-R) M} \sum_{k=1}^{\left| \mathcal{L}(t) \right|} \hat{\sigma}(x_{i}^{(k)}),
\]

where $|\mathcal{L}(t)| = \sum_{j=1}^{M} p_j(T)$.

Applying this into (2), we get the result:

\[
\text{Regret}_T \leq \left( \frac{\tau(R)}{\tau(-R)} NC_R + C_R + \frac{\tau(R) N(R + 1)}{\tau(-R) M} \sum_{k=1}^{\left| \mathcal{L}(t) \right|} \hat{\sigma}(x_{i}^{(k)}) \right) \sum_{i=1}^{N} c(x_{i}^{*}).
\]

Lemma 5. Suppose $A$ is an $n \times n$ positive definite matrix, and we divide it into 4 blocks, $A = \begin{pmatrix} A_{n-1} & B \\ B^T & a \end{pmatrix}$, here $A_{n-1}$ is an $(n-1) \times (n-1)$ matrix. Then we have

\[
\frac{\det(A)}{\det(A_{n-1})} = a - B^T A_{n-1}^{-1} B.
\]
Proof. We have the following
\[
A \begin{pmatrix}
I_{n-1} & -A^{-1}B \\
0 & 1
\end{pmatrix} = \begin{pmatrix}
A_{n-1} & B \\
B^T & a
\end{pmatrix} \begin{pmatrix}
I_{n-1} & -A^{-1}B \\
0 & 1
\end{pmatrix} = \begin{pmatrix}
A_{n-1} & 0 \\
B^T & a - B^T A^{-1} B
\end{pmatrix},
\]
here, \( I_{n-1} \) is an \((n-1) \times (n-1)\) identity matrix. Computing the determinant of both sides of the equality, we get:
\[
\det(A) = \det(A_{n-1})(a - B^T A_{n-1}^{-1} B).
\]
\[\square\]

Proof of Theorem 2. We only have to prove that
\[
\sum_{s=1}^{\left|\mathcal{L}(t)\right|} \tilde{\sigma}(x_{\text{test}}^{(s)}) \leq M + \text{MIU}(T, K).
\]
Then, combing with Lemma 4, we complete the proof.

For model \(x_{\text{test}}^{(s)} (s > M)\), when this model begins to test, there are \(s - 1\) models having finished testing or is testing. There are \(M\) devices, and \(x_{\text{test}}^{(s)}\) should occupy one device, so at least \(s - 1 - (M - 1) = s - M \geq 1\) models having finished testing. Suppose these finish-testing models are \(x_{f_1}, x_{f_2}, \ldots, x_{f_{s-M}}\).

Let \(P\) denotes the variance matrix of random variable \(z(x_{f_1}), z(x_{f_2}), \ldots, z(x_{f_{s-M}}), z(x_{\text{test}}^{(s)})\) (rows and columns are arranged by this order). Notice that \(P\) is an \((s + 1 - M) \times (s + 1 - M)\) matrix.

From (7), we know that the conditional variance is smaller than unconditional variance for a multivariable Gaussian distribution (or Gaussian process), so we have
\[
\tilde{\sigma}(x_{\text{test}}^{(s)}) = \sigma(x_{\text{test}}^{(s)} | x_{\text{test}}^{(1)}, x_{\text{test}}^{(2)}, \ldots, x_{\text{test}}^{(\hat{p}(T)-1)}) \leq \sigma(x_{\text{test}}^{(s)} | x_{f_1}, x_{f_2}, \ldots, x_{f_{s-M}}).
\]
Let’s rewrite \(P\) as \(\begin{pmatrix} P_1 & B \\ B^T & p \end{pmatrix}\), where \(P_1\) is an \((s - M) \times (s - M)\) matrix. Also from (7), we have \(\sigma(x_{\text{test}}^{(k)} | x_{f_1}, x_{f_2}, \ldots, x_{f_{s-M}}) = p - B^T P_1^{-1} B\). From Lemma 5, we have: \(p - B^T P_1^{-1} B = \frac{\det(P_1)}{\det(P)}\). From the definition of \(\text{MIU}_{s-M+1}(K)\), we have \(\frac{\det(P_1)}{\det(P)} \leq \text{MIU}_{s-M+1}(K)\). Together with (13), we obtain
\[
\tilde{\sigma}(x_{\text{test}}^{(s)}) \leq \text{MIU}_{s-M+1}(K).
\]
Particularly, because \(s > M\), we have \(s - M + 1 \geq 2\).

For model \(x_{\text{test}}^{(s)} (s \leq M)\). Again, because the conditional variance is smaller than unconditional variance for a multivariable Gaussian distribution (or Gaussian process), we have
\[
\tilde{\sigma}(x_{\text{test}}^{(s)}) \leq \sigma(x_{\text{test}}^{(s)}) \leq 1.
\]
From (14) and (15), we conclude that
\[
\sum_{s=1}^{\hat{p}(T)} \tilde{\sigma}(x_{\text{test}}^{(s)}) \leq M + \text{MIU}(T, K),
\]
which complete the proof as analyzing at the beginning of this proof. \[\square\]