IEICE TRANS. INF. & SYST., VOL.E103–D, NO.12 DECEMBER 2020

PAPER

Practical Evaluation of Online Heterogeneous Machine Learning

Kazuki SESHIMO[†], **Akira OTA**[†], **Daichi NISHIO**[†], ***Nonmembers***, ***and*** **Satoshi YAMANE**[†a)], ***Member***

SUMMARY In recent years, the use of big data has attracted more attention, and many techniques for data analysis have been proposed. Big data analysis is difficult, however, because such data varies greatly in its regularity. Heterogeneous mixture machine learning is one algorithm for analyzing such data efficiently. In this study, we propose online heterogeneous learning based on an online EM algorithm. Experiments show that this algorithm has higher learning accuracy than that of a conventional method and is practical. The online learning approach will make this algorithm useful in the field of data analysis.

key words: *machine-learning, big data, mixture model, EM algorithm*

1. Introduction

1.1 Research Background

In recent years, the diversity of networkable devices has increased, resulting in a huge variety of data. Moreover, the amount of data handled by information systems is increasing at an accelerated rate. On the other hand, the performance of software and hardware has also improved, enabling a huge variety of data to be acquired, stored, and analyzed. Analysis techniques for such data are attracting attention, leading us to benefit from predictive analytics in our lives [1].

Thus, in the real world, the basis of data collection bases has advanced. Hence, there is increasing demand for analytical methods that can run at high speed and with high accuracy.

1.2 Research Purpose

"Big data" means data that has a large size and is heterogeneous, with multiple different patterns and varying regularity. This makes heuristic analysis difficult. Heterogeneous mixture learning is one technology to solve this problem. This technology automatically decides an optimal data combination algorithmically and extracts rules accordingly. The purpose of our study is to improve this technology for application in online learning to further accelerate big data analysis.

Manuscript received January 30, 2020.
Manuscript revised June 22, 2020.
Manuscript publicized August 31, 2020.

[†]The authors are with Kanazawa University, Kanazawa-shi, 920–1192 Japan.
a) E-mail: syamane@is.t.kanazawa-u.ac.jp

DOI: 10.1587/transinf.2020EDP7020

2. Background Technologies

In this section, we describe heterogeneous mixture learning. Especially, we describe Factorized Information Criterion and Factorized Asymptotic Bayesian inference. Also we describe the basic of machine learning in Appendix.

2.1 Heterogeneous Mixture Learning

NEC developed heterogeneous mixture learning, a technology for accurately analyzing heterogeneous data.

Conventional big data analysis techniques cannot handle heterogeneous data well, thus reducing the prediction accuracy. To avoid that problem, data scientists proposed dividing data into units and extracting rules for each unit. As the amount of data increases, however, the number of data division patterns also increases.

Hence, to analyze heterogeneous data with high accuracy, it is necessary to solve three problems simultaneously. The first is the problem of dividing the data by determining the number of prediction formulas. The second is the problem of determining the explanatory variables used in each prediction formula and their combinations. The third is the problem of deciding the rules for dividing data. Because of the huge number of model candidates for mixing data, it is difficult (practically impossible) to solve these problems at the same time with conventional machine learning methods. The central prediction algorithm of heterogeneous mixture learning is a piecewise sparse linear prediction model [5]. This model learns the decision rules of a decision tree in the feature space and the contributions of the features in each region [6]–[9].

To provide background for the proposed algorithm, this section describes the factorized information criterion (FIC) and factorized asymptotic Bayesian (FAB) inference, which are important techniques for evaluating and learning models in heterogeneous mixture learning.

2.2 Factorized Information Criterion (FIC)

An information criterion is a criterion for judging whether a model obtained from data has good accuracy. The FIC is an information criterion designed for heterogeneous prediction models. We calculate the FIC by using the log-likelihood of the conditional probability $p(x^N|M)$ of the observed data z^N in the model M and the lower bound of the variational

probability q of z^N.

First, the lower bound of the variational probability of a latent variable is expressed as

$$\log p\left(\boldsymbol{x}^N|M\right) \geq \sum_{z^N} q\left(\mathbf{z}^N\right) \log\left(\frac{p\left(\boldsymbol{x}^N, \mathbf{z}^N|M\right)}{q\left(\mathbf{z}^N\right)}\right)$$

$$p\left(\boldsymbol{x}^N, \mathbf{z}^N|M\right) = \int p\left(\mathbf{z}^N|\alpha\right) \prod_{c=1}^{C} p_c\left(\boldsymbol{x}^N|\mathbf{z}_c^N, \varphi_c\right) p(\boldsymbol{\theta}|M) d\boldsymbol{\theta}$$

where C is the number of mixtures, and α is a mixing coefficient. We then apply the Laplace approximation to $p(\boldsymbol{z}^N|\alpha)$ and $p_c(\boldsymbol{x}^N|\boldsymbol{z}^N, \varphi_c)$:

$$\log p\left(\boldsymbol{x}^N, \mathbf{z}^N|\theta\right) \approx \log p\left(\boldsymbol{x}^N, \mathbf{z}^N|\overline{\theta}\right) - \frac{N}{2}\left[\overline{F}_Z, (\alpha - \overline{\alpha})\right] - \sum_{c=1}^{C} \frac{\sum_{n=1}^{N} z_{nc}}{2}\left[\overline{F}_c, (\varphi_c - \overline{\varphi}_c)\right],$$

$$\overline{F}_Z = -\frac{1}{N} \left.\frac{\partial^2 \log p\left(\mathbf{z}^N|\alpha\right)}{\partial\alpha\partial\boldsymbol{\alpha}^T}\right|_{\alpha=\overline{\alpha}}$$

$$\overline{F}_c = -\frac{1}{\sum_{n=1}^{N} z_{nc}} \left.\frac{\partial^2 \log p_c\left(\boldsymbol{x}^N|\mathbf{z}_c^N, \varphi_c\right)}{\partial\varphi_c\partial\varphi_c^T}\right|_{\varphi_c=\overline{\varphi}_c}.$$

Next, we approximate the observed and conditional probabilities of the latent variable $p(\boldsymbol{x}^N, \boldsymbol{z}^N|M)$ for the model. Here, D_α means the dimension of $D(\alpha)$:

$$\mathrm{FIC}\left(\boldsymbol{x}^N, M\right) = \max_q \left\{J\left(q, \overline{\theta}, \boldsymbol{x}^N\right)\right\}$$

$$J\left(q, \bar{\theta}, \boldsymbol{x}^N\right) = \sum_{z^N} q\left(\mathbf{z}^N\right)\left[\log p\left(\boldsymbol{x}^N, \mathbf{z}^N|\bar{\theta}\right) - \frac{D_\alpha}{2}\log N - \sum_{c=1}^{c} \frac{D_c}{2}\left\{\log\left(\sum_{n=1}^{N} z_{nc}\right) - \log q\left(z^N\right)\right\}\right].$$

When the FIC value is higher, a heterogeneous prediction model is generally more accurate [6].

2.3 Factorized Asymptotic Bayesian (FAB) Inference

FAB is an algorithm for maximizing the FIC. Because we cannot analytically determine the parameters, we cannot evaluate the FIC directly. Instead, we evaluate the FIC by asymptotically maximizing its lower bound. First, we define the FIC's lower bound, where $\log\left(\sum_{n=1}^{N} z_{nc}\right)$ is $\hat{q}$: $\log\left(\sum_{n=1}^{N} z_{nc}\right)$ is $\hat{q}$.

$$\mathrm{FIC}\left(\boldsymbol{x}^N|M\right) \geq G\left(q, \tilde{q}, \theta, x^N\right) \equiv \sum_{\mathbf{z}^N} q\left(\mathbf{z}^N\right)\left[\log p\left(\boldsymbol{x}^N, \mathbf{z}^N|\bar{\theta}\right) - \frac{D_\alpha}{2}\log N - \sum_{c=1}^{C} \frac{D_C}{2}\left\{L\left(\sum_{n=1}^{N} z_{nc}, \sum_{n=1}^{N} \tilde{q}\left(z_{nc}\right)\right) - \log q\left(z^N\right)\right\}\right].$$

We express the parameters for maximization as

$$M^*, q^*, \boldsymbol{\theta}^*, \tilde{q}^* = \arg\max_{M,q,\theta,\tilde{q}} G\left(q, \tilde{q}, \theta, x^N\right).$$

Because there is a limited number of model candidates, it is possible to select the best model after obtaining the FIC value for each candidate. The combination of the number of models and their prediction formulas reaches enormous size, however, increasing the amount of calculation. To avoid this, we use an iterative optimization algorithm.

We set the mixed number of data as C and consider combinatorial optimization problems that maximize the FIC's lower bound. We denote the model for class a in the mixed model as S_a and express the problem as the following:

$$\boldsymbol{S}^*, q^*, \boldsymbol{\theta}^*, \tilde{q}^* = \underset{\boldsymbol{S},q,\boldsymbol{\theta},\tilde{q}}{\mathrm{argmax}} G\left(q, \tilde{q}, \boldsymbol{\theta}, \boldsymbol{x}^N\right).$$

Here, we repeat the two steps t times to solve this problem.

Next, we optimize the variational probability q of the latent variable $\boldsymbol{z}^N$ as the following:

$$q^{(t)} = \underset{q}{\mathrm{argmax}} \left\{G\left(q, \tilde{q} = q^{(t-1)}, \boldsymbol{\theta}^{(t-1)}, \boldsymbol{x}^N\right)\right\}$$

$$q^{(t)}\left(z_{nc}\right) \propto \alpha_c^{(t-1)} p\left(\boldsymbol{x}_n|\varphi_c^{(t-1)}\right) \exp\left(\frac{-D_c}{2\alpha_c^{(t-1)}N}\right)$$

$$q^{(t)}\left(z_{nc}\right) = \begin{cases} 0 & : \sum_{n=1}^{N} q^{(t)}\left(z_{nc}\right) < \delta \\ q^{(t)}\left(z_{nc}\right)/\mathcal{Q}_c^{(t)} & : \sum_{n=1}^{N} q^{(t)}\left(z_{nc}\right) \geq \delta \end{cases}.$$

M Step: We optimize the component $\boldsymbol{S}$ of the mixed model and its parameter θ as follows. Then, we can optimize S_c and φ_c individually to avoid a combinatorial explosion:

$$\boldsymbol{S}^{(t)}, \boldsymbol{\theta}^{(t)} = \underset{\boldsymbol{s},\boldsymbol{\theta}}{\mathrm{argmax}} G\left(q^{(t)}, \tilde{q} = q^{(t)}, \theta, \boldsymbol{x}^N\right)$$

$$\alpha_c^{(t)} = \frac{\sum_{n=1}^{N} q^{(t)}\left(z_{nc}\right)}{N}$$

$$S_c^{(t)}, \varphi_c^{(t)} = \arg\max_{S_c\varphi_c} H_c\left(q^{(t)}, q^{(t)}, \varphi_c, x^N\right)$$

$$H_c\left(q^{(t)}, q^{(t)}, \varphi_c, \boldsymbol{x}^N\right) = \sum_{n=1}^{N} q^{(t)}\left(z_{nc}\right) \log p\left(\boldsymbol{x}_n|\varphi_c\right) - \frac{D_c}{2} L\left(\sum_{n=1}^{N} q^{(t)}\left(z_{nc}\right), \sum_{n=1}^{N} q^{(t)}\left(z_{nc}\right)\right).$$

Here, $H_c\left(q^{(t)}, q^{(t)}, \varphi_c, x^N\right)$ is related to the c-th model, where the number of data divisions is C. Because the number of candidates for division is finite, we optimize φ_c for that of S_c and seek the best one by comparing them.

In these two steps, the following inequality holds.

$$FIC_{LB}^{(t)}\left(\boldsymbol{x}^N, M\right) \geq FIC_{LB}^{(t-1)}\left(x^N, M\right) \equiv G\left(q^{(t)}, q^{(t)}, \theta^{(t)}, \boldsymbol{x}^N\right).$$

We use this to define the following end condition ϵ for iteration of the two steps:

$$FIC_{LB}^{(t)}\left(x^N, M\right) - FIC_{LB}^{(t-1)}\left(x^N, M\right) \leq \varepsilon.$$

Finally, we list the batch heterogeneous mixture learning algorithm for the GMM below.

Algorithm 1 Algorithm FAB for Mixture Models

Require: $\boldsymbol{x}^N, C_{\max}, \boldsymbol{S}, \varepsilon, \delta$
Ensure: $C^*, S^*, \boldsymbol{\theta}^*, q^*\left(\mathbf{z}^N\right), FIC_{LB}^*$
$FIC_{LB}^* = -\infty$
$t = 0, FIC_{LB}^{(0)} = -\infty, \delta = 0, T_c = 0$
$q^{(0)}\left(\mathbf{z}^N\right)$Random initialization
while $FIC_{LB}^{(t)}\left(x^N, M\right) - FIC_{LB}^{(t-1)}\left(x^N, M\right) \leq \varepsilon$ **do**
for $c = 1, \cdots, C_{max}$ **do**
//V Step:$q^{(t)}$calculate
$q^{(t)}\left(z_{nc}\right) \propto \alpha_c^{(t-1)} p\left(\boldsymbol{x}_n|\varphi_c^{(t-1)}\right) \exp\left(\frac{-D_c}{2\alpha_c^{(t-1)}N}\right)$
$q^{(t)}\left(z_{nc}\right) = \begin{cases} 0 & : \sum_{n=1}^N q^{(t)}\left(z_{nc}\right) < \delta \\ q^{(t)}\left(z_{nc}\right)/Q_c^{(t)} & : \sum_{n=1}^N q^{(t)}\left(z_{nc}\right) \geq \delta \end{cases}$
$FIC_{LB}^{(t)}\left(x^N, M\right) = \sum_{z^N} q\left(\mathbf{z}^N\right)\left[\log p\left(\boldsymbol{x}^N, \mathbf{z}^N|\overline{\theta}\right)\right.$
$\left. - \frac{D_\alpha}{2}\log N - \sum_{c=1}^c \frac{D_c}{2}\left\{\log\left(\sum_{n=1}^N z_{nc}\right) - \log q\left(z^N\right)\right\}\right]$
//M Step : $\boldsymbol{S}^{(t)}, \boldsymbol{\theta}^{(t)}$calculate
$\alpha_C^{(t)} = \frac{\sum_{n=1}^N q^{(t)}(z_{nc})}{N}$
$S_c^{(t)}, \varphi_c^{(t)} = \arg\max_{S_c, \varphi_c}\left(q^{(t)}, q^{(t)}, \varphi_c, x^N\right)$
$t = t + 1$
end for
$T_c = t, FIC_{LB}^{(T_c)}, \boldsymbol{S}^{(T_c)}, \boldsymbol{\theta}^{(T_c)}, q^{(T_c)}\left(\mathbf{z}^N\right)$
end while
$//M^* = (C^*, S^*), \boldsymbol{\theta}^*, q^*\left(\mathbf{z}^N\right)$choose
$M^*, q^*, \boldsymbol{\theta}^*, \tilde{q}^* = \arg\max_{M,q,\theta,\tilde{q}} G\left(q, \tilde{q}, \theta, x^N\right)$

3. Related Works

This section introduces important works on the technologies composing heterogeneous mixture learning. First, in 2012, Fujimaki proposed FAB, a new variational Bayesian method for mixture models [6]. Previously, data scientists proposed dividing data into units and extracting rules for each unit. The purpose of our work is to automate this task and quickly derive an accurate prediction model. We use the FIC, a new criterion with marginal likelihood for mixed normal distributions, and FAB, an algorithm to maximize this. Because FAB has a reduction mechanism, it is highly regarded for its model selectivity and computational efficiency.

Also in 2012, an extension of FAB for hidden Markov models was proposed [7]. The following year, FAB was further extended for latent feature models [8]. In 2017, piecewise sparse linear discrimination using FAB was devised [5], [9]. This changed the prediction model from linear regression to logistic regression. Because logistic regression is computationally expensive, we obtain an approximate closed-form solution of logistic regression by using the FIC quadratic lower bound for efficient calculation [11].

In 2016, a distributed approach for heterogeneous mixture learning was devised using Apache Spark [12], [13]. This approach distributes data to multiple computers at random and generates a prediction model locally on each computer without redistributing or rereading the data. In our work, we collect the generated prediction models on only one server and integrate them to enable generation of a consistent prediction model while learning independently from each computer. This is faster than the previous approach for heterogeneous mixture learning. Specifically, this increases the number of computers to generate a predictive model without limiting the amount of data, enabling big data analysis. Also in 2016, mini-batch heterogeneous learning, which divides data into small units for learning, was announced [14]. Batch FAB has a reduction mechanism [15] that automatically shrinks models with poor accuracy. In contrast, mini-batch FAB has not only this mechanism but also a mechanism that reconsiders omitted models and integrates them each time the mini-batch model is updated. Hence, mini-batch heterogeneous mixture learning can calculate models for streaming data. Heterogeneous mixture learning has been put to practical use in various predictive analysis solutions, such as building-power forecasting for efficient resource utilization, parts-demand forecasting for inventory optimization in retail stores, and calculating optimal order quantities in an automated system.

Future works related to these studies will include expanding FAB for more diverse models, improving it for faster learning, and improving online learning.

In this paper, we propose online heterogeneous mixture learning for a GMM. We use an online EM algorithm and change the update timing of FAB and the FIC for each unit of learning data. Online heterogeneous mixture learning has better convergence than conventional batch heterogeneous mixture learning does.

4. Proposal of Online Heterogeneous Mixture Learning

As described above, FAB uses an EM algorithm to maximize the FIC. Therefore, we use an online EM algorithm for a GMM to implement heterogeneous mixture learning online.

4.1 Incremental EM Algorithm

We change the parameter update timing in the EM algorithm for each addition of new data. Specifically, we change the calculation of the E and M steps as follows, giving the incremental EM algorithm [16]–[18].

$E^\dagger$ step: We fix the parameters $\boldsymbol{\theta}$ and calculate the responsibility γ and the difference s_k between the responsibilities of the previous and current data. First, we update the responsibility for a single unit $\boldsymbol{x_n}$ of the observed data $\boldsymbol{x}_N$:

$$\gamma\left(z_{mk}\right)^{(t+1)} = \gamma\left(z_{mk}\right)^{(t)}, (n \neq n)$$
$$\gamma\left(z_{nk}\right)^{(t+1)} = \frac{\pi_k^{(t)} N\left(\boldsymbol{x}_n|\boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)}\right)}{\sum_{j=1}^K \pi_j^{(t)} N\left(\boldsymbol{x}_n|\boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)}\right)}, (k = 1, \ldots, K).$$

† "†" means incremental.

Then, we calculate the difference s_{nk} between the responsibilities of the previous data x_n and current data x_{n-1}:

$$s_{nk}^{(t+1)} = \gamma\left(z_{nk}\right)^{(t+1)} - \gamma\left(z_{nk}\right)^{(t)}$$
$$N_k^{(t+1)} = \sum_{n=1}^{N} \gamma\left(z_{nk}\right)^{(t)} + s_{nk}^{(t+1)} = N_k^{(t)} + s_{nk}^{(t)}.$$

$M^{\dagger}$ step: Here, we fix the responsibility $\gamma(z_nk)$ and the difference s_{nk} between the responsibilities of the previous and current data, and then we update the parameters:

$$\pi_k^{(t+1)} = \pi_k^{(t)} + \frac{s_{nk}^{(t+1)}}{N}$$
$$\mu_k^{(t+1)} = \mu_k^{(t)} + \frac{s_{nk}^{(t+1)}}{N_k^{(t+1)}}\left(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)}\right)$$
$$\boldsymbol{\Sigma}_k^{(t+1)} = \left(1 - \frac{s_{nk}^{(t+1)}}{N_k^{(t+1)}}\right)\left\{\boldsymbol{\Sigma}_k^{(t)} + \frac{s_{nk}^{(t+1)}}{N_k^{(t+1)}}\left(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)}\right)\left(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)}\right)^T\right\}.$$

4.2 Online Learning of Heterogeneous Mixture Learning

First, we improve the FIC, which is the model evaluation criterion, and then we improve the FAB to accommodate it.

4.2.1 Improvement for Online FIC

To obtain the information criterion for online learning, we change the FIC's update timing for each unit of learning data added. We add the data one by one (i.e., the n-th data unit x_n) and update the FIC sequentially by adding the increments of the FIC with additional data. Here, we use the same notation as in Sect. 2.

We first express the lower bound of the variational probability q of the latent variable $\boldsymbol{z^{N-1}}$ before updating with the added learning data as the following:

$$\log p\left(x^{N-1}|M\right) \geq \sum_{\mathbf{z}^{N-1}} q\left(\mathbf{z}^{N-1}\right) \log\left(\frac{p\left(\boldsymbol{x}^{N-1}, \mathbf{z}^{N-1}|M\right)}{q\left(\mathbf{z}^{N-1}\right)}\right)$$
$$p\left(x^{N-1}, \mathbf{z}^{N-1}|M\right) = \int p\left(\mathbf{z}^{N-1}|\alpha\right) \prod_{c=1}^{c} p_c\left(x^{N-1}|\mathbf{z}_c^{N-1}, \varphi_c\right) p(\boldsymbol{\theta}|M) d\boldsymbol{\theta}.$$

We then express the FIC as

$$J\left(q, \overline{\theta}, \boldsymbol{x}^N\right) = \sum_{\mathbf{z}^N} q\left(\mathbf{z}^N\right)\left[\log p\left(\boldsymbol{x}^N, \mathbf{z}^N|\bar{\theta}\right) - \frac{D_\alpha}{2}\log N - \sum_{c=1}^{C}\frac{D_c}{2}\left\{\log\left(\sum_{N=1}^{N} z_{nc}\right) - \log q\left(z^N\right)\right\}\right].$$

Next, we express the increment of the lower bound of the variational probability q of the latent variable $\boldsymbol{z_{nc}}$ for the added learning data x_n as the following:

$$q\left(z_{nc}\right)\log\left(\frac{p\left(x_n, z_{nc}|M\right)}{q\left(z_{nc}\right)}\right)$$
$$p\left(x_n, z_{nc}|M\right) = \int p\left(z_{nc}|\alpha\right)\prod_{c=1}^{C} p_c\left(x_n|z_{nc}, \varphi_c\right) p(\boldsymbol{\theta}|M) d\boldsymbol{\theta}.$$

Moreover, we express the increment of the FIC with additional data, denoted by FIC_+, as

$$\mathrm{FIC}_+\left(x_n, M\right) = \max_q\left\{J_+\left(q, \overline{\theta}, x_n\right)\right\}$$
$$J_+\left(q, \bar{\theta}, x_n\right) = q\left(z_{nc}\right)\left[\log p\left(x_n, z_{nc}|\bar{\theta}\right) - \frac{1}{2}\log N - \sum_{c=1}^{C}\frac{D_c}{2}\left\{\log z_{nc} - \log q\left(z_{nc}\right)\right\}\right].$$

Finally, from the above, we express the FIC for online heterogeneous mixture learning, FIC^*, as the following:

$$FIC_*\left(x_n, M\right) = \mathrm{FIC}\left(\boldsymbol{x}^{N-1}, M\right) + FIC_+\left(x_n, M\right) = \max_q\left\{J_*\left(q, \overline{\theta}, x_n\right)\right\}$$
$$J_*\left(q, \overline{\theta}, x_n\right) = J_*\left(q, \overline{\theta}, x_{n-1}\right) + q\left(z_{nc}\right)\left[\log p\left(x_n, z_{nc}|\bar{\theta}\right) - \frac{1}{2}\log N - \sum_{c=1}^{C}\frac{D_c}{2}\left\{\log z_{nc} - \log q\left(z_{nc}\right)\right\}\right]$$

4.2.2 Improvement for Online FAB

We cannot analytically obtain parameters, so we cannot directly evaluate $FIC_*{}^{\dagger}$. Instead, we use the variation of the variational probability of the latent variables to update the added data sequentially.

First, we define the FIC's lower bound as the following:

$$FIC_*\left(x_n, M\right) \geq G_*\left(q, \tilde{q}, \theta, x_n\right)$$
$$\sum_{z^N} q\left(\mathbf{z}^{N-1}\right)\left[\log p\left(x^{N-1}, \mathbf{z}^{N-1}|\bar{\theta}\right) - \frac{D_\alpha}{2}\log(N-1) - \sum_{c=1}^{C}\frac{D_C}{2}L\left(\sum_{n=1}^{N-1} z_{nc}, \sum_{n=1}^{N-1}\tilde{q}\left(z_{nc}\right)\right) - \log q\left(z^{N-1}\right)\right]$$
$$+q\left(z_{nc}\right)\left[\log p\left(x_n, z_{nc}|\bar{\theta}\right) - \frac{1}{2}\log N - \sum_{c=1}^{c}\frac{D_c}{2}\left(z_{nc}, \tilde{q}\left(z_{nc}\right)\right) - \log q\left(z_{nc}\right)\right]$$

† "*" means online.

Similarly to batch FAB, we fix the number of mixed data units, C, from the data partitioning candidates and consider a combinatorial optimization problem that maximizes the FIC's lower bound in the following way:

$$\boldsymbol{S}^{'}, q^{'}, \boldsymbol{\theta}^{'}, \tilde{q}^{'} = \underset{s,q,\boldsymbol{\theta},\tilde{\boldsymbol{q}}}{\operatorname{argmax}} G_{*}\left(q, \tilde{q}, \boldsymbol{\theta}, x_n\right)$$

We then repeat the following two steps t times to obtain $S^{'}, q^{'}, \boldsymbol{\theta}^{'}, \tilde{q}^{'}$ sequentially.

$V_{*}Step$: We optimize the distribution $q(z_{nc})$ of the latent variable $\boldsymbol{z}^N$ as follows. For the distribution, we also calculate the changes s_nc for additional data x_n:

$$q^{(t)} = \operatorname{argmax}_q \left\{G_{*}\left(q, \tilde{q} = q^{(t-1)}, \boldsymbol{\theta}^{(t-1)}, x_n\right)\right\}$$

$$q^{(t)}\left(z_{nc}\right) \propto \alpha_c^{(t-1)} p\left(x_n|\varphi_c^{(t-1)}\right) \exp\left(\frac{-D_c}{2\alpha_c^{(t-1)}N}\right) q^{(t)}\left(z_{nc}\right) \propto$$

$$\alpha_c^{(t-1)} p\left(x_n|\varphi_c^{(t-1)}\right) \exp\left(\frac{-D_c}{2\alpha_c^{(t-1)}N}\right)$$

$$q^{(t)}\left(z_{nc}\right) = \begin{cases} 0 & : \sum_{n=1}^{N} q^{(t)}\left(z_{nc}\right) < \delta \\ q^{(t)}\left(z_{nc}\right) / Q_C^{(t)} & : \sum_{n=1}^{N} q^{(t)}\left(z_{nc}\right) \geq \delta \end{cases}$$

$$s_{nc}^{(t)} = q^{(t)}\left(z_{nc}\right) - q^{(t-1)}\left(z_{nc}\right), (c = 1, \ldots, C)$$

$$q^{(t)}\left(z_{nc}\right) = q^{(t-1)}\left(z_{nc}\right) + s_{nc}^{(t)}$$

$M_{*}Step$: We optimize the components of the mixed mode S and parameter θ as follows. Here, $H_{*_c}\left(q^{(t)}, q^{(t)}, \varphi_c, x_n\right)$ is $G_{*}\left(q, \tilde{q}, \boldsymbol{\theta}, x_n\right)$ for the c-th model when the number of data divisions is C. Because the number of candidates for division is finite, we optimize φ_c for each S_c and compare to find the optimal S_c:

$$\boldsymbol{S}^{(t)}, \boldsymbol{\theta}^{(t)} = \underset{\boldsymbol{S},\boldsymbol{\theta}}{\operatorname{grgmax}} G\left(q^{(t)}, \tilde{q} = q^{(t)}, \theta, x_n\right)$$

$$\alpha_c^{(t)} = \frac{\sum_{n=1}^{N} q^{(t-1)}\left(z_{nc}\right)}{N} + \frac{\sum_{n=1}^{N} s_{nc}^{(t)}}{N} = \alpha_c^{(t-1)} + \frac{\sum_{n=1}^{N} s_{nc}^{(t)}}{N}$$

$$S_c^{(t)}, \varphi_c^{(t)} = \arg\max_{S_c,\varphi_c} H_{*}\left(q^{(t)}, q^{(t)}, \varphi_c, x_n\right)$$

$$H_{*}\left(q^{(t)}, q^{(t)}, \varphi_c, x_n\right) = \sum_{n=1}^{N} q^{(t)}\left(z_{nc}\right) \log p\left(x_n|\varphi_c\right)$$

$$-\frac{D_c}{2} L\left(\sum_{n=1}^{N-1} q^{(t)}\left(z_{nc}\right), \sum_{n=1}^{N-1} q^{(t)}\left(z_{nc}\right)\right)$$

$$-\frac{D_c}{2} L\left(q^{(t)}\left(z_{nc}\right), q^{(t)}\left(z_{nc}\right)\right)$$

In these two steps, the following inequality holds.

$$FIC_{*}^{(t)}\left(x_n, M\right) \geq FIC_{*}^{(t-1)}\left(x_n, M\right) \equiv G\left(q^{(t)}, q^{(t)}, \theta^{(t)}, x_n\right)$$

We use this to define the following condition ϵ for termination after learning all the data:

$$FIC_{*}^{(t)}\left(\boldsymbol{x}^{\boldsymbol{N}}, M\right) - FIC_{*}^{(t-1)}\left(\boldsymbol{x}^{\boldsymbol{N}}, M\right) \leq \varepsilon$$

If the termination condition is not met, we learn all the data again.

Here, we propose an online heterogeneous mixture learning algorithm for a GMM as listed below.

Algorithm 2 Online FAB algorithm for mixture models

Require: $\boldsymbol{x}^N, C_{\max}, \boldsymbol{S}, \varepsilon, \delta$
Ensure: $C^{'}, S^{'}, \boldsymbol{\theta}^{'}, q^{'}\left(\mathbf{z}^N\right), FIC_{LB}^{*}$
$FIC_{LB}^{*} = -\infty$
$t = 0, FIC_{LB}^{(0)} = -\infty, \delta = 0, T_c = 0$ //T_c:Maximum number of repetitions for C mixtures
$q^{(0)}\left(\mathbf{z}^N\right)$Random initialization
while $FIC_{LB}^{(t)}\left(x^N, M\right) - FIC_{LB}^{(t-1)}\left(x^N, M\right) \leq \varepsilon$ **do**
  Randomize order of data x
  for $x = 1, \cdots x_n$ **do**
    for $c = 1, \cdots, C_{max}$ **do**
      //V Step:$q^{(t)}$calculation
      $q^{(t)}\left(z_{nc}\right) \propto \alpha_c^{(t-1)} p\left(\boldsymbol{x}_n|\varphi_c^{(t-1)}\right) \exp\left(\frac{-D_c}{2\alpha_c^{(t-1)}N}\right)$
      $q^{(t)}\left(z_{nc}\right) = \begin{cases} 0 & : \sum_{n=1}^{N} q^{(t)}\left(z_{nc}\right) < \delta \\ q^{(t)}\left(z_{nc}\right) / Q_c^{(t)} & : \sum_{n=1}^{N} q^{(t)}\left(z_{nc}\right) \geq \delta \end{cases}$
      $s_{nc}^{(t)} = q\left(z_{nc}\right)^{(t)} - q\left(z_{nc}\right)^{(t-1)}, (c = 1, \ldots, C)$
      //Preparation confirmation for end condition
      $FIC_{LB}^{(t)}\left(x^N, M\right) += FIC_{LB}^{(t)}\left(x^n, M\right) - FIC_{LB}^{(t-1)}\left(x^n, M\right)$
      //M Step : $\boldsymbol{S}^{(t)}, \boldsymbol{\theta}^{(t)}$calculate
      $\alpha_c^{(t)} = \frac{\sum_{n=1}^{N} q^{(t-1)}\left(z_{nc}\right)}{N} + \frac{\sum_{n=1}^{N} s_{nc}^{(t)}}{N} = \alpha_c^{(t-1)} + \frac{\sum_{n=1}^{N} s_{nc}^{(t)}}{N}$
      $S_c^{(t)}, \varphi_c^{(t)} = \arg\max_{S_c,\varphi_c}\left(q^{(t)}, q^{(t)}, \varphi_c, x^N\right)$
      $t = t + 1$
    end for
  end for
  $T_c = t, FIC_{LB}^{(T_c)}, \boldsymbol{S}^{(T_c)}, \boldsymbol{\theta}^{(T_c)}, q^{(T_c)}\left(\mathbf{z}^N\right)$
end while
$//M^{'} = \left(C^{'}, S^{'}\right), \boldsymbol{\theta}^{'}, q^{'}\left(\mathbf{z}^N\right)$choose
$M^{'}, q^{'}, \boldsymbol{\theta}^{'}, \tilde{q}^{'} = \arg\max_{M,q,\theta,\tilde{q}} G\left(q, \tilde{q}, \theta, x^N\right)$

5. Experiments and Evaluation

In this section, we compare the learning results with the conventional batch heterogeneous mixture learning and the online heterogeneous mixture learning proposed in this paper.

5.1 Experimental Environment

We implemented the learning algorithm source code in C++. We used the arithmetic library Eigen[†] for some of the numerical calculations. We evaluated the learning algorithms in the environment listed in Table 1.

5.2 Experimental Data

The data used for learning in this experiment was generated with a Gaussian random number from the GMM. We preset

Table 1 Experimental environment

OS	macOS Mojave version10.14.12
CPU	Intel(R) Core i5
Operating frequency	2.70GHz
RAM	8GB
Eigen version	3.3.4
C++ compiler	Apple LLVM version 9.0.0 (clang-900.0.38)

[†]http://eigen.tuxfamily.org/index.php

Table 2 Experimental data

Number of data units	10,000
Number of mixtures	4
Mixing coefficient	0.1, 0.2, 0.3, 0.4
Mean, covariance matrix	random
Number of dimensions	10

the parameters necessary for the GMM, such as the mixing coefficient, mean, covariance matrix, and number of dimensions. Because the covariance matrix is a semipositive definite matrix, we changed the random condition so that all eigenvalues were 0 or greater. Table 2 lists the details of the data set.

5.3 Kullback-Leibler (KL) Divergence

The KL divergence is a measure of the similarity between two probability distributions. Given continuous probability distributions P and Q, p and q represent their respective probability density functions. We define the KL divergence as the following:

$$D_{\mathrm{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx.$$

Because a GMM is a superposition of multiple normal distributions, we apply a variational approximation [10] to two multivariate mixed normal distributions to approximate the KL divergence:

$$D_{\mathrm{KL}}(P\|Q) = \sum_a \pi_a \log \left(\frac{\sum_{a'} \pi_{a'} exp(-KL(N(;\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a), N(;\boldsymbol{\mu}_{a'}, \boldsymbol{\Sigma}_{a'})))}{\sum_b \pi_b exp(-KL(N(;\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a), N(;\boldsymbol{\mu}_b, \boldsymbol{\Sigma}_b)))} \right).$$

This allows us to calculate the KL divergence in a closed form even for a GMM.

5.4 Experimental Method

Using the above experimental data, we compared and examined the results of heterogeneous mixed learning of the batch and online types. We used the KL divergence to compare the true parameters of the experimental data with the predicted parameters obtained by learning.

In online learning the speed of learning convergence is also important, so we measured the change in the FIC for each iteration. We also evaluated the number of iterations until convergence.

We conducted experiments 10 times on each FIC value, iteration count, KL divergence, and execution time after learning for each algorithm, and we regarded the averages as the results.

5.5 Experimental Results

We can roughly divide the experiments into three types. The first type was an experiment performed with different numbers of data units: 10000, 500, 1000, and 100000. The second type was an experiment performed with different numbers of dimensions: 2, 4, 20, and 40. Finally, the third was an experiment performed with different numbers of mixtures: 1, 2, 8, and 12.

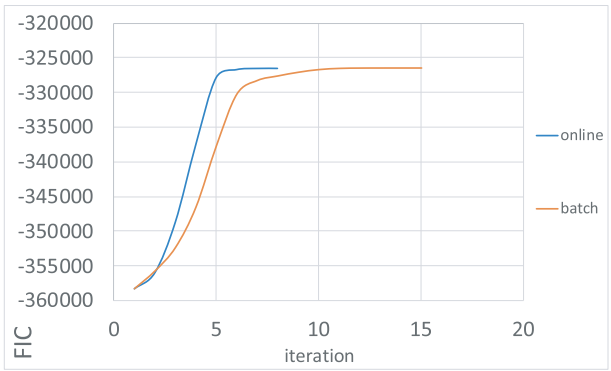

Fig. 1 Change in FIC for each iteration on number of 10000 data units

Table 3 Experimental results (10000 data units)

	Batch	Online
FIC	−326,477	−326,478
KL-divergence	0.012616227	0.012652214
Iteration	13.9	8.1
Execution time	1.1s	3.3s

5.5.1 Different Numbers of Data Units

First, for 10000 data units, Fig. 1 shows the change in the FIC with the number of iterations for the conventional batch heterogeneous mixture learning and online heterogeneous mixture learning.

For each learning method in this experiment, Table 3 lists the FIC value, number of iterations, KL divergence, and execution time at convergence.

Next, we changed the number of data units to 500. For each method, Fig. 2 shows the resulting change in the FIC with the number of iterations, and Table 4 lists the FIC value, number of iterations, KL divergence, and execution time at convergence.

Next, we changed the number of data units to 1000. Figure 3 shows the change in the FIC, and Table 5 lists the results.

Finally, we changed the number of data units to 100000. Figure 4 shows the change in the FIC, and Table 6 lists the results.

When the number of data units was reduced to 500, the KL divergence was larger than in the other experiments. We think that, because the number of data units was small, a highly accurate model could not be generated with either the online or batch method. When the number of data units was 10000, the FIC value was higher with the online method than with the batch method. In many of the experiments, the FIC and KL divergence were accurate with the online

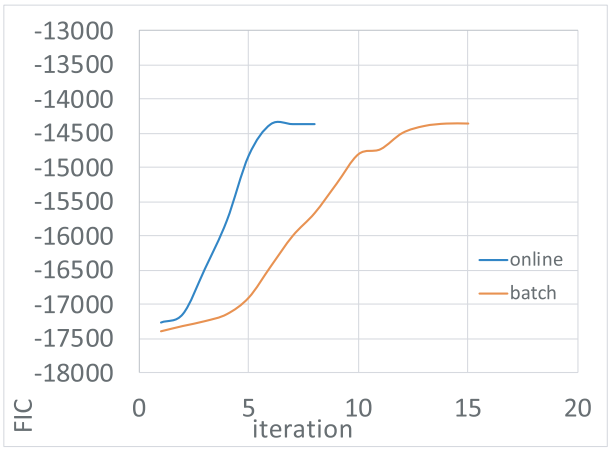

Fig. 2 Change in FIC for each iteration on the number of 500 data units

Table 4 Experimental results (500 data units)

	Batch	Online
FIC	−14,588.58	−144,65.34
KL-divergence	1.330983532	1.330983532
Iteration	14.4	9.6
Execution time	0.06s	0.24s

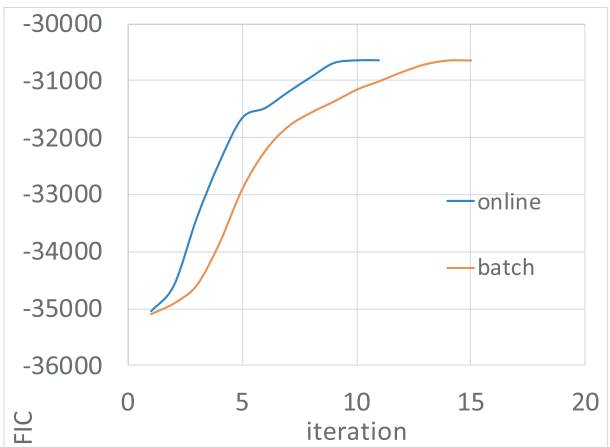

Fig. 3 Change in FIC for each iteration on the number of 1000 data units

Table 5 Experimental results (1000 data units)

	Batch	Online
FIC	−30645.2	−30645.2
KL-divergence	0.147320073	0.148577096
Iteration	14.2	13
Execution time	0.1207s	0.6028s

method. Therefore, we conclude that it is better to use the online method when accuracy is important.

5.5.2 Experiments by Changing the Number of Dimensions

I have compared experiments of dimensions 2, 4, 20, and 40. Figure 5 which is the dimensions 2 experiment shows that

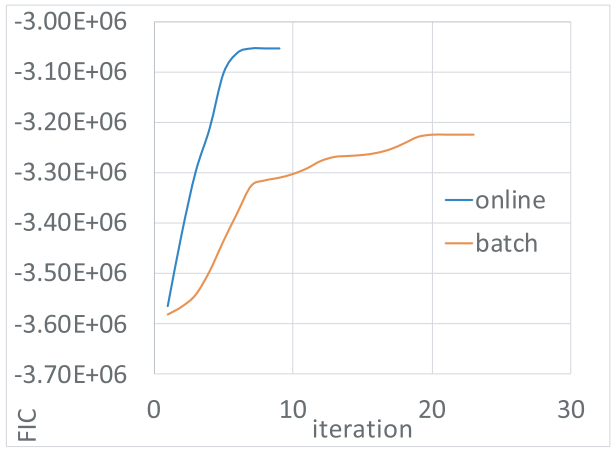

Fig. 4 Change in FIC for each iteration on the number of 100000 data units

Table 6 Experimental results (100000 data units)

	Batch	Online
FIC	−3,110,312	−3,052,230
KL-divergence	0.59304168	0.001532966
Iteration	19.8	9.6
Execution time	16.4s	39.1s

the online method takes about twice as long as the batch method. Figure 6 which is the dimensions 4 experiment shows that the online method takes about twice as long as the batch method. Figure 7 which is the dimensions 20 experiment shows that the online method takes about 7 times as long as the batch method. Figure 8 which is the dimensions 40 experiment shows that the online method takes about 16 times as long as the batch method. Therefore, as the dimension increases, the execution time of the online method becomes more than that of the batch method.

On the other hand, Fig. 12 which is the mixtures 12 experiment shows that the online method only takes about 3 times as long as the batch method.

First, we set the number of dimensions to 2. Figure 5 shows the change in the FIC with the number of iterations for the conventional batch heterogeneous mixture learning and online heterogeneous mixture learning. For each learning method, Table 7 lists the FIC value, number of iterations, KL divergence, and execution time at convergence.

Next, we changed the number of dimensions to 4. For each method, Fig. 6 shows the resulting change in the FIC, and Table 8 lists the FIC value, number of iterations, KL divergence, and execution time at convergence.

Next, we changed the number of dimensions to 20. Figure 7 shows the change in the FIC, and Table 9 lists the results.

Finally, we changed the number of dimensions to 40. Figure 8 shows the change in the FIC, and Table 10 lists the results.

In these experiments changing the number of dimensions, there was no difference in accuracy between the online and batch methods. The online method converged with

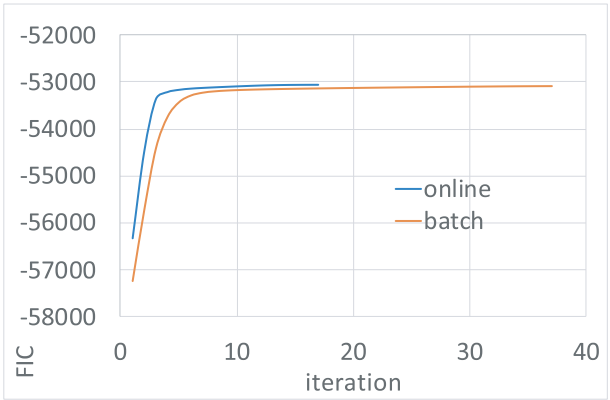

Fig. 5 Change in FIC for each iteration on the number of 2 dimensions

Table 7 Experimental results (2 dimensions)

	Batch	Online
FIC	−53096.36	−53074.46
KL-divergence	0.062051642	0.028706503
Iteration	33.2	21.8
Execution time	2.0355s	5.2325s

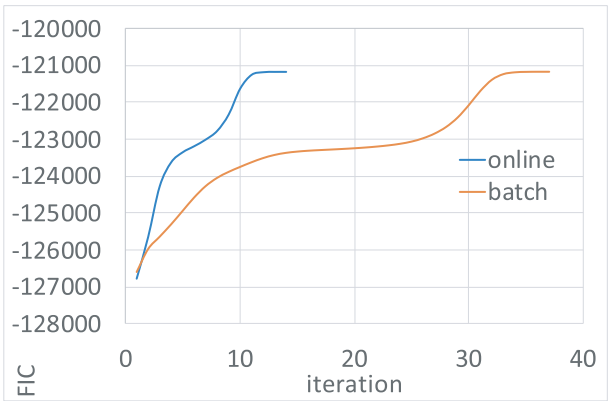

Fig. 6 Change in FIC for each iteration on the number of 4 dimensions

Table 8 Experimental results (4 dimensions)

	Batch	Online
FIC	−121184.8	−121182.6
KL divergence	0.004023055	0.003438413
Iteration	39	18.6
Execution time	2.5107 s	4.8880 s

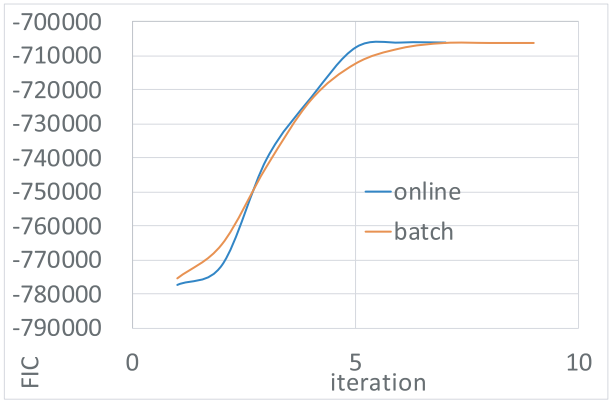

Fig. 7 Change in FIC for each iteration on the number of 20 dimensions

Table 9 Experimental results (20 dimensions)

	Batch	Online
FIC	−706,172	−706,172
KL divergence	0.044111843	0.044104172
Iteration	9.8	6.8
Execution time	0.954 s	6.124 s

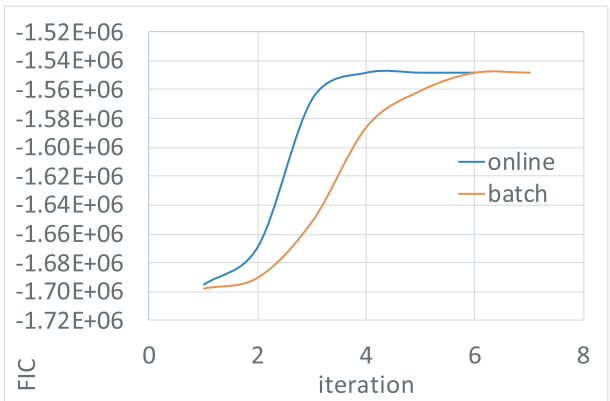

Fig. 8 Change in FIC for each iteration on the number of 40 dimensions

Table 10 Experimental results (40 dimensions)

	Batch	Online
FIC	−1,558,110	−1,558,110
KL divergence	1.113930988	1.113931093
Iteration	7	6.2
Execution time	1.2189 s	17.8644 s

a small number of iterations, however, in each experiment. On the other hand, the execution time for the online method was longer than that for the batch method.

5.5.3 Different Numbers of Mixtures

First, we set the number of mixtures to 1. Figure 9 shows the change in the FIC with the number of iterations for the conventional batch heterogeneous mixture learning and online heterogeneous mixture learning. For each learning method, Table 11 lists the FIC value, number of iterations, KL divergence, and execution time at convergence.

Next, we changed the number of mixtures to 2. Figure 10 shows the resulting change in the FIC, and Table 12 lists the FIC value, number of iterations, KL divergence, and execution time at convergence.

Next, we changed the number of mixtures to 8. Figure 11 shows the change in the FIC, and Table 13 lists the results.

Finally, we changed the number of mixtures to 12. Figure 12 shows the change in the FIC, and Table 14 lists the results.

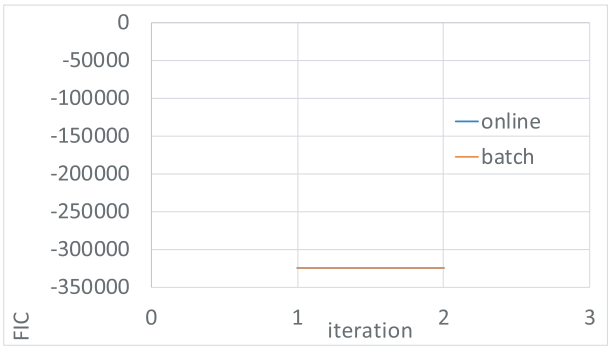

Fig. 9 Change in FIC for each iteration on the number of 1 mixture

Table 11 Experimental results (1 mixture)

	Batch	Online
FIC	−324751	−324751
KL divergence	0.003790533	0.003790533
Iteration	2	2
Execution time	0.0440 s	0.4055 s

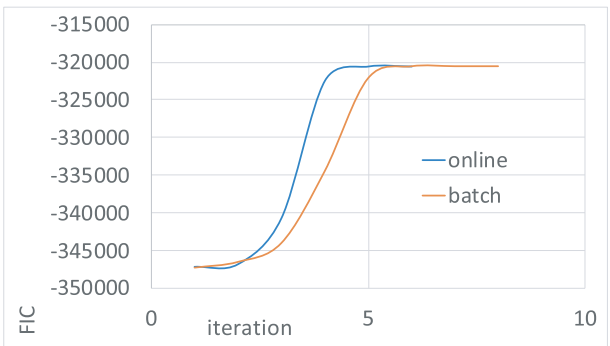

Fig. 10 Change in FIC for each iteration on the number of 2 mixtures

Table 12 Experimental results (2 mixtures)

	Batch	Online
FIC	−320545	−320545
KL divergence	0.007279893	0.007279893
Iteration	9.4	7
Execution time	0.3609 s	1.6481 s

In changing the number of mixtures, the experimental results did not vary from those with one mixture. A GMM with one mixture is simply a normal distribution. Therefore, we did not have to choose a model. In the cases of two and four mixtures, both the FIC and the KL divergence for the online method were superior to those for the batch method. On the other hand, with 12 mixtures, both the FIC and the KL divergence for the batch method were superior to those for the online method. When the number of mixtures is large, the mixing coefficient is small, meaning that a model that does not have a big influence on the whole may be ignored. We think that this affects the accuracy.

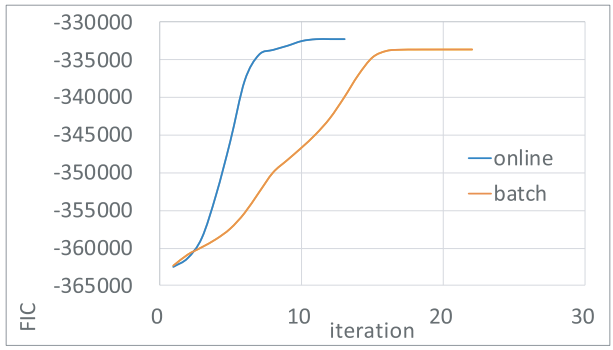

Fig. 11 Change in FIC for each iteration on the number of 8 mixtures

Table 13 Experimental results (8 mixtures)

	Batch	Online
FIC	−336,621	−333,082
KL divergence	0.152345248	0.128347962
Iteration	22.4	15
Execution time	3.37 s	11.6 s

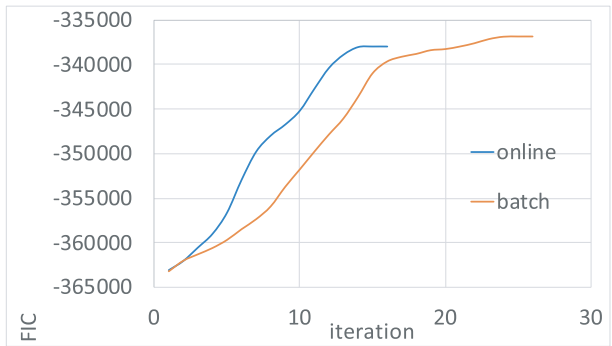

Fig. 12 Change in FIC for each iteration on the number of 12 mixtures

Table 14 Experimental results (12 mixtures)

	Batch	Online
FIC	−335076.6	−336666.2
KL divergence	0.315016827	0.478644756
Iteration	26.8	16.4
Execution time	5.946836 s	18.3395 s

5.6 Evaluation and Discussion

The points of evaluation of online heterogeneous mixture learning are the followings:

- The average number of iterations for online heterogeneous mixture learning is less than it is for batch heterogeneous mixture learning.
- The online method is superior to the batch method in terms of using the results even while learning.
- The execution time of the online method is about three times longer than that of the batch method.

The details of the evaluation are as follows:

As summarized above, when we see a change in the

FIC for each iteration, we know that the average number of iterations is lower for the online method than for the batch method. Furthermore, the online method can use results even while learning, while quickly converging. On the other hand, the execution time for the online method is about three times longer than that of the batch method. The computational cost of the determinant and inverse matrix required for data update is very large. Online heterogeneous learning calculates parameters as data enters the learning machine, so the amount of calculation for determinants and inverses increases. We think that the execution time increases mainly because of that. Indeed, 75% percent of the first experiment's run time was for that calculation.

In GMM parameter prediction, we think that the FIC value and KL divergence for the online method are generally better than for the batch method. Moreover, the number of iterations until convergence for the online method is less than for the batch method. In addition, because online learning updates parameters one by one, we can use results even during learning, which is a major advantage of the online method over the batch method. In general, however, here are advantages and disadvantages to both online heterogeneous mixture learning and batch heterogeneous mixture learning. We think that it is important to select the appropriate algorithm by considering various factors, such as the data and the learning environment.

6. Conclusion and Future Works

The motivation of this research is to improve the speed of convergence of machine learning for data with heterogeneity, as the quantity of such data is expected to increase in the real world from Internet of Things (IoT) devices. To that end, we proposed heterogeneous mixture learning with an online learning method. We then experimentally evaluated both online heterogeneous mixture learning and batch heterogeneous mixture learning. We found a major advantage of the online method over the batch method, namely, that online heterogeneous mixture learning is useful when real-time results are necessary. Our future works will include eliminating bottlenecks in the determinant and inverse matrix calculation costs by applying a variant of the online EM algorithm called the stepwise EM algorithm. This algorithm is scalable in the work area, which enables learning of heterogeneous streaming data in real time without having to keep all the data. By applying the stepwise EM algorithm in heterogeneous mixture learning, we expect to implement heterogeneous mixture learning with real-time characteristics.

References

[1] J. Jose, Internet of Things, Khanna Publishing House, 2018.
[2] C.M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer, 2007.
[3] A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin, "Bayesian Data Analysis," Chapman and Hall/CRC, 2013.
[4] K. Murphy, "Machine Learning: A Probabilistic Perspective," 2012.
[5] R. Hujimaki, Y. Yamaguchi, and R. Eto, "Piecewise Sparse Linear Discrimination by FAB," Journal of the Japanese Society for Artificial Intelligence, vol.32, no.1, pp.30–38, Jan. 2017.
[6] R. Fujimaki and S. Morinaga, "Factorized Asymptotic Bayesian Inference for Mixture Modeling," Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, PMLR, vol.22, pp.400–408, 2012.
[7] R. Fujimaki and K. Hayashi, "Factorized Asymptotic Bayesian Hidden Markov Models," International Conference on Machine Learning (ICML), pp.799–806, 2012.
[8] K. Hayashi and R. Fujimaki, "Factorized Asymptotic Bayesian Inference for Latent Feature Models," Advances in Neural Information Processing Systems (NIPS), pp.1214–1222, 2013.
[9] R. Eto, R. Fujimaki, S. Morinaga, and H. Tamano, "Fully-Automatic Bayesian Piecewise Sparse Linear Models," International Workshop on Artificial Intelligence and Statistics (AISTATS), pp.238–246, 2014.
[10] J.R. Hershey and P.A. Olsen, "Approximating the Kullback Leibler Divergence Between Gaussian Mixture Models," 2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07, 2007.
[11] T. Zhang, "Adaptive Forward-Backward Greedy Algorithm for Learning Sparse Representations," IEEE Transactions on Information Theory, vol.57, no.7, pp.4689–4708, July 2011.
[12] M. Asahara and R. Fujimaki, "Big Data Heterogeneous Mixture Learning on Spark," Hadoop Summit San Jose, 2016.
[13] M. Asahara and R. Fujimaki, "Distributed Heterogeneous Mixture Learning On Spark," Spark Summit, 2016.
[14] C. Liu, L. Feng, and R. Fujimaki, "Streaming Model Selection via Online Factorized Asymptotic Bayesian Inference," IEEE International Conference on Data Mining (ICDM), pp.271–280, 2016
[15] L. Theis and M. Hoffman, "A Trust-Region Method for Stochastic Variational Inference with Applications to Streaming Data," International Conference on Machine Learning (ICML), pp.2503–2511, 2015.
[16] P. Liang and D. Klein, "Online EM for Unsupervised Models," NAACL '09 Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp.611–619, 2009.
[17] J. Zeng, Z.-Q. Liu, and X.-Q. Cao, "Fast Online EM for Big Topic Modeling," IEEE Transactions on Knowledge and Data Engineering, vol.28, no.3, pp.675–688, 2016.
[18] O. Cappé and E. Moulines, "On-line Expectation-Maximization Algorithm for Latent Data Models," Journal of the Royal Statistics Society: Series B (Statistical Methodology), vol.71, no.3, pp.593–613, 2009.

Appendix A: Data Processing in Machine Learning

Data processing in machine learning is roughly classified into batch learning, online learning, and mini-batch learning [2].

A.1 Batch Learning

Batch learning is a machine learning method in which all data is first inputted to a learning machine, and then the parameters are updated using the entire data. The benefits of batch learning are higher generalization performance and stability as compared to online learning. On the other hand, the disadvantages of this method are that it requires accumulating the entire learning data for learning and that the learning results cannot be used until learning ends.

A.2 Online Learning

On the other hand, in online learning only a small amount of data is given to the learning machine. The learning parameters are then updated as data is added. If the learning data has a large size and includes dependencies within the data, then online learning has the benefit of getting results faster than batch learning does. On the other hand, the disadvantages are that the learning rate must be set appropriately and the order of learning data depends on the learning results.

A.3 Mini-Batch Learning

Finally, mini-batch learning is an intermediate method between batch learning and online learning. This method divides the learning data into groups of roughly the same size and then updates the parameters for each group. Increasing the size of the groups eventually leads to batch learning, while decreasing the size of the groups leads to online learning. Hence, mini-batch learning requires considering how to divide the data into groups (i.e., considering the batch size).

Appendix B: Gaussian Mixture Model and EM Algorithm

The EM algorithm [2] is one method for maximum likelihood estimation [3] of probability model parameters in machine learning. In this study, we use the EM algorithm to estimate a Gaussian mixture model (GMM) [4]. Hence, this section describes the multi-dimensional GMM and the EM algorithm.

B.1 Normal Distribution

A normal distribution is a probability distribution representing data that accumulates near the mean. For a normal distribution, the probability density function $N(x;\mu,\sigma^2)$ is expressed as

$$N\left(x;\mu,\sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

where x is the input data, μ is the mean, and $\sigma^2 > 0$ is the variance. A normal distribution with multiple dimensions is called a multivariate normal distribution.

The probability density function of an n-variate normal distribution $N_n(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma})$ is expressed as

$$N_n(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

where $\boldsymbol{\Sigma}$ is the covariance matrix.

B.2 Gaussian Mixture Model (GMM)

A GMM is expressed as a linear superposition of normal distributions. The number of mixtures is K, while the weighting factor for the linear combination is π. The probability density function of a GMM $p(\boldsymbol{x};\boldsymbol{\theta})$ is then expressed as

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K}\pi_k N(\boldsymbol{x}|\boldsymbol{\mu}_k,\boldsymbol{\sigma}_k^2), 0 \le \pi_k \le 1, \sum_{k=1}^{K}\pi_k = 1$$
$$\boldsymbol{\theta} = \left(\boldsymbol{\mu}_1^\top,\ldots,\boldsymbol{\mu}_K^\top,\boldsymbol{\sigma}_1^2,\ldots,\boldsymbol{\sigma}_K^2,\pi_1,\ldots,\pi_K\right)^\top$$

where $\boldsymbol{\mu_k}$ is the vector of means belonging to class k, and $\boldsymbol{\sigma_k^2}$ is the vector of variances belonging to k. Here, we estimate the parameters $\boldsymbol{\theta}$. The probability density function of an n-variate GMM $p_n(\boldsymbol{x}|\boldsymbol{\theta})$ is then expressed as

$$p_n(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K}\pi_k N_n(\boldsymbol{x}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k), 0 \le \pi_k \le 1, \sum_{k=1}^{K}\pi_k = 1$$
$$\boldsymbol{\theta} = \left(\boldsymbol{\mu}_1^\top,\ldots,\boldsymbol{\mu}_K^\top,\boldsymbol{\Sigma}_1,\ldots,\boldsymbol{\Sigma}_K,\pi_1,\ldots,\pi_K\right)^\top$$

where $\boldsymbol{\Sigma_k}$ is the n-dimensional covariance matrix belonging to class k.

B.3 Responsibility

The variable z represents the number of normal distributions of a GMM from the observed data x_n:

$$z_{nk} = \begin{cases} 1 : \text{Input data}\boldsymbol{x_n}\text{belong to class k} \\ 0 : \text{Input data}\boldsymbol{x_n}\text{does not belong to class k} \end{cases}$$

We can regard the posterior probability of z given x as the responsibility $\gamma(z_k)$:

$$\gamma\left(z_{nk}\right) \equiv p\left(z_k = 1|\boldsymbol{x_n}\right) = \frac{\pi_k N(\boldsymbol{x_n}|\boldsymbol{\mu_k},\boldsymbol{\Sigma_k})}{\sum_{j=1}^{K}\pi_j N(\boldsymbol{x_n}|\boldsymbol{\mu_j},\boldsymbol{\Sigma_j})}.$$

Here, the responsibility is an expected value that indicates whether observed data belongs to class k.

B.4 Maximum Likelihood Estimation by EM Algorithm

Next, we consider the problem of fitting observation data to the GMM. The posterior probability of observed data $\boldsymbol{x_n}$ given the parameter $\boldsymbol{\theta}$ is called the likelihood. Then, the likelihood $L(\boldsymbol{\theta})$ is expressed as

$$L(\boldsymbol{\theta}) := \prod_{n=1}^{N} p\left(\boldsymbol{x}_n|\boldsymbol{\theta}\right).$$

Next, the logarithm of the likelihood, or log-likelihood, is expressed as

$$\log L(\boldsymbol{\theta}) = \log\prod_{n=1}^{N} p\left(\boldsymbol{x}_n|\boldsymbol{\theta}\right)$$
$$= \sum_{n=1}^{N}\log p\left(\boldsymbol{x}_n|\boldsymbol{\theta}\right).$$

From this, we formulate the parameter estimation of the GMM.

$$\hat{\theta} := \underset{\theta}{\operatorname{argmax}} L(\boldsymbol{\theta}) \text{ subject to } 0 \leq \pi_k \leq 1, \sum_{k=1}^{K} \pi_k = 1.$$

Then, we derive the maximum likelihood estimate:

$$\begin{cases} \hat{\pi}_k = \frac{1}{N} \sum_{n=1}^{N} \gamma(z_{nk}) = \frac{N_k}{N} \\ \widehat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n \\ \widehat{\boldsymbol{\Sigma}}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \end{cases}$$

$$\begin{cases} N_k = \sum_{n=1}^{N} \gamma(z_{nk}) \\ \gamma(z_{nk}) = \frac{\pi_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j N(\boldsymbol{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \end{cases}$$

Because the log-likelihood of the GMM has a log-sum part, we cannot solve it analytically. Instead, we use the EM algorithm to approximate the solution.

On the expectation step (E step), we calculate the expected value of the model's log-likelihood by using the currently estimated probability distribution of the latent variables. On the maximization step (M step), we calculate a parameter that maximizes that expectation. The EM algorithm is thus an iterative method that repeats the E and M steps to perform maximum likelihood estimation. Here, we describe the EM algorithm for the GMM.

Step 1: Input initial values for the prior probability π_k, the mean π_k, and the covariance $\boldsymbol{\Sigma_k}$.
Step 2 (E step): Calculate the responsibility $\gamma(z_{nk})$ for each data group given the current parameters.

$$\gamma(z_{nk}) = \frac{\pi_k N(\boldsymbol{x_n}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j N(\boldsymbol{x_n}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

Step 3 (M step): Calculate the maximum likelihood estimator given the responsibility.

$$\begin{cases} \hat{\pi}_k = \frac{1}{N} \sum_{n=1}^{N} \gamma(z_{nk}) \\ \widehat{\boldsymbol{\mu}}_k = \frac{1}{\sum_{n=1}^{N} \gamma(z_{nk})} \sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n \\ \widehat{\boldsymbol{\Sigma}}_k = \frac{1}{\sum_{n=1}^{N} \gamma(z_{nk})} \sum_{n=1}^{N} \gamma(z_{nk}) (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \end{cases}$$

Step 4: Calculate the log-likelihood $logL(\boldsymbol{\theta})$. If its increase is below a threshold, then the algorithm finishes; otherwise, it returns to step 2.

Kazuki Seshimo received a B.S. degree from Kanazawa University in 2018. He is now an M.S. student studying Bayesian statistics.

Akira Ota received an M.S. degree from Kanazawa University in 2019, studying Bayesian statistics.

Daichi Nishio received a B.S. degree from Kanazawa University in 2017. He is now an M.S. student studying reinforcement learning.

Satoshi Yamane received B.S., M.S., and Ph.D. degrees from Kyoto University. He is now a professor at Kanazawa University, focusing on formal verification of real-time and distributed computing.