論文 / 著書情報
Article / Book Information

Title	A Fast MAP Adaptation Technique for GMM-supervector-based Video Semantic Indexing Systems
Author	Nakamasa Inoue, Koichi Shinoda
Citation(English)	Proc. ACM Multimedia 2011, Vol. , No. , pp. 1357-1360
Issue date	2011, 11
Copyright	Copyright (c) 2011 Association for Computing Machinery
Set statement	Copyright (C)2011 Association for Computing Machinery(ACM), . This is the author's version of the work. It is posted here by personal use. Not for redistribution. The definitive version of Record was published in Nakamasa Inoue, Koichi Shinoda, ACM Multimedia'11,2011,pp. 1357-1360.http://dx.doi.org/10.1145/2072298.2072014
Note	このファイルは著者（最終）版です。 This file is author (final) version

A Fast MAP Adaptation Technique for GMM-supervector-based Video Semantic Indexing Systems

Nakamasa Inoue, Koichi Shinoda
Dept. of Computer Science,
Tokyo Institute of Technology, Japan
{inoue, shinoda}@ks.cs.titech.ac.jp

ABSTRACT

We propose a fast maximum a posteriori (MAP) adaptation technique for a GMM-supervectors-based video semantic indexing system. The use of GMM supervectors is one of the state-of-the-art methods in which MAP adaptation is needed for estimating the distribution of local features extracted from video data. The proposed method cuts the calculation time of the MAP adaptation step. With the proposed method, a tree-structured GMM is constructed to quickly calculate posterior probabilities for each mixture component of a GMM. The basic idea of the tree-structured GMM is to cluster Gaussian components and approximate them with a single Gaussian. Leaf nodes of the tree correspond to the mixture components, and each non-leaf node has a single Gaussian that approximates its descendant Gaussian distributions. Experimental evaluation on the TRECVID 2010 dataset demonstrates the effectiveness of the proposed method. The calculation time of the MAP adaptation step is reduced by 76.2% compared to that of a conventional method and resulting accuracy (in terms of Mean average precision) was 10.2%.

Categories and Subject Descriptors

I.4.9 [**Computing Methodologies**]: Image processing and computer vision—*Applications*

General Terms

Algorithm, Experimentations

Keywords

Video Semantic Indexing, GMM Supervectors, MAP Adaptation

1. INTRODUCTION

Video semantic indexing is one of the fundamental challenges in the field of computer vision. Its goal is to develop a generic method for automatically assigning semantic tags to video data. Detecting semantic concepts including objects, events and scenes is a challenging task due to within-class variations in shapes, colors, illuminations and backgrounds.

Many studies have started to focus on large-scale video recognition—a problem that requires a large amount of computational resources in recent years. For example, in the TREC Video Retrieval Evaluation (TRECVID) [1] workshop, which provides researchers with common tasks of video analysis, the size of target video resources has been doubling every year. While developing an accurate classification method is a top priority, a fast method for processing such a large amount of data is also needed. In response to these requirements, the motivation of the present study is to develop a fast and accurate semantic indexing system.

In recent researches, supervector coding of local features [2] has been proposed as an image-annotation method and applied to video recognition. For example, using a combination of Gaussian mixture model (GMM) supervectors and support vector machines (SVMs) has achieved better performance than that achieved by the bag-of-visual-words approach [3, 4]. With this method, a probability density function (pdf) of extracted local features is estimated by a GMM using maximum a posteriori (MAP) estimation. This MAP estimation step is called MAP adaptation. Aiming to reduce computational costs, most previous studies focused on reducing the cost of feature extraction (e.g., GPU implementations in [5]). The calculation time of the MAP adaptation step, however, has become the bottleneck of a system.

In the present study, we propose a fast maximum a posteriori (MAP) adaptation technique for a GMM-supervectors-based video semantic indexing system. We evaluated the technique by testing it on the TRECVID 2010 dataset.

2. PROPOSED METHOD

2.1 Gaussian Mixture Model

A probability distribution function (pdf) of local visual (or audio) features is estimated for each video shot. Here, Gaussian mixture models (GMMs), whose pdf is given by

$$p(x|\theta) = \sum_{k=1}^{K} w_k g_k(x), \quad g_k(x) = \mathcal{N}(x|\mu_k, \Sigma_k), \qquad (1)$$

are employed where x is a local feature, $\theta = \{w_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ is a set of parameters, K is the number of mixture components, w_k is a mixture coefficient, and $g_k(x)$ is a pdf with a mean vector μ_k and a covariance matrix Σ_k.

The GMM parameters are estimated by using an expectation maximization (EM) algorithm with a maximum a posteriori (MAP) criterion. For MAP adaptation, a GMM for prior distribution, namely a universal background model (UBM), is first needed. The UBM presents how the features are distributed in the general case: therefore, the parameters used for the UBM, $\hat{\theta}^{(\mathrm{U})}$, is estimated by applying the EM algorithm to all features in training videos.

With the proposed method, only mean vectors are adapted for each shot. The MAP solution gives the following equations:

$$\hat{\mu}_k^{(s)} = \frac{\tau \hat{\mu}_k^{(\mathrm{U})} + \sum_{i=1}^n c_{ik} x_i}{\tau + \sum_{i=1}^n c_{ik}}, \quad c_{ik} = \frac{w_k g_k(x_i)}{\sum_{k=1}^K w_k g_k(x_i)}, \quad (2)$$

where $X_s = \{x_i\}_{i=1}^n$ is a set of (one type of) feature vectors extracted from the s-th shot, $\hat{\theta}^{(\mathrm{U})}$ is the UBM parameter, and τ is a predefined hyper-parameter.

2.2 Tree-structured GMMs

Given the UBM, a tree structure of Gaussian components that makes calculation of Eq. (2) efficient is constructed. The basic idea is to cluster Gaussian components by a single Gaussian. For given Gaussian components $g_m = \mathcal{N}(\cdot|\mu_m, \Sigma_m)$ and combination coefficients α_m ($\alpha_m \geq 0, \sum \alpha_m = 1$) ($m = 1, 2, \cdots, M$), we define a combined single Gaussian by

$$\alpha_1 g_1 \oplus \alpha_2 g_2 \oplus \cdots \oplus \alpha_M g_M = \mathcal{N}(\cdot|\bar{\mu}, \bar{\Sigma}), \quad (3)$$

$$\bar{\mu} = \sum_{m=1}^M \alpha_m \mu_m, \quad \bar{\Sigma} = \sum_{m=1}^M \alpha_m (\Sigma_m + \mu_m \mu_m^\mathsf{T}) - \bar{\mu}\bar{\mu}^\mathsf{T}. \quad (4)$$

If only a set of Gaussians components $G = \{g_1, g_2, \cdots, g_M\}$ is given,

$$g(G) = \frac{1}{M} g_1 \oplus \frac{1}{M} g_2 \oplus \cdots \oplus \frac{1}{M} g_M. \quad (5)$$

is simply used. Figure 1 shows an example of a tree-structured GMM. Each leaf node corresponds to a Gaussian component of the GMM, and each other node has a single Gaussian obtained by combining corresponding Gaussian pdfs of descendant nodes. As for the following tree-construction algorithm, it is assumed that the maximum number of child nodes for each layer (with the exception of the leaf layer) is given. For example, if the maximum number of child nodes for the first layer is two and that for the second layer is three, the resulting tree will be designed as shown in Figure 1. In this case, a tree with a depth of three (including the leaf layer) is obtained. This tree-structured GMM is denoted as $\mathcal{T}_{(2,3)} = (V, E, G_{\mathrm{TREE}})$ where V is a set of nodes, E is a set of edges, and $G_{\mathrm{TREE}} = \{g^{(v)} | v \in V\}$ is a set of Gaussian pdfs. In general, a node at the t-th layer of a tree $\mathcal{T}_{(P_1, P_2, \cdots, P_T)}$ has, at most, P_t child nodes.

The node pdfs $g^{(v)}$ of a tree $\mathcal{T}_{(P_1, P_2, \cdots, P_T)}$ are created by the following algorithm. Note that $G_{\mathrm{GMM}} = \{g_1, g_2, \cdots, g_K\}$ is a set of mixture components of the UBM, $G^{(v)}$ is a subset of G_{GMM} corresponding to node v, $g^{(v)}$ is a Gaussian pdf for node v, and $d(g_m, g_n)$ is the sum of Kullback-Leibler divergence from g_m to g_n and that of from g_n to g_m.

1. Prepare a queue and enqueue $(r, G^{(r)})$, where r is the root node, and $g^{(r)} \leftarrow g(G^{(r)})$ ($G^{(r)} = G_{\mathrm{GMM}}$).

2. (Min-max Initialization) Dequeue $(v, G^{(v)})$. Let $\{c_p\}_{p=1}^P$ be the child nodes of node v. For $p = 1, 2, \cdots, P$, initialize child Gaussian pdfs as follows:

$$g^{(c_p)} \leftarrow \alpha \tilde{g}^{(c_p)} \oplus (1 - \alpha) g^{(v)} \quad (6)$$

where $0 \leq \alpha \leq 1$, and $\tilde{g}^{(c_p)}$ is chosen from $G^{(v)}$ by

$$\tilde{g}^{(c_p)} = \begin{cases} \underset{g \in G^{(v)}}{\operatorname{argmax}} \; d(g, g^{(v)}) & (p = 1) \\ \underset{g \in G^{(v)}}{\operatorname{argmax}} \; \underset{1 \leq p' < p}{\min} d(g, \tilde{g}^{(c_{p'})}) & (\text{otherwise}) \\ \quad g \neq \tilde{g}^{(c_1)}, \cdots, \tilde{g}^{(c_{p-1})} & \end{cases}.$$

3. (Clustering by k-means) Assign each Gaussian component in $G^{(v)}$ to the nearest child node, i.e.

$$G^{(c_p)} \leftarrow \{g \in G^{(v)} \mid p = \underset{p'}{\operatorname{argmin}} \, d(g, g^{(c_{p'})})\}. \quad (7)$$

Update $g^{(c_p)}$ by using Eq. (5) as $g^{(c_p)} \leftarrow g(G^{(c_p)})$. Repeat this step until the following sum converges:

$$D = \sum_{p=1}^P \sum_{g \in G^{(c_p)}} d(g, g^{(c_p)}). \quad (8)$$

4. For $p = 1, 2, \cdots, P$, enqueue $(c_p, G^{(c_p)})$ if c_p is not in the $(T+1)$-th layer and $|G^{(c_p)}| > 1$. Go to step 5 if the queue is empty; otherwise, return to step 2.

5. For each node v in the $(T+1)$-th layer, create leaf nodes ℓ for each $g_k \in G^{(v)} \subset G_{\mathrm{GMM}}$ and set $g^{(\ell)} = g_k$.

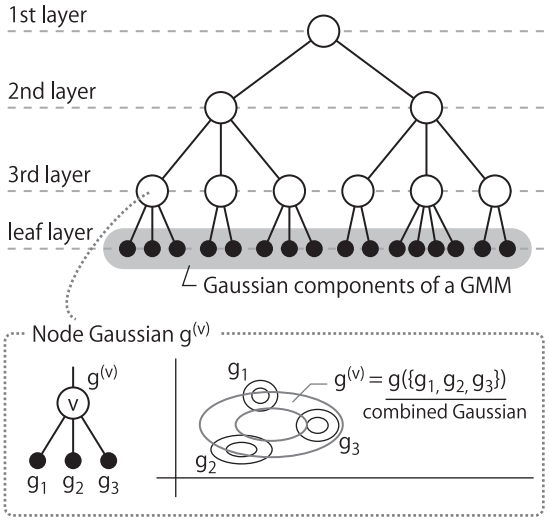

Figure 1: An example of a tree-structured GMM.

2.3 Fast MAP Adaptation

A fast MAP adaptation technique which estimates c_{ik} in Eq. (2) efficiently by using a tree-structured GMM is explained in the following. For a constructed tree-structured GMM $\mathcal{T}_{(P_1, P_2, \cdots, P_T)}$, node weights are first defined as follows:

1. For each leaf node, set $w^{(\ell)} = w_k$ if $g^{(\ell)} = g_k \in G_{\mathrm{GMM}}$.

2. Calculate weights for $t = T+1, T, \cdots, 1$ as follows. For each node v in the t-th layer,

$$w^{(v)} = \sum_{c \in C(v)} w^{(c)}, \quad (9)$$

Table 1: The 30 target semantic concepts in the TRECVID 2010 dataset

Airplane Flying	Animal	Asian People	Bicycling	Boat ship	Bus
Car Racing	Cheering	Cityscape	Classroom	Dancing	Dark-skinned People
Demonstration Or Protest	Doorway	Explosion Fire	Female Human Face Closeup	Flowers	Ground Vehicles
Hand	Mountain	Nighttime	Old People	Running	Singing
Sitting down	Swimming	Telephones	Throwing	Vehicle	Walking

where $C(v)$ is a set of child nodes of the node v.

The algorithm for estimating c_{ik} for feature vector x_i is described as follows. The key idea is to construct a GMM of active nodes V_A, which means vector x_i will be assigned to descendants of nodes in V_A. $|V_A|$ is kept small by obtaining active nodes from the root node.

1. Set $V_A \leftarrow \{r\}$, where r is the root node.
2. Expand active nodes by making child nodes of the active nodes active:
$$V_A \leftarrow \bigcup_{v \in V_A} C(v), \tag{10}$$
where $C(v)$ is a set of child nodes of the node v. Here, $C(\ell) = \{\ell\}$ is used for leaf nodes ℓ to keep the leaf nodes active.
3. Consider an active GMM given by
$$p(x|V_A) = \sum_{v \in V_A} \tilde{w}^{(v)} g^{(v)}(x), \quad \tilde{w}^{(v)} = \frac{w^{(v)}}{\sum_{v \in V_A} w^{(v)}}.$$
Calculate
$$c_i^{(v)} = \frac{\tilde{w}^{(v)} g^{(v)}(x_i)}{\sum_{v \in V_A} \tilde{w}^{(v)} g^{(v)}(x_i)} = \frac{w^{(v)} g^{(v)}(x_i)}{\sum_{v \in V_A} w^{(v)} g^{(v)}(x_i)}.$$
4. Keep a node v active if $c_i^{(v)}$ is larger than the predetermined threshold c_{TH}, i.e.
$$V_A \leftarrow \{v \in V_A \mid c_i^{(v)} > c_{\mathrm{TH}}\} \tag{11}$$
5. If all nodes in V_A are leaf nodes, output
$$\hat{c}_{ik} = \begin{cases} c_i^{(\ell)} & (\ell \in V_A,\ g^{(\ell)} = g_k) \\ 0 & (\text{otherwise}) \end{cases}. \tag{12}$$
Otherwise, return to step 2.

Since the observed $\hat{c}_{ik}$ for non-active nodes is 0, the sum in Eq. (2) can be calculated for non-zero $\hat{c}_{ik}$ only. Moreover, calculation speed and levels of approximation can be controlled by selecting the threshold in $0 < c_{\mathrm{TH}} \leq 1$. Note that the same c_{ik} as given by Eq. (2) is obtained if c_{TH} is set to 0 (because all leaf nodes will be active at the final step).

2.4 GMM Supervector

The combination of GMM supervectors and support vector machines (SVMs) was first proposed as a speaker recognition method [6] and has been applied to image and video recognition [3, 4]. GMM supervectors are created for each shot given by

$$\phi(X_s) = \begin{pmatrix} \tilde{\mu}_1^{(s)} \\ \tilde{\mu}_2^{(s)} \\ \vdots \\ \tilde{\mu}_K^{(s)} \end{pmatrix}, \ \tilde{\mu}_k^{(s)} = \sqrt{w_k^{(\mathrm{U})}} \left(\Sigma_k^{(\mathrm{U})}\right)^{-\frac{1}{2}} \hat{\mu}_k^{(s)}, \tag{13}$$

where $\hat{\mu}_k^{(s)}$ is an adapted mean vector obtained from Eq. (2), and $\theta^{(\mathrm{U})}$ is the GMM parameter for the UBM. Finally, SVMs are trained for each semantic concepts and for each type of features by using RBK kernels.

Note that while the scope of this paper is fast creation of the GMM supervectors, the proposed technique can be used for creating the Fisher vectors [7].

3. EXPERIMENTS

3.1 Database and Task

Our experiments were conducted on the TRECVID 2010 dataset [1]. The dataset consists of 400 hours of Internet archive videos with creative commons licenses. The shot boundaries are automatically detected and provided with the video data. Half of the videos were used for training, and the others were used for testing. The number of shots was 119,685 for training and 146,788 for testing. The task of semantic indexing in the experiment is to detect the 30 semantic concepts listed in Table 1. The target 30 concepts (including objects, events and scenes) were selected at the TRECVID 2010 workshop. They are considered to be useful for video searching.

The evaluation measures are Mean average precision (Mean AP) and calculation time (CT) of the testing phase. Mean AP is defined as the mean of APs over all 30 target concepts, and APs are estimated by using a method called inferred average precision (Inf AP), proposed in [8].

3.2 Experimental Conditions

The following four types of visual and audio features are extracted from video data: 1) SIFT features with Harris-Affine detector (SIFT-Har), 2) SIFT features with Hessian-Affine detector (SIFT-Hes), 3) SIFT and hue histogram with dense sampling (SIFTH-Dense), and 4) MFCC audio features (MFCC).

The number of mixtures (vocabulary size) K for GMMs was 512 for visual features and 256 for audio features. For computational efficiency, it was assumed that covariance matrices are diagonal. Hyper parameter τ in the MAP adaptation was set to 20.0. SVMs were trained for each semantic concepts by using the libSVM implementation [9]. For tree-structured GMMs, the optimal tree structure $\mathcal{T}_{\mathrm{opt}}$ was selected as

$$\mathcal{T}_{\mathrm{opt}} = \operatorname*{argmin}_{\mathcal{T} \in \mathcal{S}} \mathrm{CT}(\mathcal{T}), \tag{14}$$

$$\mathcal{S} = \{\mathcal{T}_{(P_1, P_2, \cdots, P_T)} \mid 1 \leq T \leq 5, 1 \leq P_t \leq 5\}, \tag{15}$$

where $\mathrm{CT}(\mathcal{T})$ is calculation time when the tree $\mathcal{T}$ is used. The trees $\mathcal{T}_{(4,4,5)}$, $\mathcal{T}_{(4,5,5)}$, $\mathcal{T}_{(3,4,4,5)}$ and $\mathcal{T}_{(3,3)}$ were selected for SIFT-Har, SIFT-Hes, SIFTH-Dense and MFCC, respectively. Parameter α in Eq. (6) was fixed to 0.1. Threshold c_{TH} for the fast MAP adaptation was set to 0.001. Here, a low threshold was set so as to keep detection performance

Table 2: Comparison of Mean Inf AP (%) in terms of different features.

Feature	No tree	$\mathcal{T}_{\text{opt}}$
SIFT-Har	6.30	6.32
SIFT-Hes	5.96	6.08
SIFTH-Dense	7.10	6.95
MFCC	1.99	2.00
Fusion	10.15	10.16

Table 3: Calculation time (sec) for MAP adaptation.

Feature	No tree	$\mathcal{T}_{\text{opt}}$	$\mathcal{T}_{\text{binary}}$
SIFT-Har	1.62	0.49	0.98
SIFT-Hes	1.67	0.48	1.00
SIFTH-Dense	2.89	0.81	1.89
MFCC	0.22	0.03	0.08

high. Experiments using different thresholds were also conducted (see Subection 3.3.3).

In the experiments, calculation time was measured by using one Intel Xeon 2.93 GHz CPU.

3.3 Results

3.3.1 Mean Inf APs

Table 2 summarizes obtained Mean Inf APs for each features and a fusion method. The Fusion combines four features by calculating the weighted sum of detection scores (fusion weights were optimized by using two-fold cross validation on training data). As a result, we can see that the Mean Inf APs using tree-structured GMMs are comparable to those using no trees.

Estimation errors of c_{ik} were also evaluated from the mean absolute error (MAE), given as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}|\hat{c}_{ik} - c_{ik}|, \tag{16}$$

where $\hat{c}_{ik}$ and c_{ik} are given by Eq. (12) and Eq. (2), respectively. The MAE for SIFTH-Dense was 0.32 on average (note that $0 \leq \text{MAE} \leq 2$). Although we have estimation errors of c_{ik} in the fast MAP adaptation algorithm, they can be cancelled when the distance between two GMM supervectors is calculated since the same errors occur in training and testing phases.

3.3.2 Calculation Time

Table 3 lists calculation times for MAP adaptation using different features and different trees. The results for binary trees ($\mathcal{T}_{\text{binary}}$) are also listed in the table. The calculation speed when the optimal tree is used on average 4.2 times faster than when trees were not used; that is, calculation time was reduced by 76.2%. The MAP adaptation step took, on average, 0.055 msec per feature.

3.3.3 Effect of using different thresholds

Table 4 lists the results obtained using different thresholds c_{TH} for the fast MAP adaptation. The number of leaf nodes that are active (at least once in Eq. (10)) and MAE are also listed in the table.

Table 4: Comparison of Mean Inf AP (%), calculation time (sec) for MAP adaptation (CT), number of leaf nodes $|V_A|$ and Mean absolute error (MAE) by using different thresholds c_{TH} (for SIFTH-Dense).

c_{TH}	Mean Inf AP	CT	$\lvert V_A\rvert$	MAE
0.001	6.95	0.81	17.0	0.32
0.01	6.99	0.68	11.2	0.53
0.1	6.60	0.59	7.3	0.80
0.5	6.41	0.53	5.4	0.98

As c_{TH} gets higher, the calculation time shortens, but Mean Inf AP was decreased when $c_{\text{TH}} = 0.1$ and 0.5. Moreover, the number of active leaf nodes decreases, and MAE increases. It can thus be concluded that calculation time should be reduced not by setting a high threshold c_{TH} but by selecting a better-structured tree to keep detection performance high. In particular, c_{TH} should be equal to or smaller than 0.01 in this case.

4. CONCLUSION

A fast MAP adaptation technique using tree-structured GMMs for a video semantic indexing system was proposed. The detection time was reduced by 56.6%, compared to that of convention method, while high detection performance was maintained. The best result, in terms of Mean Inf AP, attained by our fusion method tested on the TRECVID 2010 dataset was 10.16% when tree-structured GMMs were used and 10.15% when no trees were used. Our future work will focus on a GPU implementation of the fast MAP adaptation. As a result, feature extraction and MAP adaptation will be independently conducted for each local feature by GPUs.

5. REFERENCES

[1] A. F. Smeaton, and et al. Evaluation campaigns and trecvid. In Proc. of *ACM Multimedia MIR* workshop, pages 321–330, 2006.

[2] X. Zhou, and et al. Image classification using super-vector coding of local image descriptors. In Proc. of *ECCV*, 2010.

[3] X. Zhou, and et al. Sift-bag kernel for video event analysis. In Proc. of *ACM Multimedia*, 2008.

[4] N. Inoue, and et al. High-Level Feature Extraction using SIFT GMMs and Audio Models. In Proc. of *ICPR*, 2010.

[5] C. Wu. SiftGPU: A GPU implementation of sift. http://cs.unc.edu/~ccwu/siftgpu, 2007.

[6] W. M. Campbell, and et al. Support vector machines using gmm supervectors for speaker verification. In *IEEE Signal Processing Letters*, 13:308–311, 2006.

[7] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *In Advances in Neural Information Processing Systems 11*, pages 487–493. MIT Press, 1998.

[8] E. Yilmaz, and et al. A simple and efficient sampling method for estimating ap and ndcg. In Proc. of *ACM SIGIR*, pages 603–610, 2008.

[9] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm, 2001.