FAST ALGORITHM FOR GMM-BASED PATTERN CLASSIFIER

Shogo MURAMATSU and Hidenori WATANABE

Department of Electrical and Electronic Eng., Niigata University
8050 Ikarashi 2-no-cho, Nishi-ku, Niigata, 950-2181, Japan
phone: + (81) 25-262-6746, fax: + (81) 25-262-7010, email: shogo@eng.niigata-u.ac.jp

ABSTRACT

This work proposes a fast decision algorithm in pattern classification based on Gaussian mixture models (GMM). Statistical pattern classification problems often meet a situation that comparison between probabilities is obvious and involve redundant computations. When GMM is adopted for the probability model, the exponential function should be evaluated. This work firstly reduces the exponential computations to simple and rough interval calculations. The exponential function is realized by scaling and multiplication with powers of two so that the decision is efficiently realized. For finer decision, a refinement process is also proposed. In order to verify the significance, experimental results on TI DM6437 EVM board are shown through the application to a skin-color extraction problem. It is verified that the classification was almost completed without any refinement process and the refinement process can proceed the residual decisions.

Index Terms— Pattern classification, Gaussian mixture model, Bayesian decision, Efficient implementation, Skin-color extraction

1. INTRODUCTION

Pattern classification is a technique that automatically identifies an observation target based on data acquired by sensors and meets a wide range of applications such as product inspection, medial diagnosis, intelligent vehicles and surveillance systems [1]. Some classifiers assume probability models for observed data, and identifies the class according to Bayesian decision rule.

Gaussian distribution is a popular model and easy to handle for the mathematical beauty. The distribution is, however, not suitable for modeling a multi-modal distribution, and the application is limited. This problem can be solved by adopting a Gaussian mixture model (GMM), which is a linear combination of multiple Gaussian distributions. GMM is successfully used for several applications such as skin-color extraction and speech recognition [2–4]. The parameter estimation problem has also been studied by several researchers [5–7]. One of the disadvantages of GMM is the computational complexity. Unlike Gaussian distributions, the comparison among GMMs cannot exclude the exponential function even if the logarithm is taken. Increasing the computational complexities becomes obstacle to the applications requiring high speed and low power consumption. For example, wireless sensor networks demand low power classifiers in their sensor nodes.

In the article [8], the exponential function is computed with a piecewise linear approximation. Although the approximation is simply computed, wrong classification is possibly made when the precision is low. On the other hand, higher precision requires more operations and memories. It is true for other approaches such as Maclaurin expansion and CORDIC [9].

This work proposes to operate the exponential function by introducing simple interval calculations and an adaptive control of computational precision. In a typical classification problem, precise operations are required only for subtle cases and most are obvious. The proposed method replaces the exponential function to scaling and multiplication with powers of two, and the decision if the comparison is obvious is efficiently calculated. A refinement process is also proposed for subtle cases, where the computational accuracy is successively refined.

2. GMM-BASED CLASSIFICATION

This section reviews Bayesian decision rule and GMM.

2.1. Bayesian decision rule

Bayesian decision rule enables us to identify the class of an observed datum or feature vector $\mathbf{x}$ provided possible classes are known a priori. Given a feature vector $\mathbf{x}$, Bayesian decision scheme compares the posterior probabilities of every class, i.e. $P[c = C_k|\mathbf{x}]$, and then select the class of the highest probability. This decision is known to give the smallest error rate [1, 7]. Since the Bayesian theorem tells us $P[c|\mathbf{x}] \propto p(\mathbf{x}|c)P[c]$, the Bayesian decision can be reduced to the evaluation with the following discriminant function:

$$f_{k,\ell}(\mathbf{x}) = p(\mathbf{x}|C_k)P[C_k] - p(\mathbf{x}|C_\ell)P[C_\ell]. \quad (1)$$

If $f_{k,\ell}(\mathbf{x}) > 0$, then Class C_ℓ is dropped from the candidates. If $f_{k,\ell}(\mathbf{x}) < 0$, then Class C_k remains nominated.

2.2. Gaussian mixture model (GMM)

Equation (1) requires us to know the conditional density $p(\mathbf{x}|c)$ a priori. This work assumes GMM for $p(\mathbf{x}|c)$ [7]. Let $\mathcal{N}(\mathbf{x}|\mu,\Sigma)$ be the density of Gaussian distribution. $\mathcal{N}(\mathbf{x}|\mu,\Sigma)$ is defined by

$$\mathcal{N}(\mathbf{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right\}, \quad (2)$$

where D is the number of variables, μ and Σ are a $D \times 1$ mean vector and $D \times D$ covariance matrix, respectively. Then, the density of GMM, $\mathcal{M}(\mathbf{x}|\Theta)$, is represented by

$$\mathcal{M}(\mathbf{x}|\Theta) = \sum_{n=0}^{N-1} \alpha_n \mathcal{N}(\mathbf{x}|\mu_n,\Sigma_n), \quad (3)$$

where Θ is a set of parameters, that is $\Theta = \{\{\alpha_n\},\{\mu_n\},\{\Sigma_n\}\}$, N is the number of Gaussian distributions and α_n is the mixture ratio of the n-th distribution. The mixture ratios satisfy the conditions $\sum_{n=0}^{N-1}\alpha_n = 1$ and $0 \le \alpha_n \le 1$.

The authors would like to acknowledge the support from the Texas Instruments University program.

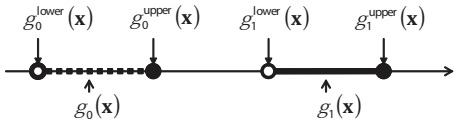


Fig. 1. Case that the interval is used for classification

2.3. Discriminant function for GMM

Since the probability of $\mathbf{x}$ given c, i.e. $p(\mathbf{x}|c)$, is assumed to be GMM in Eq. (3), Eq. (1) is represented by

$$f_{k,\ell}(\mathbf{x}) = g_k(\mathbf{x}) - g_\ell(\mathbf{x}), \tag{4}$$

$$g_k(\mathbf{x}) = \sum_{n=0}^{N_k-1} K_{k,n} \exp(-z_{k,n}(\mathbf{x})), \tag{5}$$

where

$$z_{k,n}(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mu_{k,n})^T \Sigma_{k,n}^{-1}(\mathbf{x}-\mu_{k,n}), \tag{6}$$

$$K_{k,n} = \frac{P[C_k]\alpha_{k,n}}{(2\pi)^{\frac{D}{2}}|\Sigma_{k,n}|^{\frac{1}{2}}}, \tag{7}$$

where $\{\alpha_{k,n}\}$, $\{\mu_{k,n}\}$ and $\{\Sigma_{k,n}\}$ are parameter sets of GMM for Class C_k [8]. Since the covariance matrix $\Sigma_{k,n}$ is positive definite, function $z_{k,n}(\mathbf{x})$ obtained by the quadratic form in Eq. (6) is guaranteed to be non-negative. As well, the conditions on $\alpha_{k,n}$ guarantee for constant $K_{k,n}$ to be non-negative. Therefore, Eq. (5) must be non-negative. For the sake of convenience, function $z_{k,n}(\mathbf{x})$ is simply represented by variable $z_{k,n}$ in the following discussion.

3. CLASSIFICATION WITH INTERVAL COMPUTATION

On the assumption that a feature vector $\mathbf{x}$ is drawn from a Gaussian distribution, the Bayesian decision is achieved by comparing scaled exponential functions, i.e. $K\exp(-z)$, where K and z are non-negative constant and variable. Usually, the comparison is reduced by taking their logarithms, i.e. $\ln K\exp(-z) = \ln K - z$, where $\ln K$ is a constant. Note that the classifier does not require any exponential function. On the other hand, it is not true for GMM.

In this section, we propose a simple interval calculations and an adaptive control of computational precision for the exponential operatons in GMM. The following facts are used in our proposal.

- Required result is only the sign of Eq. (4).
- Constant $K_{k,n}$ and variable $z_{k,n}$ in Eq. (5) are all non-negative.

3.1. Initial decision process

First of all, we give an inequality which represents an interval covering the function in Eq. (5) with integer powers of two. From the fact that $\exp(-z) = 2^{-z\log_2 e}$ and $z \geq 0$, we have the relation

$$2^{-(\lfloor z\log_2 e\rfloor+1)} < \exp(-z) \leq 2^{-\lfloor z\log_2 e\rfloor}, \tag{8}$$

where $\lfloor x\rfloor$ is the integer part of x. Furthermore, Eq. (8) leads

$$\sum_{n=0}^{N-1} K_n 2^{-(\lfloor z_n\log_2 e\rfloor+1)} < \sum_{n=0}^{N-1} K_n \exp(-z_n) \leq \sum_{n=0}^{N-1} K_n 2^{-\lfloor z_n\log_2 e\rfloor}. \tag{9}$$

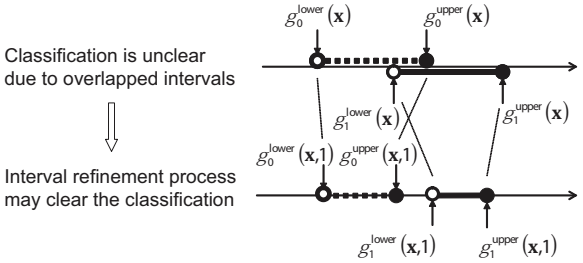


Fig. 2. Case that the interval computation is insufficient (top). An expected effect of the refinement process (bottom).

The lower and upper bound of this interval is calculated only by scaling constant K_n with an integer power of two and accumulating the results. If K_n is in a fixed-point representation, this operation is executed by the right bit-shift. If it is in a floating-point one, the index decrement realizes the computation. Figure 1 illustrates a case that the class C_0 is clearly delated from nominations with the interval computation, where we define

$$g_k^{\text{upper}}(\mathbf{x}) = \sum_{n=0}^{N_k-1} K_{k,n} 2^{-\lfloor z_{k,n}\log_2 e\rfloor}, \tag{10}$$

$$g_k^{\text{lower}}(\mathbf{x}) = \sum_{n=0}^{N_k-1} K_{k,n} 2^{-(\lfloor z_{k,n}\log_2 e\rfloor+1)} = \frac{1}{2} g_k^{\text{upper}}(\mathbf{x}). \tag{11}$$

Note that the lower bound is half the upper one and this fact is important in terms of computational cost. Each interval guarantees to include the true value of $g_k(\mathbf{x})$. The above decision process makes it possible to avoid precise calculations.

3.2. Refinement process

When the probability of C_k given $\mathbf{x}$ close to another, the initial decision process confuses. In the followings, we propose a refinement process for successively improving the computational precision and narrowing the interval. Figure 2 illustrates a case that the decision is unclear from the interval and the refinement process improves each interval which includes the true value $g_k(\mathbf{x})$.

Let us begin with the relation

$$\exp(-z) = 2^{-\lfloor z\log_2 e\rfloor} 2^{-\beta} = 2^{-(\lfloor z\log_2 e\rfloor+1)} 2^{(1-\beta)}, \tag{12}$$

where β is the fractional part of $z\log_2 e$ and in the range $0 \leq \beta < 1$. Representing the i-th fractional bit of β as $\beta^{[i]} \in \{0,1\}$, we have

$$\beta = \sum_{i=1}^{L} \beta^{[i]} 2^{-i}, \tag{13}$$

where L denotes the number of fractional bits. Furthermore, defining $T[i] = 2^{-2^{-i}}$, we obtain the following expressions:

$$2^{-\beta} = 2^{-\sum_{i=1}^{L}\beta^{[i]}2^{-i}} = \prod_{i=1}^{L} 2^{-\beta^{[i]}2^{-i}} = \prod_{i=1}^{L} T[i]^{\beta^{[i]}}, \tag{14}$$

$$2^{(1-\beta)} = 2^{2^{-L}} \prod_{i=1}^{L} 2^{\bar{\beta}^{[i]}2^{-i}} = T[L]^{-1} \prod_{i=1}^{L} T[i]^{-\bar{\beta}^{[i]}}, \tag{15}$$

where $\bar{\beta}^{[i]}$ is the bit inverse of $\beta^{[i]}$.

Then, let us consider an update process for refining the interval. According to Eq. (15), the upper and lower bound in Eqs. (10) and (11) are improved by using the following bounds for $i \geq 1$:

$$g_k^{\text{upper}}(\mathbf{x}, i) = \sum_{n=0}^{N_k-1} h_{k,n}^{\text{upper}}(\mathbf{x}, i), \tag{16}$$

$$g_k^{\text{lower}}(\mathbf{x}, i) = \sum_{n=0}^{N_k-1} h_{k,n}^{\text{lower}}(\mathbf{x}, i), \tag{17}$$

where

$$h_{k,n}^{\text{upper}}(\mathbf{x}, i) = \begin{cases} K_{k,n} 2^{-\lfloor z_{k,n} \log_2 e \rfloor}, & i = 0 \\ K_{k,n} 2^{-\lfloor z_{k,n} \log_2 e \rfloor} \prod_{j=1}^{i} T[j]^{\beta_{k,n}^{[j]}}, & i \geq 1 \end{cases} \tag{18}$$

$$h_{k,n}^{\text{lower}}(\mathbf{x}, i) = \begin{cases} K_{k,n} 2^{-(\lfloor z_{k,n} \log_2 e \rfloor + 1)}, & i = 0 \\ K_{k,n} 2^{-(\lfloor z_{k,n} \log_2 e \rfloor + 1)} \prod_{j=1}^{i} T[j]^{-\overline{\beta}_{k,n}^{[j]}}, & i \geq 1 \end{cases} \tag{19}$$

where $\beta_{k,n}^{[i]}$ denotes the i-th fractional bit of $z_{k,n} \log_2 e$.

For $i \geq 1$, we have the relations

$$h_{k,n}^{\text{upper}}(\mathbf{x}, i) = h_{k,n}^{\text{upper}}(\mathbf{x}, i-1) T[i]^{\beta_{k,n}^{[i]}}, \tag{20}$$

$$h_{k,n}^{\text{lower}}(\mathbf{x}, i) = h_{k,n}^{\text{lower}}(\mathbf{x}, i-1) T[i]^{-\overline{\beta}_{k,n}^{[i]}}. \tag{21}$$

Since the inequalities

$$K_{k,n} \exp(-z_{k,n}) \leq h_{k,n}^{\text{upper}}(\mathbf{x}, i) \leq h_{k,n}^{\text{upper}}(\mathbf{x}, i-1), \tag{22}$$

$$h_{k,n}^{\text{lower}}(\mathbf{x}, i-1) \leq h_{k,n}^{\text{lower}}(\mathbf{x}, i) \leq K_{k,n} \exp(-z_{k,n}), \tag{23}$$

hold, Eqs. (16) and (17) become closer to the true evaluation $g_k(\mathbf{x})$ as i increases. Consequently, the intervals are refined as shown in Fig. 2. Although both of Eqs. (20) and (21) include multiplications, one refinement process requires only either of $T[i]$ or $T[i]^{-1}$ since $\overline{\beta}_{k,n}^{[i]}$ is the bit inverse of $\beta_{k,n}^{[i]}$. Both of $T[i]$ and $T[i]^{-1}$ can be calculated before and stored in look up tables (LUTs).

3.3. Termination

The refined bounds approach to a true value on a principal by continuing the update operations. However, in an actual situation, computations must be completed in a finite bit accuracy. As an option, we provide a termination process. Let L be the maximum number of refinement process. Then, the average value of the bounds may be a good approximation of $g_k(\mathbf{x})$. The average is obtained by

$$g_k^{\text{pseudo}}(\mathbf{x}) = \frac{1}{2} \left\{ g_k^{\text{lower}}(\mathbf{x}, L) \cdot T[L]^{-1} + g_k^{\text{upper}}(\mathbf{x}, L) \right\}. \tag{24}$$

Then, $g_k(\mathbf{x})$ in Eq. (5) can be replaced by $g_k^{\text{pseudo}}(\mathbf{x})$ for a pseudo decision. Note that $T[L]^{-1} \approx 1$ for a large L.

3.4. Proposed algorithm

Let us summerize the procedures described from 3.1 to 3.3, where we represent $2^{-i}x$ as $x \gg i$ and assume a two class case.

Step 1 Calculate Eqs. (6) and (7) for all $k \in \{0,1\}$ and $n = 0, 1, \cdots, N_k - 1$. Then, obtain

$$h_{k,n}^{\text{upper}} = K_{k,n} \gg \lfloor z_{k,n} \log_2 e \rfloor,$$
$$h_{k,n}^{\text{lower}} = h_{k,n}^{\text{upper}} \gg 1,$$
$$\beta_{k,n} = \text{frac}(z_{k,n} \log_2 e),$$

and set $i = 0$, where $\text{frac}(x)$ is the fractional part of x.

Step 2 Calculate the following equations for all $k \in \{0,1\}$:

$$g_k^{\text{upper}} = \sum_{n=0}^{N_k-1} h_{k,n}^{\text{upper}}$$

$$g_k^{\text{lower}} = \begin{cases} g_k^{\text{upper}} \gg 1, & i = 0 \\ \sum_{n=0}^{N_k-1} h_{k,n}^{\text{lower}}, & i > 0 \end{cases}.$$

Step 3 If $g_0^{\text{upper}} \leq g_1^{\text{lower}}$, then decide on Class C_1 and quit. If $g_1^{\text{upper}} \leq g_0^{\text{lower}}$ then decide on Class C_0 and quit.

Step 4 Increment i as $i \leftarrow i+1$. If i exceeds the upper limit L, then go to Step 6.

Step 5 Update the interval for all $k \in \{0,1\}$ and $n = 0, 1, \cdots, N_k - 1$. If $\beta_{k,n}^{[i]} = 1$, then apply

$$h_{k,n}^{\text{upper}} \overset{\text{update}}{\longleftarrow} h_{k,n}^{\text{upper}} \cdot T[i].$$

Otherwise, apply

$$h_{k,n}^{\text{lower}} \overset{\text{update}}{\longleftarrow} h_{k,n}^{\text{lower}} \cdot T[i]^{-1}.$$

Return to Step 2.

Step 6 Decide the class with the pseudo-function

$$g_k^{\text{pseudo}} = \frac{1}{2} \left\{ g_k^{\text{lower}} \cdot T[L]^{-1} + g_k^{\text{upper}} \right\}.$$

If $g_0^{\text{pseudo}} < g_1^{\text{pseudo}}$, then decide on Class C_1. Otherwise, decide on Class C_0. Finally, terminate the process.

4. PERFORMANCE EVALUATION

In this section, we verify the significance of the initial decision, and then evaluate the computational cost of the refinement process.

4.1. Simulation of skin-color extraction

More the initial decision completes the classification, the lower the computational cost becomes. Let us show some simulation results of the application to skin-color extraction. The followings summerize the simulation procedure:

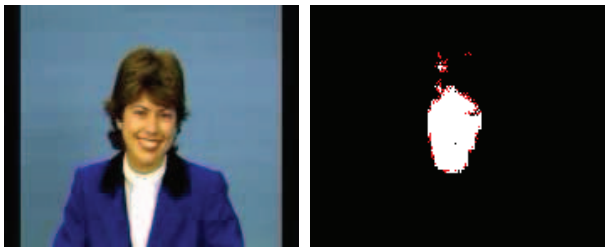
- Claire, a sequence of size 144×176 in YUV format, is used.
- 2×1 feature vector $\mathbf{x}$ is defined by the U and V component.
- GMMs are assumed to the skin and non-skin class.
- The first frame is used to train the classifier[1].
- The 494-th frame is used as an observed picture.

Figure 3 shows an initial decision result, where the number of distributions are assumed to be four. It is observed that the initial decision can almost complete the classification. The ratios of the completion for different numbers of distributions resulted in

- 99.641% for 2 distributions ($N_0 = N_1 = 2$),
- 99.617% for 3 distributions ($N_0 = N_1 = 3$),
- 99.515% for 4 distributions ($N_0 = N_1 = 4$).

These simulation results tell us that any precise evaluation is not necessary for the exponential function for over 99.5% of pixels. As well, we verified that the number of updates in which the decisions completed in the double precision were 7, 6 and 11 updates for 2, 3 and 4 distributions, respectively. LUTs for $T[i]$ and $T[i]^{-1}$ may typically be not expensive since the maximum number of updates L determines the number of entries.

[1] EM algorithm was used [7, 10].



(a) Observation (b) Result with $N_0 = N_1 = 4$

Fig. 3. Initial decision results, where the white, black and red region express the skin, non-skin and undecided region.

Table 1. Average computational time per feature vector for evaluating Eq. (4) on DM6437 EVM board, where $D = 2$ and $N_0 = N_1 = 2$. The quadratic form in Eq. (6) consumes around 5.5μs in floating-point calculations of single precision, which is excluded below.

Type of constant $K_{k,n}$	Fixed-point	Float.-point
Cons. (w. `expf()`)	-	12.44μs
Cons. (w. `exp()`)	-	26.07μs
Props. ($L = 0$, w.o. Refine.)	1.47μs	2.21μs
Props. ($L = 8$, w. Refine.)	1.55μs	2.26μs

4.2. Computational cost

Let $r[i]$ be the ratio of progression to the i-th refinement process. Note that relation $0 \le r[i+1] \le r[i] \le 1$ holds. Denoting τ_{init} and τ_{refine} as the time required for the initial decision per feature vector in Steps 1, 2 and 3 and the time for a refinement process in Steps 5, 2 and 3, respectively, we obtain the average decision time per feature vector with L refinement processes as

$$\tau_{\text{ave}} = \tau_{\text{init}} + \sum_{i=1}^{L} r[i]\,\tau_{\text{refine}}, \tag{25}$$

where the termination in Step 6 is omitted for simplicity since it is optional and less expensive than the refinement process. If τ_{init} is much less than the time τ required with some existing exponential implementation, and if the progression ratios $r[i]$ are close to zero, then some acceleration can be expected since $\tau_{\text{ave}} \approx \tau_{\text{init}} < \tau$.

For evaluating the computational cost, we implemented the skin-color extraction on TI DM6437 EVM board. The followings summerize the procedure:

- TI CCS Ver. 3.3 was used as the compiler with the 'Speed Most Critical' and 'Function Level' option.
- Classifiers with standard exponential functions and ones with our proposal were implemented on the board.
- The 494-th frame of Claire was used as an observed picture, which was enlarged and converted into NTSC video so that it can be acquired by the board.

Table 1 summerizes the computational speed of the experimental results of the UV-based skin-color extraction. The abbreviations 'Cons.' and 'Props.' denote the classifiers with the standard functions and the proposed algorithms, respectively. As the standard exponential functions, we adopted '`expf()`' of single precision and '`exp()`' of double precision from the standard C library. For the proposal, the results only with the initial decision process ($L = 0$) and those with the refinement process up to 8 updates ($L = 8$) are given. Two types of representations, i.e. fixed- and floating-point one, for constant $K_{k,n}$ are also compared. Since the refinement process is rarely required, the overhead is negligible in both cases.

We can verify that as the distributions of classes become close to each other, e.g. Kullback-Leibler divergence [7] becomes close to zero, the refinement ratios have a tendency to increase and may fail to accelerate the process. Although our proposed method is not universally efficient, it shows advantages of acceleration (or power saving) when the distributions are far from each other.

5. CONCLUSIONS

In this work, we reduced the comparison of GMM to simple interval calculations and proposed an efficient Bayesian decision scheme. In order to verify the significance, some experimental results of the application to skin-color extraction were shown. It was verified that the classification was almost completed with the simple initial decision process, and that the refinement process was able to improve the classification with low overhead in terms of the computational speed. Since the proposed algorithm is hardware-friendly, the implementation on ASIC/FPGA is expected as well as on programmable DSPs.

6. REFERENCES

[1] Anil K. Jain, Robert P. W. Duin, and Jianchang Mao, "Statistical pattern recognition: A review," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, Jan. 2000.

[2] S. J. McKenna, S. Gong, and Y. Raja, "Modelling facial colour and identity with Gaussian mixtures," *Pattern Recognition*, vol. 31, no. 12, pp. 1883–1892, 1998.

[3] Son Lam Phung, Abdesselam Bouzerdoum, and Douglas Chai, "Skin segmentation using color pixel classification: Analysis and comparison," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 1, pp. 148–154, Jan. 2005.

[4] X. Cui and Y. Gong, "A study of variable-parameter Gaussian mixture hidden markov modeling for noisy speech recognition," *IEEE Trans. on Audio, Speech and Language Proc.*, vol. 15, no. 4, pp. 1366–1376, May 2007.

[5] Naonori Ueda and Zoubin Ghahramani, "Bayesian model search for mixture models based on optimizing variational bounds," *Neural Networks*, vol. 15, pp. 1223–1241, 2002.

[6] Nikolaos Nasios and Adrian G. Bors, "Variational learning for Gaussian mixture models," *IEEE Trans. on Systems, Man, and Cybernetics - Part B*, vol. 36, no. 4, pp. 849–862, Aug. 2006.

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

[8] Minghua Shi, A. Bermak, S. Chandrasekaran, and A. Amira, "An efficient FPGA implementation of Gaussian mixture models-based classifier using distributed arithmetic," in *IEEE Proc. of Int. Conf. on Electronics, Circuits and Systems '06*, Dec. 2006, pp. 1276–1279.

[9] Uwe Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays, 3rd Ed.*, Springer, 2007.

[10] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical Recipes: The Art of Scientifice Computing, 3rd Ed.*, Cambridge Univ. Pr., Aug. 2007.