PAPER

Look-Up-Table-Based Exponential Computation and Application to an EM Algorithm for GMM

Hidenori WATANABE[†a)], *Student Member and* Shogo MURAMATSU[††b)], *Senior Member*

SUMMARY This work proposes an exponential computation with low-computational complexity and applies this technique to the expectation-maximization (EM) algorithm for Gaussian mixture model (GMM). For certain machine-learning techniques, such as the EM algorithm for the GMM, fast and low-cost implementations are preferred over high precision ones. Since the exponential function is frequently used in machine-learning algorithms, this work proposes reducing computational complexity by transforming the function into powers of two and introducing a look-up table. Moreover, to improve efficiency the look-up table is scaled. To verify the validity of the proposed technique, this work obtains simulation results for the EM algorithm used for parameter estimation and evaluates the performances of the results in terms of the mean absolute error and computational time. This work compares our proposed method against the Taylor expansion and the exp() function in a standard C library, and shows that the computational time of the EM algorithm is reduced while maintaining comparable precision in the estimation results.

key words: *EM algorithm, Gaussian mixture model, exponential function*

1. Introduction

This paper proposes a method for reducing the computational cost of exponential function in EM algorithm for Gaussian mixture model (GMM).

GMM is a probabilistic distributions expressed by the weighted sum of Gaussian distribution. The GMM has a wide application since it is able to represent a complicate probabilistic distribution. Fujimoto et al. modeled speech sounds with GMM for noise reduction in car environment [1]. Rotem et al. used in their proposed approach for image segmentation [2].

GMM-based applications require GMM parameters, such as the mixture ratio, mean, and variance. If the number of mixture is known, the EM algorithm is used to estimate the GMM parameters. The EM algorithm proposed by Dempster et al. [3] and then researchers improved the precision of parameter estimation and computational cost. Since the computational cost of the EM algorithm is non-negligible, it is seldom used in embedded systems. For example, Lie et al. noted that the EM algorithm could not be processed in real-time, and instead proposed a non EM-based method for fast parameter estimation [4]. However, their proposed method can only be used in applications where inaccurate parameter estimation is acceptable. Fujimoto et al. proposed an EM-based method that estimates only the mean vector and the other parameters are considered as constants [1]. As seen above, the EM algorithm is not well-suited for embedded systems, and a reasonable compromise must be applied to reduce its computational cost.

This paper focuses on the computational cost of the exponential function of the GMM. Since the exponential function is typically approximated through recursive computations, such as the Taylor expansion, it is not well-suited for embedded systems. If a low-computational algorithm could be used to approximate the exponential function, it would not be necessary to apply any constraint to reduce the computational cost for estimating the parameters.

In previous work, we proposed a fast GMM-based classification method that reduces the computational cost of the exponential function [5], [6]. In our previous method, we reduced the computational cost of a GMM-based classifier by approximating the exponential function using a look-up table and its bit shift. In this work, we will apply this idea to the EM algorithm used to estimate the GMM parameters.

Specifically, in this work we use the method proposed in our prior work to approximate the exponential function. To verify the effectiveness of the proposed method, we perform simulations to evaluate the precision of the estimated parameters and the computational time.

2. Review of EM Algorithm for GMM

In this section, we review the EM algorithm for the GMM and discuss the computational complexity.

2.1 E-Step

The EM algorithm consists of two principal processes. First, the expectation-step (E-step) calculates probabilities, which are called responsibilities. Then, the maximization-step (M-step) updates the parameters of GMM by using these responsibilities. These two steps are repeated until the parameters converge. In the followings, let us show the equation of the E-step where the notations are based on [7].

In the E-step, we compute the responsibilities $\gamma_{k,n}$, where k is the distribution number and n is the data index. Suppose that we observe input vectors $\{\mathbf{x}_n\}_{n=0}^{N-1}$, where $\mathbf{x}_n \in \mathbb{R}^{D\times 1}$ and N is the number of data points. The respon-

Manuscript received October 9, 2012.
Manuscript revised January 11, 2013.

[†]The author is with the Graduate School of Science and Technology, Niigata University, Niigata-shi, 950-2181 Japan.
[††]The author is with the Faculty of Engineering, Niigata University, Niigata-shi, 950-2181 Japan.
a) E-mail: hide-w@telecom0.eng.niigata-u.ac.jp
b) E-mail: shogo@eng.niigata-u.ac.jp

DOI: 10.1587/transfun.E96.A.935

sibility $\gamma_{k,n}$ of the k-th distribution for the n-th input vector $\mathbf{x}_n$ is given by

$$\gamma_{k,n} = \frac{\alpha_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=0}^{K-1} \alpha_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}, \tag{1}$$

where K is the number of Gaussian distributions, α_k is the mixture ratio of the k-th distribution, and $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are the mean vector and covariance matrix of the k-th distribution, respectively, where $\boldsymbol{\mu}_k \in \mathbb{R}^{D\times 1}$ and $\boldsymbol{\Sigma}_k \in \mathbb{R}^{D\times D}$. $\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the k-th multivariate Gaussian distribution given as

$$\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = C_k \cdot \exp(-y_{k,n}), \tag{2}$$

where

$$C_k = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_k|^{1/2}}, \tag{3}$$

and

$$y_{k,n} = \frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k). \tag{4}$$

Note that C_k and $y_{k,n}$ are non-negative scalar values.

For the M-step, refer to the article [7].

2.2 Computation of EM Algorithm for GMM

In the followings, we present our approach for reducing the cost associated with computing the E-step. Let us decompose the computational procedures in Eq. (1) as follows:

1. $y_{k,n} = \frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k)$,
2. $u_{k,n} = \exp(-y_{k,n})$,
3. $C_k = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_k|^{1/2}}$,
4. $t_{k,n} = \alpha_k C_k u_{k,n}$,
5. $s_n = \sum_{k=0}^{K-1} t_{k,n}$,
6. $\bar{s}_n = 1/s_n$,
7. $\gamma_{k,n} = t_{k,n} \cdot \bar{s}_n$.

Operations 1, 2, and 3 presented above are prone to bottlenecks due to matrix products, exponential functions, and determinant operations. Even when the number of data n is large, the computational cost of C_k is not significant because C_k is independent of n. Thus, we need to reduce the computational cost of Operations 1 and 2. The most significant operation in the E-step is listed in Eq. (2).

The aim of this work is to reduce the operations in Eq. (2). The exponential function in operation 2 of the E-step can be approximated by taking into account Eq. (1). In the next section, we propose a technique for approximating the exponential function.

3. EM Algorithm with Look-Up-Table-Based Exponential Function

In this section, we propose an approximation method for the exponential function used by the EM algorithm that estimates the GMM parameters. The approximation method reduces computational cost by taking Eq. (1) into account.

The idea of transforming non-linear function into look-up table (LUT) was used in various studies. For example, Fiori used the idea for neural network [8]–[10].

This work approximates the exponential function by using a bit-shift and a LUT. First, the exponential function is represented by powers of two. Then, the powers of two are approximated using a bit-shift and a LUT. Furthermore, to simplify the implementation, the LUT is scaled by a constant coefficient.

3.1 Look-up-Table-Based Exponential Function

The exponential function can be expressed as a power of two as

$$\exp(-z) = 2^{-z\cdot\log_2 e}, \tag{5}$$

where z is a variable and e is Napier's number. The right-hand side of Eq. (5) can be separated into two components

$$2^{-z\cdot\log_2 e} = 2^{-(\lfloor z\cdot\log_2 e\rfloor+\beta)} = 2^{-\lfloor z\cdot\log_2 e\rfloor} \cdot 2^{-\beta}, \tag{6}$$

where $\lfloor x\rfloor$ represents the integer part of x and $\beta = z\cdot\log_2 e - \lfloor z\cdot\log_2 e\rfloor$, i.e., the fractional part of $z\cdot\log_2 e$.

From Eqs. (5) and (6) we conclude that in the binary digit system, exponential function can be realized by a bit-shift of $2^{-\beta}$. However, computing $2^{-\beta}$ still remains an issue. In particular, the power of two can be computed using the Taylor series expansion. Since the later approach is much simpler than the former, we propose introducing a LUT that contains approximated values of $2^{-\beta}$.

To construct a LUT with a finite number of contents, we use a bit string $\hat{\beta}$ that is the L-bit approximation of β. By using the bit string $\hat{\beta}$, the function $2^{-\beta}$ can be approximated by:

$$2^{-\beta} \approx 2^{-\hat{\beta}} = 2^{-\sum_{i=1}^{L} 2^{-i}\cdot\hat{\beta}^{[i]}}, \tag{7}$$

where, $\hat{\beta}^{[i]} \in \{0, 1\}$ is the i-th bit of $\hat{\beta}$. Note that the most significant bit and least significant bit of $\hat{\beta}$ are $\hat{\beta}^{[1]}$ and $\hat{\beta}^{[L]}$, respectively. By applying Eq. (7), the entries of the LUT $T[\hat{\beta}]$ are obtained by

$$T[\hat{\beta}] = 2^{-\sum_{i=1}^{L} 2^{-i}\cdot\hat{\beta}^{[i]}}. \tag{8}$$

Since LUT does not depend on any data, it can be computed off-line. In Table 1, we present the values of the LUT for $L = 2$.

Using Eqs. (6) and (8), the exponential function is approximated by

Table 1 Values of the LUT $T[\hat{\beta}]$ for $L = 2$. The symbol $(\,)_2$ indicates that the bit string $\hat{\beta}$ is represented in binary form.

$\hat{\beta}$	$T[\hat{\beta}]$
$(00)_2$	$1.000000\cdots$
$(01)_2$	$0.840896\cdots$
$(10)_2$	$0.707107\cdots$
$(11)_2$	$0.594604\cdots$

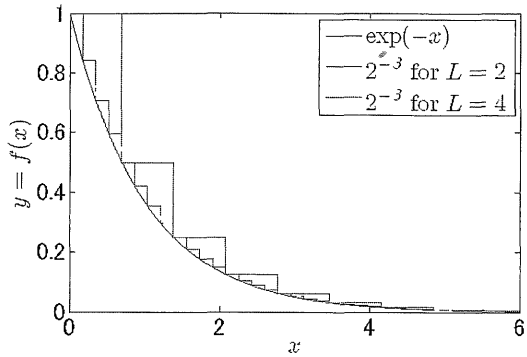


Fig. 1 Exponential function and its approximations. The horizontal axis indicates the input x and the vertical axis indicates the exponential function and its approximations. The solid, bold, and dotted lines represent $\exp(-x)$, $2^{-\hat{\beta}}$ for $L = 2$ and $2^{-\hat{\beta}}$ for $L = 4$, respectively.

$$\exp(-z) \approx 2^{-\lfloor z \log_2 e \rfloor} \cdot T[\hat{\beta}]. \tag{9}$$

Equation (9) indicates that the exponential function can be computed by shifting the bits of an entry of the LUT. Figure 1 shows plots of the exponential function and its approximations using LUTs for $L = 2$ and $L = 4$.

3.2 Scale Adjustment for Look-up Table

According to Eq. (8), the range of the entries of the LUT is $(0.5, 1.0]$. Two issues must be considered with respect to this range. One is the complexity of a floating-point representation, such as the IEEE 754 floating-point format. In this format, values within the range $(0.5, 1.0)$ change the mantissa part, while the value 1.0 changes the exponential part. Therefore, the value 1.0 requires exception handling. The second issue is that the values of the LUT require an additional bit because in fixed-point representation, value 1.0 requires one additional bit than the values in the range $(0.5, 1.0)$. Therefore, the ranges $[0.5, 1.0)$ and $(0.5, 1.0)$ are preferable to range $(0.5, 1.0]$.

For transforming the values of LUT into preferable range, let us consider a Gaussian distribution scaled by a constant factor. In the E-step, scaling does not affect the result of $\gamma_{k,n}$. Equation (1) can be represented by

$$\gamma_{k,n} = \frac{\alpha_k \delta \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=0}^{K-1} \alpha_j \delta \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}, \tag{10}$$

where δ is a non-zero constant. It is clear that the result of Eq. (10) is identical to that of Eq. (1). Hence, for Gaussian distributions, the scaling operation has no affect on the results of the E-step.

Table 2 Values of the LUT $\hat{T}[\hat{\beta}]$ obtained by the weighted-average method for $L = 2$ and $a_0 = a_1 = 0.5$.

$\hat{\beta}$	$\hat{T}[\hat{\beta}]$
$(00)_2$	$0.920448\cdots$
$(01)_2$	$0.774002\cdots$
$(10)_2$	$0.650855\cdots$
$(11)_2$	$0.547302\cdots$

Scaling can be used to modify the look-up-table-based exponential function. The scaled Gaussian distribution is approximated by

$$\begin{aligned}\delta\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) &= \delta C_k \exp\{-y_{k,n}\} \\ &\approx \delta C_k \cdot 2^{-\lfloor y_{k,n} \log_2 e \rfloor} T[\hat{\beta}_{k,n}] \\ &= C_k \cdot 2^{-\lfloor y_{k,n} \log_2 e \rfloor} \hat{T}[\hat{\beta}_{k,n}], \end{aligned} \tag{11}$$

where $\hat{T}[\hat{\beta}_{k,n}] = \delta T[\hat{\beta}_{k,n}]$. Hence, the LUT can be scaled by δ.

3.3 Scaling Using the Weighted Average of LUT Entries

Next, we address the problem of identifying δ that scales the LUT within the desired range. In order to obtain an appropriate constant, we consider generating the LUT entries by linearly combining neighboring entries. The next entry of $T[\hat{\beta}]$ is represented by $T[\hat{\beta} + \hat{\beta}^{[L]}]$. Note that $\hat{\beta}^{[L]}$ is the least significant bit of the bit string $\hat{\beta}$. Using a weighted average approach, the scaled value $\hat{T}[\hat{\beta}]$ is calculated by

$$\begin{aligned}\hat{T}[\hat{\beta}] &= a_0 T[\hat{\beta}] + a_1 T[\hat{\beta} + \hat{\beta}^{[L]}] \\ &= a_0 \cdot 2^{-\sum_{i=1}^{L} 2^{-i} \cdot \hat{\beta}^{[i]}} + a_1 \cdot 2^{-\{(\sum_{i=1}^{L} 2^{-i} \cdot \hat{\beta}^{[i]}) + 2^{-L}\}} \\ &= 2^{-\sum_{i=1}^{L} 2^{-i} \cdot \hat{\beta}^{[i]}} \left(a_0 + a_1 \cdot 2^{-2^{-L}}\right) \\ &= T[\hat{\beta}] \left(a_0 + a_1 \cdot 2^{-2^{-L}}\right), \end{aligned} \tag{12}$$

where a_0 and a_1 are weights, $a_0 + a_1 = 1$ and $a_0, a_1 \geq 0$. Equation (12) converts the range of the LUT into the range $[2^{-\sum_{i=1}^{L} 2^{-i}} \cdot (a_0 + a_1 \cdot 2^{-2^{-L}}), a_0 + a_1 \cdot 2^{-2^{-L}}]$, where $\sum_{i=1}^{L} 2^{-i}$ indicates that all digits of $\hat{\beta}$ are equal to one, i.e., $\hat{\beta} = (11\cdots11)_2$. Note that because $\left(a_0 + a_1 \cdot 2^{-2^{-L}}\right)$ is independent of $\hat{\beta}$, it becomes a constant. Thus, the constant $\left(a_0 + a_1 \cdot 2^{-2^{-L}}\right)$ can be used as the scaling factor δ. In Table 2, we present the values of the LUT for $L = 2$ and $a_0 = a_1 = 0.5$.

4. Performance Evaluation

To validate the effectiveness of the proposed method, we generate simulation results and evaluate the precision of the estimated parameters and computational time of the EM algorithm.

Random numbers from two-component mixture of Gaussian distributions are generated as follows:

Table 3 Parameters of a mixture Gaussian distribution for generating normal random numbers, where α is the mixture ratio, μ is the mean, and σ is the variance. The step sizes of μ_1 and Σ_1 are 0.2.

Params.	Dist. 0	Dist. 1
α	0.5	0.5
μ	0.0	From 1.0 to 5.0
Σ	1.0	From 0.2 to 2.0

Table 4 Initial values used in the EM algorithm.

Params.	Dist. 0	Dist. 1
α^{init}	0.5	0.5
μ^{init}	-0.5	μ_1+0.5
Σ^{init}	1.0	1.0

- The Mersenne twister method is used to generate uniform random numbers [11].
- The Box-Muller method is used to generate Gaussian random numbers [12].
- The number of data points is 100,000.

The parameters of the distributions are summarized in Table 3.

Next, we use the EM algorithm to estimate the parameters. We apply the E-step and M-step 30 times in a loop. The initial parameters used for the EM algorithm are summarized in Table 4. Note that initial mean values are different from the original values.

The exponential function in the EM algorithm was implemented using the proposed method, the Taylor expansion, and the exp() function in the standard C library. Moreover, during the simulation, we adopted the IEEE 754 double precision format.

The evaluation program was implemented using the C programming language. The specifications of the software development environment and hardware environment used are:

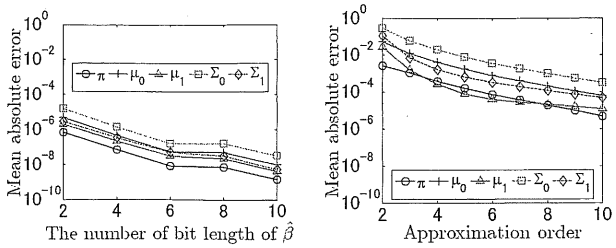
- OS: MS Windows 7 (64 bit edition)
- Development environment: MS Visual Studio 2010
- SDK: MS Windows SDK for Windows 7
- Optimization flags: /Ox and /arch:SSE2
- CPU: Intel Core 2 duo E8500 (3.16 GHz)
- Memory: 8 GB dual channel DDR2 SDRAM (PC2-6400)

The simulation results were evaluated on a single core.

4.1 Precision of Parameter Estimation

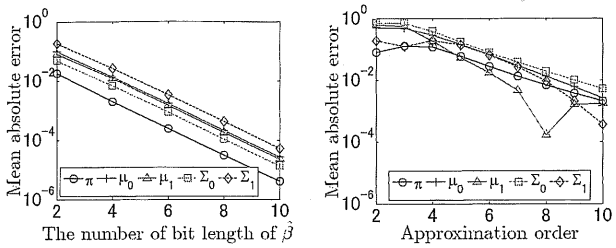
In this section, we present representative simulation results. The simulation results were evaluated in terms of the mean absolute error. The results obtained by the exp() function are considered as the true values.

The first simulation was performed with $\mu_1 = 5.0$ and $\Sigma_1 = 0.2$. The second was performed with $\mu_1 = 1.0$ and $\Sigma_1 = 2.0$. In Fig. 2, we present simulation results obtained for $\mu_1 = 5.0$ and $\Sigma_1 = 0.2$. The results obtained from the proposed method are more precise than those obtained from the Taylor expansion. Figure 3 shows the results for $\mu_1 = 5.0$ and $\Sigma_1 = 0.2$. For this parameter combination, the two distributions are close and overlap. In this case, it is difficult to estimate the parameters. The results obtained by the proposed method are comparable to those of the Taylor expansion.



(a) Proposed method. (b) Taylor expansion.

Fig. 2 Simulation results obtained for $\mu_1 = 5.0$ and $\Sigma_1 = 0.2$. The vertical axis represents the mean absolute error. The horizontal axis in (a) represents the bit-length of $\hat{\beta}$, while the horizontal axis in (b) is approximation order. The averages obtained by exp() were $\pi_0 = 0.500, \pi_1 = 0.500, \mu_0 = 6.204 \times 10^{-4}, \mu_1 = 5.000, \Sigma_0 = 1.000$, and $\Sigma_1 = 0.200$.



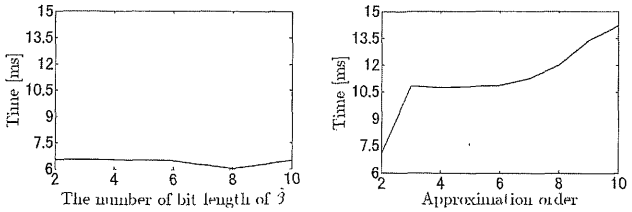
(a) Proposed method. (b) Taylor expansion.

Fig. 3 Simulation results obtained for $\mu_1 = 1.0$ and $\Sigma_1 = 2.0$. The vertical axis represents the mean absolute error. The horizontal axis in (a) represents the bit-length of $\hat{\beta}$, while the horizontal axis in (b) is the approximation order. The averages obtained by exp() were $\pi_0 = 0.549, \pi_1 = 0.451, \mu_0 = 2.046 \times 10^{-3}, \mu_1 = 1.106, \Sigma_0 = 1.045$, and $\Sigma_1 = 1.934$.

4.2 Computational Speed

Next, let us discuss the computational time of E-step using the proposed method, Taylor expansion, and the standard exp() function. We recorded the computational times of 30 loops of E-step and M-step by obtaining 100 measurements using 100 different random number seeds, and computed the average time.

The computational time of the E-step using the standard exp() function and the direct implementation of M-step was 14.26 [ms] and 1.26 [ms], respectively. These results demonstrate that for signal variables, the computational time of E-step is dominant in the EM algorithm. Figure 4 shows the computational time of the E-step using the proposed method and the Taylor expansion. The proposed method took less than 6.54 [ms], while the Taylor expansion took over 7.01 [ms]. Specifically, the Taylor expansion with higher-order approximation required over 10.73 [ms]. By comparing the proposed method with 10-bits and the Taylor



(a) Proposed method. (b) Taylor expansion.

Fig. 4 Computational time results. The vertical axis shows the computational time. The horizontal axis in (a) is the bit-length of $\hat{\beta}$, and the horizontal axis in (b) is the approximation order.

expansion with the 10-th order approximation, we observed that the proposed method reduces the computational time by 45.62% while achieving more precise estimation results. Similarly, by comparing the proposed method with the standard exp() function, we observe that the proposed method reduces computational time by more than 45.86%.

5. Conclusion

In this paper, we proposed a low-cost implementation of the exponential function for the EM algorithm used to estimate the parameters in a GMM. First, we converted the exponential function in the E-step into a power of two. Then, we computed the exponential function using a LUT. To reduce computational complexity, the LUT was scaled using a weighted-average technique. Through simulation results, we demonstrated that the mean absolute error and computational time were reduced compared to the Taylor expansion. The proposed method was also shown to maintain high precision.

In future work, we will evaluate the proposed method for estimating parameters using other distributions. The idea of this work can be used for parameter estimation for a mixture of exponential families. For example, Hidden Markov model (HMM) also involves the computation of an exponential function. Thus, we will evaluate if the proposed method can be used to reduce the computational cost of HMMs.

References

[1] M. Fujimoto and Y. Riki, "Robust speech recognition in additive and channel noise environments using gmm and em algorithm," IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. Proc. (ICASSP'04). vol.1, pp.I – 941–4, 2004.

[2] O. Rotem, H. Greenspan, and J. Goldberger, "Combining region and edge cues for image segmentation in a probabilistic gaussian mixture framework," IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07. pp.1–8, 2007.

[3] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," J. Royal Statistical Society, Series B, vol.39, pp.1–22.

[4] L. Lu and H.J. Zhang, "Speaker change detection and tracking in real-time news broadcasting analysis," Proc. 10th ACM International Conference on Multimedia, MULTIMEDIA'02, pp.602–610, ACM, 2002.

[5] H. Watanabe and S. Muramatsu, "Fast algorithm and efficient implementation of GMM-based pattern classifiers," J. Signal Process., vol.63, no.1, pp.107–116, 2011.

[6] H. Watanabe, S. Muramatsu, and H. Kikuchi, "Multiplierless refinement scheme for interval calculation of GMM-based classification," Proc. 2009 APSIPA Annual Summit and Conference, pp.282–285, 2009.

[7] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

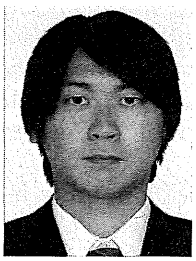
[8] S. Fiori, "Hybrid independent component analysis by adaptive lut activation function neurons," Neural Netw., vol.15, no.1, pp.85–94, 2002.

[9] S. Fiori, "Generation of pseudorandom numbers with arbitrary distribution by learnable look-up-table-type neural networks," IEEE International Joint Conference on Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). pp.1787–1792, June 2008.

[10] S. Fiori, "Fast statistical regression in presence of a dominant independent variable," Neural Computing and Applications, pp.1–12, 2012.

[11] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator," Trans. Modeling and Computer Simulation, vol.8, no.1, pp.3–30, 1998.

[12] G.E.P. Box and M.E. Muller, "A note on the generation of random normal deviates," Annals of Mathematical Statistics, vol:29, no.2, pp.610–611, 1958.



Hidenori Watanabe received B.E. and M.E. degrees in electrical engineering from Niigata University in 2008 and 2010, respectively. He is currently a Ph.D. candidate at Niigata University. His research interests are in digital signal processing and implementation.

Shogo Muramatsu received the B.E., M.E., and Ph.D. degrees from Tokyo Metropolitan University, Tokyo, Japan, in 1993, 1995, and 1998, respectively. From 1997 to 1999, he worked with Tokyo Metropolitan University. In 1999, he joined Niigata University, Niigata, Japan. From 2003 to 2004, he was a Visiting Researcher with the University of Florence, Florence, Italy. He is currently an Associate Professor with the Department of Electrical and Electronics Engineering, Faculty of Engineering, Niigata University. His research interests include multidimensional signal processing, multirate systems, image processing, video analysis and embedded vision systems. Dr. Muramatsu is a member of the Institute of Image Information and Television Engineers of Japan.