Variational maximization-maximization of Bayesian mixture models and application to unsupervised image classification

Lim, Kart-Leong

2018

Lim, K.-L. (2018). Variational maximization-maximization of Bayesian mixture models and application to unsupervised image classification. Doctoral thesis, Nanyang Technological University, Singapore.

<http://hdl.handle.net/10356/73199>

<https://doi.org/10.32657/10356/73199>

Variational Maximization-Maximization of Bayesian Mixture Models and Application to Unsupervised Image Classification

Kart-Leong, Lim

School of Electrical and Electronics Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2017

Acknowledgments

I wish to thank the following people who helped me during my PhD at School of EEE, NTU. My gratitude to Prof. Rusli for supporting my case for a change of supervisor. My current supervisor, Prof. Wang Han for his support which is immeasurable. My ex-colleague Dr Wang Shiping for his helpful discussions. The supportive EEE management and staffs, Prof. Pita, Ms Koh, Ms Wong and Ms Tay.

Thank you all!

Contents

Acknowledgments	i
Contents	ii
Summary	vii
List of Figures	ix
List of Tables	xiii
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
1.2.1 New Learning Framework for Faster Optimization of BMM	2
1.2.2 Improve Image Modeling with Specialized BMM	3
1.3 Major Contributions of Thesis	3
1.4 Organization of the Thesis	6
2 Literature Review	9
2.1 Introduction	9
2.2 Maximum Likelihood of Mixture Models	9
2.2.1 Classical EM algorithm for GMM	9
2.2.2 Supervised GMM	12
2.2.3 GMM with Regularization	14
2.3 Variational Inference of Bayesian Mixture Models	16
2.3.1 Variational Inference Framework	17
2.3.2 Learning Algorithms for Variational Posteriors	18
2.3.3 Variational Maximization-Expectation Algorithm for Bayesian K-means	19

2.3.4	Approximating Non-iid Image Representation with the Variational Expectation-Expectation Algorithm	20
2.4	Fast Algorithms of Bayesian Nonparametrics	23
2.4.1	DPsearch	24
2.4.2	DPmeans	26
2.4.3	MAP-DPM	27
2.5	Relating Sparse Coding to Bayesian Mixture Mode	29
2.5.1	From K-means to Sparse Coding	29
2.5.2	Probablistic Interpretation of Sparse Coding	30
2.6	Bayesian Mixture Model based Fisher Vector	32
2.6.1	GMM based Fisher Vector	33
2.6.2	Extending Fisher Vector with Prior Models	34
2.6.3	When Fisher Vector meets ConvNet	36
2.7	Strategies for Modeling Image using Local Structure	38
2.7.1	A Local Pairwise Codebook	39
2.7.2	Detecting Spatially Close Keypoints	40
2.7.3	Deep Learning Approach	42
3	A New Method for Learning Bayesian Mixture Model	43
3.1	Introduction	43
3.2	Related Work	44
3.3	Background	45
3.3.1	Gaussian Mixture Model	45
3.3.2	Bayesian Approach	45
3.3.3	Variational Inference and Learning Objective	46
3.3.4	Variational Learning of Bayesian GMM	48
3.4	Proposed Method, MM-GMM	51
3.4.1	Defining Variational Posteriors	52
3.4.2	Approximating Expectation of Variational Posteriors	52

3.4.3	Jensen’s Inequality for Cluster Assignment	54
3.5	Proposed Theoretical Analysis	54
3.5.1	Proof to Analytical Solution of Proposed Method	54
3.5.2	Convergence of Proposed Method	55
3.5.3	Comparison to Variational Inference	57
3.5.4	Comparison to Maximum Likelihood	60
3.6	Experiments	61
3.6.1	Unsupervised Image Classification on Object Categories	62
3.6.2	Unsupervised Image Classification on Scene Categories .	65
3.6.3	Unsupervised Feature Learning For Multiview Car Pose and Object Classification	68
3.7	Conclusion	73
4	Learning Bayesian Mixture Model with Dirichlet Process	75
4.1	Introduction	75
4.2	Related Work	76
4.2.1	Fast DPM Algorithms	76
4.3	Background	80
4.3.1	Dirichlet Process Mixture	80
4.3.2	Variational Posteriors	81
4.3.3	Problem Formulation	82
4.3.4	Learning Algorithm of Variational Inference	83
4.4	Proposed Method, MM-DPM	84
4.4.1	Expectation of Variational Posteriors	84
4.4.2	Proof to Analytical Solution of Expectations	86
4.4.3	Convergence of Proposed Method	90
4.4.4	Theoretical Comparision to Variational Inference	92
4.4.5	Proposed Algorithm, MM-DPM	94
4.5	Proposed Future Work	96

4.5.1	Modeling Hyperparameter in DPM	96
4.6	Experiments	98
4.6.1	Synthetic Dataset	98
4.6.2	Expectation of Variational Posteriors: Estimated Lower Bound vs Actual	103
4.6.3	UCI Datasets Cited in [56, 92]	105
4.6.4	Challenging UCI Datasets	108
4.6.5	Real Image Datasets	113
4.7	Conclusion	117
5	Learning Bayesian Mixture Model with Sparse Coding	118
5.1	Introduction	118
5.2	Related Works	119
5.3	Background	119
5.3.1	GMM based Fisher Vector (GMM-FV)	119
5.3.2	Learning Sparse Coding with Convex Solver	120
5.3.3	Sparse Coding based Fisher Vector (SC-FV)	121
5.4	Proposed Model, sGMM	121
5.4.1	Sparse Coding based GMM (sGMM)	122
5.4.2	Variational Posterior Learning	123
5.4.3	sGMM based Fisher Vector (sGMM-FV)	125
5.4.4	Proof to Analytical Solution	125
5.5	Experiments	127
5.5.1	Setup	127
5.5.2	Discussion on Experimental Results	129
5.6	Conclusion	132
6	Learning Bayesian Mixture Model with Local Structure	133
6.1	Introduction	133

6.2	Related Works	133
6.3	Background	133
6.3.1	Gaussian mixture model	134
6.3.2	Maximization-maximization algorithm	135
6.4	Proposed Model, FoG	136
6.4.1	Field-of-Gaussian Mixture Model (FoG)	136
6.4.2	Proof to Analytical Solution of FoG	138
6.4.3	Hierarchical Feature Learning	139
6.5	Experiments	141
6.5.1	Field of Gaussian mixture model	141
6.5.2	Hierarchical Feature Learning	144
6.6	Conclusion	145
7	Conclusion and Recommendations	147
7.1	Conclusion	147
7.2	Recommendations for Further Research Works	149
7.2.1	Non-Analytical Approximation	150
7.2.2	Bayesian Neural Network	151
7.2.3	ConvNet based Fisher Vector	152
7.2.4	Stochastic Variational Inference	153
	Author's Publications	154
	References	155

Summary

This thesis mainly propose variational inference for Bayesian mixture models and their applications to solve machine learning problems. The mixture models addressed are the Gaussian mixture model (GMM), Dirichlet process mixture (DPM), the sparse coding based Gaussian mixture model (sGMM) and the Field-of-Gaussian (FoG) mixture model.

In most mixture models, when using a Bayesian approach, usually an inherent problem will arise from analytically intractable posterior distributions of the mixture models. Most recent works either apply Gibbs sampling or variational inference to solve this problem. Variational inference rely on two assumptions mainly the mean field theory and conjugate priors. Despite proven faster convergence than Gibbs sampling, the learning algorithm of variational inference is not computationally efficient. The learning of variational inference can be widely seen as first estimating the class assignment variable and then applying it to estimate parameters of the mixture model. The estimate is usually performed by computing the expectations of the prior models. However, learning is not exclusive to expectation. Welling and Kurihara saw other possible configurations that use different combinations of maximization or expectation for the estimation. For instance, the learner in variational inference is generalized under the expectation-expectation (EE) algorithm. Inspired by this, another variant of EE known as the maximization-maximization (MM) algorithm is proposed for the Bayesian mixture models mentioned earlier. However, MM is not without issue. Firstly, it is very rare to find any theoretical study comparing MM to EE. Secondly, the computational efficiency and the accuracy of MM is rarely compared to EE. Hence, it is difficult to convince the use of MM over a mainstream learner such as EE or even Gibbs sampling. In this thesis, a revisit to the learning objective of EE on a Bayesian GMM case is made with comparison to the MM objective.

In Bayesian nonparametrics, variational inference is not the mainstream in fast algorithms for DPM mainly due to high computational cost. Instead, most fast algorithms are largely based on MAP estimation of Gibbs sampling probabilities. However, they usually face intractable posterior and typically degenerate the conditional likelihood to overcome the inefficiency. Inspired by fast DPM algorithms, the MM algorithm is proposed for approximating expectations of variational posteriors for DPM. The proposed method outperforms variational inference in terms of clustering accuracy, model selection and has comparable computational efficiency to fast DPM algorithms.

The Sparse Coding based Fisher Vector extends traditional GMM based Fisher Vector with a sparse term. In the original work, an off-the-shelf Sparse Coding solver is required. From a statistical perspective, an off-the-shelf solver may appear as a black-box. A more elegant way is to use a probabilistic model to learn Sparse Coding. In this thesis a new GMM known as the Sparse Coding based GMM (sGMM) is proposed. It differs from standard GMM by an additional sparse coefficient hidden variable. Variational inference using the MM algorithm is then proposed to perform parameter estimation of sGMM. These parameters are then used to compute a Sparse Coding based Fisher Vector for image representation.

Most unsupervised learners such as K-means, GMM or Sparse Coding do not use structure information found from the neighbourhood of an image patch for parameter estimation. A recent image prior technique combine unsupervised learning with Markov Random Field (MRF) by replacing the MRF potential with an unsupervised learner. It uses a neighbourhood of image patches for learning the parameters of the unsupervised learner, thus capturing image structure information. In this thesis, the above image prior technique is applied to the GMM parameter estimation so that the learnt model is more expressive than GMM learnt from a single image patch alone.

List of Figures

2.1	(Left) A complete observation $\{x, z\}$ of three states which indicate where the samples came from. (Right) An incomplete observation, x which ignore the hidden state, z [6].	10
2.2	(A) Original data. (B) Unsupervised clustering mainly focus on maximizing the between-class distance and minimizing the within-class distance, or the likelihood optimization w.r.t. the cluster means. It can generalize very well but it has poor purity. (C) Supervised clustering has good purity but poor generalization (D) Proposed supervised GMM tries to combine the best of both worlds, purity and likelihood [30].	13
2.3	The maximum likelihood approach to an image model treat the image as a set of identical and independently distributed image patches. This is a poor assumption since there are dependency between image patches [16].	21
2.4	Maximum Likelihood approach to the GMM (left), where the hidden variables λ, μ, π are treated as point estimates. The Bayesian approach to GMM (right) where the hidden variables λ, μ, π are now modeled using prior distributions with hyperparameters defined as α, a, b, m, β	21
2.5	Illustration of a set of image basis for sparse coding	31
2.6	Sparse coding representation of an image patch. Noise is assumed zero here.	31
2.7	Example of max pooling in ConvNet. After the max operation, it is followed by a downsampling.	36
2.8	Illustration of VLAD computation. μ_k refers to the cluster center and the blue dash refers to the metric distance of each sample from μ_k	37

2.9	Clustering on spatially close descriptors. (left) A naive pairwise codebook approach. (right) The LPC approach. LPC exhibit a more data distribution guided clustering than the grid like clustering used by a naive approach.	39
2.10	Spatially Stable Interest Point Pairs [113]. A radius is used to determine the interest points that are spatially close.	41
2.11	Mean Average Precision on authors' dataset of image re-ranking (A1) Traditional visual word tree. (A2) pair-wise visual word tree. (A3) Visual Rank algorithm [52]	41
2.12	Each multi-layer perceptron is used in loop to form a neural network known as the recurrent neural network. The purpose of such a model is the ability to store memories.	42
3.1	Graph model of Bayesian GMM. The unshaded circles in boxes are hidden variables while the variables outside the boxes are hyperparameters. The shaded circle refers to the observed variable. The arrows show the relationships between variables and hyperparameters.	46
3.2	Visualizing the convergence of EE and MM. The true number of clusters is 4. Top left and center left: Sorted cluster weights of MM after 20 and 100 iterations respectively. Top right and center right: Sorted cluster weights of EE after 20 and 100 iterations. Bottom row: The number of iterations vs the computed negative lower bound for 40 iterations, using MM and EE respectively.	64

3.3	Visualizing the convergence of EE and MM. The true number of clusters is 4. Top left and center left: Sorted cluster weights of MM after 30 and 120 iterations respectively. Top right and center right: Sorted cluster weights of EE after 30 and 120 iterations. Bottom row: The number of iterations vs the computed negative lower bound for 40 iterations, using MM and EE respectively.	68
3.4	Performance comparison for Multi-view Car bin-pose classification	71
3.5	Performance comparison for Caltech-10 object classification . . .	71
4.1	Graph model of DPM	80
4.2	DPM with new hidden variable β	96
4.3	A randomly generated dataset of 5 Gaussian distributed clusters of 2D points. The Gaussian mixture parameters for generating the plots are given in Table 4.4 (ground truth). Note that the order of clustering is not important.	100
4.4	Proposed method correctly identifying the number of mixture components remaining (non-zeros cluster weights) after learning has converged. The initial truncation size is 15.	101
4.5	Experiment II: The number of iterations vs the computed lower bound for the synthetic dataset experiment.	101
4.6	The number of iterations vs the computed lower bound for 10 re-runs per dataset, using the proposed method: 8 UCI datasets cited in [56, 92].	109
4.7	The number of iterations vs the computed lower bound for 10 re-runs per dataset using the proposed method on 5 challenging UCI datasets.	112

4.8	Two different car model (first 5 column and next 5 column) of the Multiview Carpose dataset arranged into 8 view poses or classes (row). Each class has a minor view change of 5 image sequences for each car model. There are 20 different car models in total.	115
4.9	10 object categories or classes of Caltech 101 dataset. Each row shows an object category and each column is an object instance. There are at least 100 images per class.	115
4.10	15 classes from Scene Categories dataset. Each image shows an indoor or outdoor scene such as bedroom, sub-urban, industrial, kitchen and etc.	116
4.11	The number of iterations vs the computed lower bound for 10 re-runs per dataset, using the proposed method: 3 real datasets.	117
5.1	Probabilistic model of Sparse Coding based GMM	122
6.1	Illustration of a random image patch (maximal clique) as a set of connected subpatches (cliques)	136
6.2	Hubel and Wiesel model.	140
6.3	Graphical flowchart of proposed hierarchical feature learning algorithm	140
6.4	Evaluation of proposed method, FoG vs GMM and K-means for 400 training and 1040 testing images.	143
6.5	Evaluation of proposed hierarchical learner using DPmeans and FoG for 400 training and 400 testing images.	145
7.1	Illustration of autoencoder, an unsupervised neural network.	151
7.2	Architecture of netVLAD	153

List of Tables

2.1	Accuracy of using the zero, first and second (w, μ, σ respectively) order statistics of Fisher Vector on the PASCAL VOC 2007 dataset [96].	35
2.2	Accuracy comparison between MoG based Fisher Vector and the LatMoG based Fisher Vector on the PASCAL VOC 2007 dataset [16].	35
2.3	Preserving spatial information, local (LPC) vs global (Sgl). . . .	40
3.1	Dataset used in unsupervised classification I	63
3.2	Performance evaluation using predicted model, accuracy and computational time for 20 iterations of EE and MM. (10 average re-runs). The accuracy of lnGD, lnGau, KNN and SVM are cited in [25]. However, their computational times are not given. The best result is in bold. The proposed method is slightly poorer than EE. EE is slightly poorer than the best method.	63
3.3	EE confusion matrix (1 arbitrary run)	63
3.4	MM confusion matrix (1 arbitrary run)	64
3.5	Dataset used in unsupervised classification II	66
3.6	Performance evaluation using predicted model, accuracy and computational time for 30 iterations of EE and MM. (10 average re-runs). The best result is in bold. The proposed method using SPR vector (800) is close to the best result. EE is unable to compute the SPR vector. The accuracy of GID VAR is cited from [5] but the computational times was not given.	66
3.7	EE confusion matrix (1 arbitrary run)	66
3.8	MM confusion matrix (1 arbitrary run)	66
3.9	MM confusion matrix using SPR vector (1 arbitrary run)	66
3.10	Dataset used in unsupervised learning	69

3.11	Average Precision (AP) evaluation on the Multi-view Car dataset for 8 bin-pose.	72
3.12	Average Precision (AP) evaluation on the Caltech-10 dataset. . .	73
3.13	Comparison of Mean Average Precision on Caltech10	73
4.1	In most variational inference algorithms, expectations of varia- tional posterior distribution are typically obtained by exploiting statistical moments of well-defined conjugate prior.	83
4.2	Lookup table for updating variational posterior hyperparameters	83
4.3	Computation time and iterations needed for proposed method to convergence for synthetic datasets #1,...,#4.	102
4.4	Ground truth vs estimated hidden variables for the synthetic datasets of Gaussian mixtures. For the array that appear in each box, the rows refers to K no. of mixtures while the columns are D no. of dimensions.	102
4.5	Expectations and log-expectations of the variational posteriors computed on synthetic dataset #2 (generated using Table 4.4) using Algorithm 4.1 and 4.2 (proposed). The asterisk beside each row refers to corresponding expectations that were generated by each algorithm.	104
4.6	Expectations and log-expectations of the variational posteriors computed on UCI Iris dataset using Algorithm 4.1 and 4.2 (pro- posed). The asterisk beside each row refers to corresponding ex- pectations that were generated by each algorithm.	104
4.7	8 UCI dataset cited in [56, 92].	107
4.8	Normalized Mutual Information (using average score from 10 re- runs)	107
4.9	Model selection	107
4.10	Number of iterations to convergence	107

4.11	5 challenging UCI dataset.	111
4.12	Normalized Mutual Information (using average score from 10 re- runs)	111
4.13	Model selection (truncation or parameter in bracket)	111
4.14	Number of iterations to convergence	111
4.15	Image Categorization Datasets	116
4.16	Normalized Mutual Information (larger score is better)	116
4.17	Model selection (truncation or parameter in bracket)	116
4.18	Number of iterations to convergence	116
5.1	Image Categorization Datasets	131
5.2	Mean average precision of classification accuracy on 8 classes Multi-view Car dataset. No. of features used for training FV is 20,000 and there are 400 test images.	131
5.3	Mean average precision of classification accuracy on 10 classes Caltech dataset. No. of features used for training FV is 20,000 and there are 2985 test images.	131
5.4	Mean average precision of classification accuracy on 15 classes Scene Categories dataset. No. of features used for training FV is 75,000 and there are 2644 test images.	131
5.5	Mean average precision of classification accuracy on 67 classes MIT Indoor dataset. No. of features used for training FV is 107,200 and there are 1340 test images.	131
6.1	Best performance on the Multi-view Car dataset [84]	143
6.2	Best performance on the Multi-view Car dataset [84]	145

1 Introduction

1.1 Motivation

Business products that incorporate machine learning is now the key emerging trend in industries globally. From Google keyword search (text) to iPhone SIRI (speech) to health care (analytics) to anti-terrorism surveillance (video) and to Alipay face recognition (image). The cutting edge technology in machine learning today is without a doubt, deep neural network (DNN). In order to train a good DNN such as deep ConvNet [61] from nothing, a very huge image dataset with class labels usually by the millions is required. The main problem with such mode of learning a.k.a. supervised learning is that a class label is needed for every image or sample during training [60].

Yet according to some machine learning figureheads such as Geoffrey Hinton [55, 46], Yann LeCun [60, 61] and Michael Jordan [53, 9, 10], machine learning is still in its infancy. In fact, some of the immediate challenges of machine learning are unsupervised learning and sequential learning according to LeCun [60]. For unsupervised learning in particular (such as K-means or Gaussian mixture model), learning a model of the dataset does not require any class labels. However in these methods, it is required to know the number of classes upfront before learning. Unlike K-means, a new branch of unsupervised learning referred to as Bayesian mixture model (BMM) in this thesis, can model each dataset without specifying the number of classes before learning. In general, BMM is considered as the Bayesian approach to mixture models with infinite number of classes. The latter is also related to Bayesian Nonparametrics (BNP) [56, 25, 5, 9, 34, 13, 79], but BNP mainly emphasis on mixture models with stochastic process such as the Dirichlet process. The infinite class BMM is usually truncated [49, 9, 18] such that there is a fixed large upper limit of classes due to practical implementation constraint, although in theory it can have infinite

classes.

While DNN excels at human inspired **perception**, BMM excels at handling the **uncertainty** or the thinking part in humans [106]. Although, some very recent works have begun to investigate how **perception** and **uncertainty** can merge together to form what they called deep Bayesian neural network [106, 54, 38, 32, 4]. Their successes are currently limited to shallow neural networks with one or two hidden layers. A DNN such as the ultra deep ResNet [42] today can go up to 100 hidden layers with state-of-the-arts result. Modeling such DNN using BMM is too complex and does not exist at the moment. Moreover, there are some major problems within BMM such as large computational cost that remains unsolved and it is a hot topic in recent literature [47, 54]. Inspired by the above problems, this thesis is mainly dedicated to advancing the knowledge of BMM for machine learning.

1.2 Objectives

Before discussing the objectives, some research problems of BMM must be identified. One notable issue is the scalability of BMM to large datasets due to the complexity in implementation and the intractability in the posterior distributions. Another problem faced by BMM is the lack of a more compatible model to tailor suit to specific problems in images.

1.2.1 New Learning Framework for Faster Optimization of BMM

In the Bayesian approach to modeling mixture models such as the Gaussian mixture (GMM) and the Dirichlet process mixture (DPM), due to the additional use of prior distributions, direct posterior inference of these BMM are intractable. The well established approximate inference techniques are Gibbs sampling and the more recent variational inference. However, variational inference in particular is not scalable to large datasets as examined in chapters 3 and 4. Hence,

the first objective is to develop a faster optimization approach for BMM. The contributions of the proposed new framework to fast optimization of BMM will be briefly discussed in section 1.3.

1.2.2 Improve Image Modeling with Specialized BMM

Another objective is to introduce new BMMs which improve certain weakness of generic BMMs when they are applied to image dataset. Such issue may include improving the modeling of BMM by using structural information found from local neighbourhood (chapter 6), or adding sparsity to the mixture parameters to prevent poor fitting when higher dimensional feature vector is used (chapter 5). Before diving into the working principles, in section 1.3 the major contributions following this objective are highlighted.

1.3 Major Contributions of Thesis

A summary of the major contributions made to the thesis during my PhD candidature is discussed below.

1. The Expectation-Maximization (EM) algorithm or K-means is usually used to learn parameters from training dataset for VLAD or BoW feature representation. An alternative learner is the Maximization-Maximization (MM) algorithm and it is proposed to perform a similar task to K-means for object pose estimation on both the NTU maritime dataset and the Multi-view Car dataset. From experiments, it is found that MM is significantly faster to compute than EM and also MM outperforms K-means in terms of mean Average Precision for all cluster sizes as far as VLAD and BoW feature representation for object pose estimation is concerned. Another interesting fact is that both MM and K-means have some similarity in their objective function for cluster assignment.

2. Most unsupervised learners such as K-means or EM do not exploit local structure information found from the neighborhood of each image patch. Instead, each random training sample is an independent image patch used to learn the BMM (e.g. GMM) parameters for modeling the entire training dataset. Inspired by a recent image prior technique for inpainting and denoising known as the Fields-of-Experts, a maximal clique or a set of adjacently connected local patches for each random training sample is used. In order to learn the complex representation, a BMM called the Fields-of-Gaussian (FoG) mixture model is proposed. The MM algorithm is then extended to learn the FoG mixture model. The effect is similar to taking the product of individual likelihood of each clique or patch within a maximal clique. Experimental results on the Multi-view Car pose estimation dataset showed that FoG significantly outperforms GMM in terms of mean Average Precision on object pose estimation for all cluster sizes.

3. The learning algorithm of variational inference using the Expectation - Expectation (EE) algorithm mainly require first computing the hyperparameter updates and then using it to update the expectation of the variational posteriors. There are two problems found. Firstly, the expectations of the variational posteriors are not defined but rather are indirectly defined using the expectations of the conjugate priors. There is a well known fact that the conjugate posterior will have an expectation expression similar to the conjugate prior. This fact is only useful if the expectation of the conjugate prior has a closed form. When the EE approach do not have the mathematical convenience of conjugate priors to perform learning, analytical solution cannot be obtained. The MM approach address this issue since it makes no assumption about conjugate convenience. Instead, it directly define the expectation of the variational posteriors using a MAP estimation on the variational posteriors distributions. Secondly, it is shown in

the experiments that when increasing the feature attribute size of datasets, some BMM may converge poorly when using variational inference due to computing expectation of the cluster assignments posteriors. This is observed when using Bayesian GMM to model image categorization datasets. Fortunately, the MM approach is shown to converge much faster than EE due to its unique solution. Finally, a theoretical comparison of EE and MM is made for solving Bayesian GMM. It is surprising that both solutions actually have nearly identical closed form expressions. This is despite their differences in theoretical concepts.

4. In Bayesian nonparametrics such as Dirichlet process mixture (DPM), learning is almost exclusive to either variational inference or Gibbs sampling. Yet variational inference is seldom mainstream in fast algorithms for DPM mainly due to high computational cost. Instead, most fast algorithms are largely based on MAP estimation of Gibbs sampling probabilities. However, they usually face intractable posterior and typically degenerate the conditional likelihood to overcome the inefficiency. Scalable variational inference such as stochastic variational inference do exist but they do not directly address the dependency on conjugate assumption previously encountered in contribution #3. This constitutes to the high cost often associated with variational inference. Inspired by fast DPM algorithms, MAP estimation is proposed for learning DPM. However, undefined variational posteriors of log expectation are encountered. This problem is overcome by the use of lower bounds. When the cluster assignment also uses a MAP estimation, the global objective is known as the maximization-maximization algorithm. A revisit to the concepts of variational inference led to the observation that the analytical solutions obtained by the proposed method are once again very similar to variational inference. Lastly, a comparison of the proposed fast approach to variational inference and fast algorithms on some UCI datasets and real datasets is made. Experimental results showed that the

method outperforms variational inference in terms of clustering accuracy, model selection and has comparable computational efficiency to fast DPM algorithms.

5. A recently proposed Sparse Coding based Fisher Vector extends traditional GMM based Fisher Vector with a sparse term. Experimental results revealed that the addition of this sparse term alone significantly outperforms GMM based Fisher Vector by almost 20% improvement on small sized datasets (15- Scene, Caltech-10) and up to 5% improvement on medium sized datasets (MIT-67). In the original work, Sparse Coding based Fisher Vector requires an off-the-shelf Sparse Coding solver. From a statistical perspective, an off-the-shelf solver may appear as a black-box. A more elegant way is to use a probabilistic model to learn Sparse Coding. A probabilistic model known as the Sparse Coding based GMM is proposed. It differs from GMM by an additional sparse coefficient hidden variable. The prior model of the sparse term is assumed Gaussian distributed for tractability. Inference of the model is performed by iteratively computing a set of closed-form equations obtained via variational method. Experimental results on several well-cited datasets show that the proposed probabilistic based solver obtained on-par learning performance to an off-the-shelf solver as far as Sparse Coding based Fisher Vector is concerned.

1.4 Organization of the Thesis

The thesis is organized as follows:

Chapter 2 covers a literature review of several research areas to establish a background for the major contributions in this thesis. Firstly, the classical inference technique for mixture models is presented and examined how it can be applied to solve more complex mixture models. Secondly, several approximating techniques for learning intractable posteriors (which usually arise from the

complexity of involving mixture models with prior distributions) are discussed. Thirdly, a review on some recent Bayesian Nonparametrics methods that specifically focus on reducing computational cost or overcoming implementation difficulty is made. Fourth, a discussion on how sparse coding can be interpreted both from a K-means perspective and a probabilistic account is given. Fifth, a review on Fisher Vector for feature representation in image categorization is made and some of its recent advancement discussed. Lastly, a review on using local structure information to improve the image representation is given.

Chapter 3 address the first objective of the thesis in section 1.2.1. by establishing the motivation of using the MM algorithm in general for inferencing BMM such as in the simple case of GMM. The proposed MM approach is theoretically and empirically compared to the EE approach, a widely popular inference technique in recent literature. Also, in the experiments the proposed method is compared with the modeling results of other recently proposed BMM on public image categorization datasets. The performance difference between a Bayesian approach and a maximum likelihood approach (no prior models are used) for GMM on a feature learning problem is distinguished. Lastly, the convergence and analytical proofs of the proposed method are also given.

Chapter 4 is a second attempt at addressing section 1.2.1 with three main differences. Firstly, the BMM used here is the more complex DPM rather than GMM. Although DPM and GMM use the same prior models for the mixture, the prior models are different for the cluster assignment. This will lead to distinctly different analytical solution and clustering performance. Secondly, the application here is more specific to data mining datasets rather than limited to image datasets in Chapter 3. That means the baselines used for comparison (also known as fast DPM algorithms in general) are more focused on lowering

computational cost. Lastly, this chapter presents another opportunity to check whether the comparison findings (theoretical and empirical) made between MM and EE in chapter 3 is consistent when extending BMM to stochastic process in chapter 4.

Chapter 5 address the second thesis objective explained earlier in section 1.2.2. Adding sparsity to the GMM mixture parameters can further improve the dataset modeling. This new application specific BMM is named as the Sparse Coding based GMM (sGMM). It is inspired by the recent proposed Sparse Coding based Fisher Vector (SC-FV). The application of sGMM here is to first fit its model to the dataset and then its parameters are used to compute SC-FV for image representation.

Chapter 6 is a work that is under the second objective. It is observed that most BMM for image modeling only consider a single image patch as a random training sample. Inspired by an image prior technique known as the Field-of-Experts in inpainting and denoising, this work attempt to preserve the local structure information found from adjacent patches of the single patch when modeling a GMM. It is called the Field-of-Gaussian mixture model and is used for image modeling. Experimental results showed that exploiting local structure information indeed significantly improve the overall image representation.

Chapter 7 concludes the thesis and also identified some potential research problems for further works.

2 Literature Review

2.1 Introduction

In this section, a literature review on several research topics which are relevant to the following chapters in this thesis is provided. The first research topic highlights on the maximum likelihood approach to mixture models, which is the basic approach. It is necessary as it demonstrates the differences between variational inference in chapter 3 and maximum likelihood for parameter estimation. The second topic mainly discusses about the four different types of variational posterior learning algorithms. Also, two recent works each employing a different learning algorithm are discussed in detail. This topic also supplements the background for chapter 3. The third topic raises issues on reducing computational cost or overcoming implementation difficulty for Bayesian Nonparametrics. Three types of BNPs are discussed in depth. In chapter 4, some of these BNPs are used as baseline for comparison. The fourth and fifth review topics should provide a good reference for chapter 5 concerning sparse coding and Fisher vector. The last topic is a brief review for some recent techniques that model images using local structure. Their approaches highlight the importance of preserving local structure in chapter 6.

2.2 Maximum Likelihood of Mixture Models

2.2.1 Classical EM algorithm for GMM

A classical method in machine learning is the maximum likelihood approach [6]. For an independent and identically distributed (i.i.d) observation $x = \{x_n\}_{n=1}^N \in \mathbb{R}^D$, the likelihood $\mathcal{L}$ is statistically modeled as the sample given the parameters θ as follows

$$\mathcal{L} = \prod_{n=1}^N p(x_n | \theta) \quad (2.1)$$

Since logarithm is monotonic, the log-likelihood is easier to work as follows

$$\ln \mathcal{L} = \sum_{n=1}^N \ln p(x_n | \theta) \quad (2.2)$$

Lastly, the parameters are estimated using the maximum likelihood

$$\theta^* = \arg \max_{\theta} \sum_{n=1}^N \ln p(x_n | \theta) \quad (2.3)$$

Often, modeling with complete observation alone is not sufficient. Instead, x can be considered as the incomplete observation and the complete observation is now a joint with a hidden variable $\{x, z\}$. This hidden variable is responsible for indicating where the data comes from. This is illustrated below in Figure 2.1.

Figure 2.1: (Left) A complete observation $\{x, z\}$ of three states which indicate where the samples came from. (Right) An incomplete observation, x which ignore the hidden state, z [6].

The statistical model of the complete observation is defined as

$$p(x, z) = p(x | z)p(z) \quad (2.4)$$

The formal definition for hidden state is $z_k \in \{1, 0\}$ subjected to $\sum_{k=1}^K z_k = 1$. A good statistical model for z_k is the multinomial distribution which can express the outcome of a category as a 1-of-K vector

$$p(z) = \mathcal{M}(z \mid \pi) = \prod_{k=1}^K (\pi_k)^{z_k} \quad (2.5)$$

The conditional term can be defined using a Gaussian distribution

$$p(x \mid z) = \prod_{k=1}^K \mathcal{N}(x \mid \mu_k, \Sigma_k)^{z_k} \quad (2.6)$$

By taking the logarithm of $p(x, z)$, the following is obtained

$$\begin{aligned} \ln p(x, z) &= \ln p(x \mid z) + \ln p(z) \\ &= \sum_{k=1}^K (\mathcal{N}(x \mid \mu_k, \Sigma_k) + \pi_k) z_k \end{aligned} \quad (2.7)$$

The marginal distribution of $p(x, z)$ is given as

$$\begin{aligned} \ln p(x) &= \sum_z \ln p(x \mid z) + \ln p(z) \\ &= \sum_{k=1}^K \mathcal{N}(x \mid \mu_k, \Sigma_k) \pi_k \end{aligned} \quad (2.8)$$

An expression for GMM is obtained in the above expression [6].

If GMM is expressed as a log-likelihood, the following is obtained

$$\ln \mathcal{L} = \sum_{n=1}^N \ln p(x_n \mid \theta) = \sum_{n=1}^N \sum_{k=1}^K \mathcal{N}(x_n \mid \mu_k, \Sigma_k) \pi_k \quad (2.9)$$

Next, it is possible to apply MLE to GMM

$$\theta^* = \arg \max_{\theta} \sum_{n=1}^N \ln p(x_n \mid \theta) \quad (2.10)$$

Taking the derivative of $\ln p(x \mid \theta)$ w.r.t to each parameter and setting to zero, the parameter estimation for GMM is obtained as follows (Maximization step)

$$\mu_k^* = \frac{\sum_{n=1}^N z_{nk} x_n}{\sum_{n=1}^N z_{nk}} \quad (2.11)$$

$$\Sigma_k^* = \frac{\sum_{n=1}^N z_{nk} (x_n - \mu_k^*) (x_n - \mu_k^*)^T}{\sum_{n=1}^N z_{nk}} \quad (2.12)$$

$$\pi_k^* = \frac{1}{N} \sum_{n=1}^N z_{nk} \quad (2.13)$$

The cluster assignment z_{nk} for each n sample and each k state is defined by normalizing the GMM log-likelihood (Expectation step)

$$z_{nk}^* = \frac{\mathcal{N}(x \mid \mu_k^*, \Sigma_k^*) \pi_k^*}{\sum_j \mathcal{N}(x \mid \mu_j^*, \Sigma_j^*) \pi_j^*} \quad (2.14)$$

The MLE of GMM is solved by first randomly initializing z_{nk} and computing eqn (2.11), (2.12), (2.13) and then using them to compute eqn (2.14). This is iterated until convergence i.e. the parameters values stop changing.

This MLE approach for the complete observation or GMM is also known as the Expectation-Maximization (EM) algorithm [23, 6, 43, 44, 69, 102, 40]. The MLE approach forms the basis of many statistical machine learning problems as will be seen again in the next section.

2.2.2 Supervised GMM

The illustration in Figure 2.2 below demonstrate the simple idea that unsupervised learning such as K-means excel at generalization or cluster separation but suffer from poor cluster purity [30]. Supervising learning such as SVM on the other hand do not generalize as well but has better purity due to available class labels. The problem in [30, 63, 17] is to gain both benefits using GMM and how to learn the objective function through a maximum likelihood approach.

Figure 2.2: (A) Original data. (B) Unsupervised clustering mainly focus on maximizing the between-class distance and minimizing the within-class distance, or the likelihood optimization w.r.t. the cluster means. It can generalize very well but it has poor purity. (C) Supervised clustering has good purity but poor generalization (D) Proposed supervised GMM tries to combine the best of both worlds, purity and likelihood [30].

The definition of purity is the negative logarithm of the entropy of cluster as follows

$$F(s_i) = -\ln \left\{ -\sum_j p(c_j | s_i, \Theta_i) \times \ln p(c_j | s_i, \Theta_i) \right\} + \phi \quad (2.15)$$

where the class label c given the cluster s is $p(c_j | s_i, \Theta_i) \propto \sum_k p(x_k | c_j)p(x_k | s_i)$. The reader can refer to [30] for the complete definition of $p(c_j | s_i, \Theta_i)$.

The likelihood of GMM is as follows

$$L(x) = \sum_{i=1}^N \ln p(x_i | \Theta) \quad (2.16)$$

Then, the joint objective function of likelihood and purity is a weighted sum as follows

$$J(\Theta) = (1 - \alpha)L(x) + \alpha F(s_i) \quad (2.17)$$

The maximum likelihood approach learns from the dataset, the optimal parameters $\Theta = \{\mu, \Sigma, w\}$ of a GMM

$$\Theta^* = \arg \max_{\Theta} J(\Theta) \quad (2.18)$$

By taking the partial derivatives of Θ^* , the optimal parameter estimation for $J(\Theta)$ is obtained as follows

$$\begin{aligned} \mu_i^* &\Rightarrow \frac{dJ(\Theta)}{d\mu_i} = 0 \\ \Sigma_i^* &\Rightarrow \frac{dJ(\Theta)}{d\Sigma_i} = 0 \\ w_i^* &\Rightarrow \frac{dJ(\Theta)}{dw_i} = 0 \end{aligned} \quad (2.19)$$

Since the additional of the entropy term further complicates the objective function over standard GMM. The reader can refer to [30] for full details on the mathematical derivation for the expression above. Instead, the purpose here is to firstly report how a direct inference such as maximum likelihood approach is applied to solve a complex GMM problem. Secondly, it is important to note that the supervised GMM model here do not use prior models for the parameters. For a Bayesian approach to the supervised GMM, it is unlikely the maximum likelihood approach will remain tractable. The supervised GMM above will be discussed again and compared to the proposed MM approach to Bayesian GMM in chapter 3.

2.2.3 GMM with Regularization

The Bayesian approach to GMM adds complexity to the true posterior due to additional prior models involved as later seen in section 2.3. Before the arrival of approximate inference such as Gibbs sampling or variational inference, past works such as Bayesian Regularization [83, 27] had to rely on maximum likelihood approach in order to learn Bayesian GMM.

The goal in Bayesian Regularization [83] is to find the maximum likelihood

of standard GMM with additional regularization ϕ in the expression as follows

$$\begin{aligned}
l &= \ln \left[\prod_{n=1}^N p(x_n \mid \mu_k, \lambda_k, k) \right] + \phi \\
&= \sum_{n=1}^N \sum_{k=1}^K \pi_k \ln p(x_n \mid k, \mu_k, \lambda_k) + \\
&\quad + \ln \mathcal{D}(\pi_k \mid \alpha_0) + \left[\ln \mathcal{N}(\mu_k \mid m_0, (\eta_0 \lambda_k)^{-1}) + \ln \mathcal{W}(\lambda_k \mid a_0, b_0) \right]
\end{aligned} \tag{2.20}$$

The authors apply a M-step of the above expression in a similar EM fashion to GMM, yielding the following point estimates

$$\begin{aligned}
\pi_k^* &= \frac{\sum_{n=1}^N z_{nk} + \alpha_0 - 1}{N + \sum_{k=1}^K \alpha_0 - K} \\
\mu_k^* &= \frac{\sum_{n=1}^N z_{nk} x_n + \eta_0 m_0}{\sum_{n=1}^N z_{nk} + \eta_0} \\
\lambda_k^* &= \frac{\sum_{n=1}^N z_{nk} + 2a_0 - d}{\sum_{n=1}^N z_{nk} (x_n - \mu_k^*) (x_n - \mu_k^*)^T + (\mu_k^* - m_0) (\mu_k^* - m_0)^T + 2b_0}
\end{aligned} \tag{2.21}$$

The E-step is identical to the EM of GMM

$$z_{nk} = \frac{1}{\mathbb{Z}} \pi_k p(x_n \mid k, \mu_k, \lambda_k) \tag{2.22}$$

Not surprisingly, deriving the above closed form solution will involve a lengthy analytical workout much like supervised GMM in the earlier section. It is not easy to generalize the framework to more complex cases such as DPM.

Also, from a Bayesian perspective, the authors' approach here is certainly not a true Bayesian approach which require

$$p(\theta \mid x) = \frac{p(x \mid \theta) p(\theta)}{p(x)} \tag{2.23}$$

since they are effectively performing maximum log likelihood estimates

$$l = \arg \max_{\theta} \ln \left[\prod_{n=1}^N p(x_n \mid \theta) p(\theta) \right] \tag{2.24}$$

A true Bayesian approach will have to deal with the evidence

$$p(x) = \int p(x | \theta)p(\theta)d\theta \quad (2.25)$$

which will involve integrating out the hidden variables.

Recent literature studies show that direct inference are no longer used as they are analytically intractable for most BMM cases [26, 25, 71, 72, 5, 9, 103, 80]. Instead, this thesis focus on approximate inference which is the more acceptable trend in recent machine learning.

2.3 Variational Inference of Bayesian Mixture Models

Approximating algorithm such as variational inference [6, 3, 10, 8, 18] or Gibbs sampling [77, 91] is usually employed when learning a Bayesian mixture model due to analytically intractable integral, arising from expectation functions. Variational inference is a modern alternative to Gibbs sampling due to faster convergence and comparable accuracy. Today, variational inference is a mainstream learner to numerous Bayesian mixture models ranging from non-Gaussian mixtures [26, 71, 11, 21] to nonparametric mixtures [9, 25, 58, 108] to hierarchical mixture [101, 85] and etc. The learning algorithm of variational inference [6] can be widely seen as first estimating the class assignment variable and then using it to estimate the parameters of the mixture model for each iteration till convergence. The estimate is performed by computing the expectations of the prior models. However, the learning algorithm is not necessarily exclusive to computing expectation. Welling and Kurihara saw other possible configurations that use different combinations of maximization and expectation on estimating the class assignment variable and mixture model parameters. They call it a family of alternating learners and variational inference is generalized as the expectation-expectation (EE) algorithm.

2.3.1 Variational Inference Framework

Variational inference [18, 6] approximates the intractable integral of the marginal posterior distribution

$$p(x) = \int p(x | \theta) p(\theta) d\theta \quad (2.26)$$

arising from computing the true posterior, $p(\theta | x) = \frac{p(x|\theta)p(\theta)}{p(x)}$. This approximation can be decomposed as a sum

$$\ln p(x) = \mathcal{L} + KL_{divergence} \quad (2.27)$$

where $\mathcal{L}$ is a lower bound on the joint distribution between observed and hidden variable. A tractable distribution $q(\theta)$ is used to compute $\mathcal{L}$

$$\mathcal{L} = \int q(\theta) \ln \left(\frac{p(x, \theta)}{q(\theta)} \right) d\theta \quad (2.28)$$

$$KL_{divergence} = - \int q(\theta) \ln \left(\frac{p(\theta|x)}{q(\theta)} \right) d\theta \quad (2.29)$$

When $q(\theta) \approx p(\theta|x)$, the Kullback-Leibler divergence is minimized and $\mathcal{L} \rightarrow \ln p(X)$. The tractable distribution $q(\theta)$ is also called the variational posterior distribution and assumes the following factorization

$$q(\theta) = \prod_i q(\theta_i) \quad (2.30)$$

The variational log posterior distribution is obtained by

$$\ln q(\theta_j) = E_{i \neq j} [\ln p(x, \theta_j)] + const. \quad (2.31)$$

2.3.2 Learning Algorithms for Variational Posteriors

Welling and Kurihara [57, 109] categorized a family of alternating learners for Bayesian posteriors that are analytically intractable. They are generalized as variants of variational inference and can be classified into Expectation - Expectation (EE), Expectation-Maximization (EM), Maximization-Expectation (ME), Maximization -Maximization (MM). They have a tradeoff between computational speed and accuracy.

In the variational EE algorithm a.k.a. standard variational inference [57, 109], the objective function involves alternating between the variational posterior distributions for learning

$$q(z) \propto \exp \left(E_{q(\theta)} [\ln p(x, \theta, z)] \right) \leftrightarrow q(\theta) \propto \exp \left(E_{q(z)} [\ln p(x, \theta, z)] \right) \quad (2.32)$$

When variational EM algorithm is applied to a variational inference framework, the objective can be seen as first maintaining a distribution over the assignment variable, and then computing the MAP estimate of the other variables. The objective function is given as

$$q(z) = p(z \mid \theta^*, x) \leftrightarrow \theta^* \approx \arg \max_{\theta} E_{q(z)} [\ln p(x, \theta, z)] \quad (2.33)$$

In the variational ME algorithm [57, 109], the main idea is to use point estimate for computing $E[z]$, while maintaining a distribution over the remaining parameters. The objective function is

$$z^* \approx \arg \max_z E_{q(\theta)} [\ln p(x, \theta, z)] \leftrightarrow q(\theta) = p(\theta \mid z^*, x) \quad (2.34)$$

The main interest in this thesis is the variational MM algorithm due to its efficiency. In MM, the objective is to perform updates by alternating between

the point estimates for θ^* and z^* as follows

$$z^* \approx \arg \max_z p(x, \theta^*, z) \leftrightarrow \theta^* \approx \arg \max_\theta p(x, \theta, z^*) \quad (2.35)$$

According to [109, 57], EE is the most accurate while MM is the fastest to compute. Their objective functions and comparisons are further explained in chapter 3 and 4.

2.3.3 Variational Maximization-Expectation Algorithm for Bayesian K-means

The objective function of the variational ME algorithm is to alternate between MAP estimating the assignment variable and maintaining distributions over the remaining variables for each iteration until convergence as follows

$$z^* \approx \arg \max_z E_{q(\theta)} [\ln p(x, \theta, z)] \leftrightarrow q(\theta) = p(\theta \mid z^*, x) \quad (2.36)$$

Bayesian K-means [109] use the above expression for learning a Bayesian GMM. First, the joint distribution of GMM is defined as

$$\begin{aligned} p(x, \mu, \Omega, z, \alpha) &= p(x \mid z, \mu, \Omega) p(z \mid \alpha) p(\alpha) p(\mu, \Omega) \\ &= \mathcal{N}(x \mid z, \mu, \Omega) \mathcal{M}(z \mid \alpha) \mathcal{D}(\alpha \mid \phi_0) \mathcal{N}(\mu \mid m_0, \xi_0 \Omega) \mathcal{W}(\Omega \mid \eta_0, B_0) \end{aligned} \quad (2.37)$$

and the marginal distribution is

$$\begin{aligned} p(x, z) &= \int \int \mathcal{N}(x \mid z, \mu, \Omega) \mathcal{M}(z \mid \alpha) \mathcal{D}(\alpha \mid \phi_0) \\ &\quad \mathcal{N}(\mu \mid m_0, \xi_0 \Omega) \mathcal{W}(\Omega \mid \eta_0, B_0) d\mu d\Omega d\alpha \end{aligned} \quad (2.38)$$

The variational posteriors to approximate the intractable integral is

$$q(\mu, \Omega, \alpha) = \prod_{k=1}^K \mathcal{D}(\alpha \mid \phi_k) \mathcal{N}(\mu_k \mid m_k, \xi_k \Omega_k) \mathcal{W}(\Omega_k \mid \eta_k, B_k) \quad (2.39)$$

and the output to the RHS of eqn (2.34) (hyperparameters updates) of ME gives

$$\begin{aligned}
\xi_k &= \xi_0 + N_k & m_k &= \frac{1}{\xi_k} (N_k \bar{x}_k + \xi_0 m_0) \\
\eta_k &= \eta_0 + N_k & \phi_k &= \phi_0 + N_k \\
B_k &= B_k + N_k S_k + \frac{N_k \xi_0}{\xi_k} (\bar{x}_k - m_0)(\bar{x}_k - m_0)^T
\end{aligned} \tag{2.40}$$

Next, the MAP estimate in the LHS of eqn (2.34) is solved by computing the following

$$\begin{aligned}
z^* &= \arg \min_z E_{q(\mu, \Omega, \alpha)} [\ln p(x, \mu, \Omega, z, \alpha)] \\
z_{nk}^* &= \arg \min_{z_{nk}} \left\{ \frac{\eta_k}{2} (x_n - m_k)^T B_k^{-1} (x_n - m_k) \right. \\
&\quad \left. + \frac{1}{2} \ln \det(B_k) + \frac{D}{2\xi_k} - \frac{1}{2} \sum_{d=1}^D \psi\left(\frac{\eta_k + 1 - d}{2}\right) - \psi(\phi_k) \right\} z_{nk}
\end{aligned} \tag{2.41}$$

The entire ME algorithm rely on computing eqn (2.40) and (2.41) for each iteration until convergence.

According to the authors' argument, Bayesian K-means is similar to the K-means objective as it involves minimizing the assignment variable and updating of the hyperparameter variables.

2.3.4 Approximating Non-iid Image Representation with the Variational Expectation-Expectation Algorithm

In Figure 2.3 below, it illustrate the key difference between a maximum likelihood (ML) and a Bayesian approach for GMM. In [16], the authors claim that ML approach suffers from the assumption that samples are identical and independently distributed. However, this is a poor assumption for the case of local image patches that constitutes an image, since there are **dependencies** amongst image patches.

The problem in the non-iid image representation [16] is to capture this dependency while keeping the image representation as an unordered set of image

Figure 2.3: The maximum likelihood approach to an image model treat the image as a set of identical and independently distributed image patches. This is a poor assumption since there are dependency between image patches [16].

Figure 2.4: Maximum Likelihood approach to the GMM (left), where the hidden variables λ, μ, π are treated as point estimates. The Bayesian approach to GMM (right) where the hidden variables λ, μ, π are now modeled using prior distributions with hyperparameters defined as α, a, b, m, β .

patches. A non-iid representation is essentially a BMM that i) model the hidden parameters as prior distributions and ii) by integrating out the hidden variables, the dependencies between image patches are captured. This is shown in Figure 2.4 below for a GMM.

The non-iid image representation is given as the following marginal distribution

$$p(x) = \sum_z \int \int \int p(x \mid \mu, \lambda, \pi) p(\pi) p(\mu, \lambda) d\mu d\lambda d\pi \quad (2.42)$$

The prior distribution for the hidden variables are

$$\begin{aligned}
p(x_n \mid \mu, \lambda, \pi) &= \sum \pi_k \mathcal{N}(x_n \mid \mu_k, \lambda_k^{-1}) \\
\mu &\sim \mathcal{N}(m, (\beta\lambda)^{-1}) \\
\lambda &\sim \mathcal{G}(\lambda \mid a, b) \\
\pi &\sim \mathcal{D}(\pi \mid \alpha) \\
w &\sim \mathcal{M}(w \mid \pi)
\end{aligned} \tag{2.43}$$

Variational posterior is used to approximate $p(x)$

$$q(\mu, \lambda, w, \pi) = q(\pi)q(\mu \mid \lambda)q(\lambda)q(w) \tag{2.44}$$

The learning of the above expression uses the standard variational inference aka the EE algorithm. The objective function of EE iteratively alternates between the cluster assignment and the remaining variables

$$\begin{aligned}
q(z) &\leftrightarrow q(\theta) \\
\Rightarrow E_{q(\theta)} [\ln p(x, \theta, z)] &\leftrightarrow E_{q(z)} [\ln p(x, \theta, z)]
\end{aligned} \tag{2.45}$$

The LHS of eqn (2.32) of EE (cluster assignment) is given below as

$$\begin{aligned}
\ln q(w) &= E_{q(\pi), q(\mu, \lambda)} [\ln p(x_n \mid \mu, \lambda, \pi)] \\
\Rightarrow \ln q(w_{nk}) &= \psi(\alpha_k) - \psi(\hat{\alpha}) + \frac{1}{2} [\psi(a_k) - \ln b_k] \\
&\quad - \frac{1}{2} \left[\frac{a_k}{b_k} (x_n - m_k)^2 + (\beta_k)^{-1} \right]
\end{aligned} \tag{2.46}$$

The RHS of eqn (2.32) of EE (hyperparameter updates) output the followings

$$\begin{aligned}
\alpha_k &= \alpha_0 + s0_k \\
\beta_k &= \beta_0 + s0_k \\
m_k &= \frac{1}{\beta_k} (s1_k + \beta_0 m_0) \\
a_k &= a_k + \frac{s2_k}{2} \\
b_k &= b_0 + \frac{1}{2} (\beta_0 m_0^2 + s2_k^2) - \frac{1}{2} \beta_k m_k^2 \\
s0_k &= \sum_{n=1}^N q(w_{nk}) \\
s1_k &= \sum_{n=1}^N q(w_{nk}) x_n \\
s2_k &= \sum_{n=1}^N q(w_{nk}) x_n^2
\end{aligned} \tag{2.47}$$

The entire EE algorithm for GMM rely on computing eqn (2.45) and (2.46) for each iteration until convergence.

In section 2.6.2 this work is extended to estimate parameters for a latent MoG based Fisher Vector.

2.4 Fast Algorithms of Bayesian Nonparametrics

In clustering, the number of clusters K to use is required to know beforehand. Bayesian Nonparameteric (BNP) does not have this issue as it can learn K directly from the dataset. However, one limiting factor of BNP is the expensive cost to compute. There are existing works that addressed this issue which is named as fast algorithms of BNP in this thesis. These works are dedicated to achieving fast computation of BNP and in most cases are willing to trade accuracy for efficiency. Also, for mathematical convenience they mostly restricted to conjugate distributions from the exponential family, notably the DPM.

Several works dedicated to fast algorithms of Dirichlet process mixture (DPM) have been proposed in the past. A common trait in these methods is that MAP estimate is preferred over the slower Gibbs sampling for cluster assignment. An early work is DPsearch [22] whereby the authors use MAP estimate on the true posterior. Due to intractable MAP solution, a tighter bound is used on the con-

ditional likelihood for efficiency. Similarly, SUGS [107, 20] use MAP estimation for cluster assignment by sequentially allocating new samples to clusters that locally maximizes the posterior of DPM. However, they replaced the DPM posterior with the posterior of partition model to avoid intractable solution. Inspired by K-means, the authors in DP-means [56] applied small variance asymptotics (SVA) to the prior model of DPM. MAP estimate on the probabilities of Gibbs sampling for cluster assignment then lead to an objective function that resembles K-means. In [13], a direct posterior approach is used but because MAP estimate on the cluster assignment posterior is intractable, by assuming SVA the authors similarly arrive at an identical objective function to DP-means. While SVA greatly reduce the complexity in [13, 56], it also rob away the “rich get richer” property in DPM. MAP-DPM [92] overcame the SVA reliance in [13] by observing that the conjugacy leads to the Student-T distribution for the cluster assignment posterior. Cluster assignment probabilities for Gibbs sampling is then computed using a simple MAP estimate on the Student-T distribution. Conversely, the speed of DPM can be improved by making variational inference based DPM more efficient. Stochastic variational inference [47, 105] looked at improving the scalability of variational inference by defining a new set of rules that allow local learning from sampled batches of the full dataset. In memorized online variational inference for DPM [48], the authors mainly aimed at improving the shortcomings of the stochastic variational inference such as requiring careful choices of batch size and learning rate. More crucially, the learning approach to variational posteriors [15, 85, 41, 71, 72, 26, 25] still remained largely unchanged from past works [6, 9].

2.4.1 DPsearch

Approximate inference of DPM using Gibbs sampling require drawing sample from the cluster assignment posterior and then estimate the assignment by re-

lating to the highest posterior probability. The author of DPsearch [22] finds it costly and instead proposed to use MAP estimation on the true posterior to find cluster assignment. On the other hand, the author also consider variational inference as difficult to implement and also computationally slow.

There are three scenarios presented in this work. i) admissible but inefficient, ii) partially admissible but faster iii) inadmissible but fastest. For each scenario, there is a newly defined score function g that estimate the cluster assignment $p(x, c)$ given a sample x

$$g(c^0, x) = \arg \max_c \ln p(x | c) + \ln p(c) \quad (2.48)$$

c^0 refers to the existing clusters. $p(x | c)$ refers to the marginal distribution of modeling x and $p(c)$ refers to the cluster probability. While $\max p(c)$ has a closed form solution, $\max p(x | c)$ does not. Due to intractable MAP solution, a tighter bound is used on the conditional likelihood for efficiency. The three scenarios each differs by altering the form

$$p(x | c) = \prod_{d \in c} H(x_{c=d}) \quad (2.49)$$

H refers to the cluster assignment.

In the first scenario, the ideal case but inefficient way is to assign every datapoint

$$p(x | c) = \prod_{k \in c^0} H(x_{c^0=k}) \quad (2.50)$$

A heuristic approach is used in scenario two. To simply put it, the approach avoid re-assigning the existing datapoint. Only the new datapoint will receive a decision whether to move it to a new cluster or assign it to one of the existing

clusters as follows

$$\begin{aligned}
p(x \mid c) = \prod_{k \in c^0} H(x_{c^0=k}) \prod_{n=N^0+1}^N \max_{1 \leq k \leq K^0+1} \max_{c:} H(x_n \mid x_{c_{1:n-1}=c_m}) \\
c \mid N^0 = c^0, c_n = k
\end{aligned} \tag{2.51}$$

The last scenario is an over simplistic approach which assign a new cluster for every new datapoints. The conditional likelihood is as follows

$$p(x \mid c) = \prod_{k \in c^0} H(x_{c^0=k}) \prod_{n=N^0+1}^N H(x_n) \tag{2.52}$$

2.4.2 DPmeans

DPmeans [56] is a recently proposed nonparametric clustering. It is implemented using a Gibbs sampling algorithm [77]. By assuming noninformative prior for mean or small variance asymptotics (SVA), a closed form Gibbs probability of starting a new cluster or being assigned to an existing cluster can be obtained. Next, small variance assumption $\sigma \rightarrow 0$ results in the probability of starting a new cluster to be dependent on a single parameter λ . The DPmeans objective is found to be identical to K-means except for the addition of λ . Initially, DPmeans starts with one cluster and λ controls when new cluster are introduced. λ is chosen either heuristically or manually for each dataset [56, 92]. Experimental results on UCI datasets show that DP-means is at least on par or better than K-means [56]. The implementation of DPmeans is summarized in the algorithm below.

Algorithm 2.1 DPmeans [56]

Input: X Output: j, η, T Initialization: $\lambda, \eta_1, t = 1$ For $i = 1 : I$ For $n = 1 : N$ 1. compute $dist_n = \sum_t \|x_n - \eta_t\|^2$ 2. if $dist_n > \lambda$, $t = t + 1$ and $\eta_{t+1} = x_n$

End

3. compute the cluster assignment j_n and recompute the mean η_{dt}

$$j_n = \arg \max_{j_n} \sum_{t=1}^T \sum_{d=1}^D \ln \mathcal{N}(x_{nd} \mid \eta_{dt}) \cdot j_{nt} \quad (2.53)$$

$$\eta_{dt} = \frac{\sum_{n=1}^N j_{nt} \cdot x_{nd}}{\sum_{n=1}^N j_{nt}} \quad (2.54)$$

End

2.4.3 MAP-DPM

While SVA greatly reduce the complexity in [13, 56], it also rob away the “rich get richer” property in DPM. MAP-DPM [92] overcame the SVA reliance in [13] by observing that the conjugacy leads to the Student-T distribution for the cluster assignment posterior. Cluster assignment probabilities for Gibbs sampling is then computed using a simple MAP estimate on the Student-T distribution.

MAP-DPM [92] is centered around a decision from two sets of Student-T distributions

$$p(x_{n+1} \mid z) = St \left(x_{n+1} \mid m_k, \frac{a_k c_k}{b_k (c_k + 1)}, 2a_k \right) \quad (2.55)$$

$$p(x_{n+1} \mid \theta_0) = St \left(x_{n+1} \mid m_0, \frac{a_0 c_0}{b_0 (c_0 + 1)}, 2a_0 \right) \quad (2.56)$$

Student-T is basically a 3 parameter Gaussian with a parameter for degree of freedom. This parameter a_0 allows Student-T to flex between heavy tail and

Gaussian like. When a_0 is large enough, Student-T is approximately a Gaussian distribution.

The two Student-Ts competes for maximum likelihood score (or minimum negative likelihood). One Student-T $p(x_{n+1} | z)$ computed with means and precisions from each existing cluster and the other Student-T $p(x_{n+1} | \theta_0)$ takes the mean and precision of the new sample.

For each sample, the parameters are updated through a set of rules similar to those in [6, 77, 109]

$$\begin{aligned}
m_k &= \frac{c_0 m_0 + N_k \bar{x}_k}{c_0 + N_k} \\
c_k &= c_0 + N_k \\
a_k &= a_0 + \frac{1}{2} N_k \\
b_k &= b_0 + \frac{1}{2} \sum_{j: z_n=k} (x_j - \bar{x}_k)^2 + \frac{c_0 N_k (\bar{x}_k - m_0)^2}{2c_k}
\end{aligned} \tag{2.57}$$

The link between DPM and Student-T comes from the fact that in a Gibbs sampler the Gaussian-Gamma prior parameters are integrated out for efficiency, leading to a Student-T distribution.

The probability of assigning the new sample to a new cluster or to an existing set of clusters is computed by the two Student-T distribution. A parameter α affects the cluster assignment of starting a new cluster.

$$p(z_{n+1} = k | z, \theta_k) = N_k p(x_{n+1} | z) \tag{2.58}$$

$$p(z_{n+1} = K + 1 | \alpha, \theta_0) = \alpha p(x_{n+1} | \theta_0) \tag{2.59}$$

A similar MAP-DPM like algorithm has also appear in [41] for background subtraction. Like λ in DP-means, α is hand selected for convenience. But unlike DP-means and MAP-DPM the proposed approach does not have to refer to ground truth on class size to specifically hand select a certain parameter value

for each dataset.

2.5 Relating Sparse Coding to Bayesian Mixture Mode

2.5.1 From K-means to Sparse Coding

The K-means objective function has frequently reappeared as a starting point to many algorithms that inspired this thesis, such as the VLAD feature [50], DP-means [56], Sparse Coding based Fisher Vector [70], variational inference of Bayesian GMM [6] and etc.

The authors in [110] has presented a unique perspective of deriving sparse coding from K-means. While not truthfully following the authors' words, here the relationship between sparse coding and K-means is demonstrated by borrowing some idea from GMM [67]. Firstly, K-means has the objective function for cluster assignment

$$z_{nk}^* = \arg \max_{z_{nk}} \sum_{k=1}^K -\frac{1}{2} (x_n - B_k^*)^2 \cdot z_{nk} \quad (2.60)$$

Where $z_{nk} \in \{1, 0\}$. The above is interpreted as following: For each independent sample x_n , find the best k^{th} cluster that will minimize the distance between the sample and the cluster. Note that B_k^* is affected by the choice of z_{nk} , and the asterisk in B_k^* refers to the optimal value of B_k found from earlier iteration.

Without changing meaning to the expression above, a new coefficient term u_{nk} can be introduced as follows

$$z_{nk}^* = \arg \max_{z_{nk}} \sum_{k=1}^K -\frac{1}{2} (x_n - B_k^* u_{nk}^*)^2 \cdot z_{nk} \quad (2.61)$$

subjected to $Card(B_n) = 1, u_{nk} > 0$

$Card(u_n) = 1$ is a cardinality constraint where there is only one nonzero element for a $1 - of - K$ vector u_n .

The constaint can be relaxed by introducing a $L1$ -norm of u_n such that only

a small number of nonzero element exist for u_n in the following

$$z_{nk}^* = \arg \max_{z_{nk}} \sum_{k=1}^K -\frac{1}{2} (x_n - B_k^* u_{nk}^*)^2 \cdot z_{nk} - \lambda \sum_{k=1}^K |u_{nk}^*| \quad (2.62)$$

Notice that B_k^* and u_{nk}^* have not been solved yet. These terms are in fact solved in an EM algorithm like manner. First solve for B_k^* and u_{nk}^* then use them to solve z_{nk}^* . Alternative, the expression above can be re-expressed to obtain a sparse coding objective function as follows

$$\min_{B, \mu} \sum_{n=1}^N \left\| x_n - \sum_{k=1}^K B_k u_{nk} \right\|^2 + \lambda \sum_{k=1}^K |u_{nk}| \quad (2.63)$$

The primitive way to optimization is done by fixing B while solving μ and alternating in an iterative manner much like the EM algorithm. There is a dedicated research field to this problem known as non-negative matrix factorization and it is beyond the scope of the thesis.

2.5.2 Probabilistic Interpretation of Sparse Coding

There exist a classical work [82] that explains sparse coding from a probabilistic approach. An image can be modeled as a linear superposition of k independent basis B with coefficient u and some additive noise ϵ as follows

$$x_n = \sum_{k=1}^K B_k u_{nk} + \epsilon \quad (2.64)$$

Below in Figure 2.5, an illustration of what a bank of basis ($K = 256$) or image clusters may look like is given

Figure 2.5: Illustration of a set of image basis for sparse coding

Thus, by considering the earlier formulation, the approximation for an arbitrary image patch can be illustrated in Figure 2.6 as follows

$$\text{Image Patch} \approx 0.8 * \text{Basis 1} + 0.3 * \text{Basis 2} + 0.5 * \text{Basis 3}$$

Figure 2.6: Sparse coding representation of an image patch. Noise is assumed zero here.

By re-arranging the image model above and normalizing it, a Gaussian distribution with noise $\epsilon = 2\sigma^2$ is obtained as follows

$$p(x_n | B, u) \propto \exp \left(-\frac{\left(x_n - \sum_{k=1}^K B_k u_{nk} \right)^2}{2\sigma^2} \right) \quad (2.65)$$

However since only $p(x | B)$ is useful, by integrating out u or taking the marginal distribution of $p(x | B, u)$, the following can be obtained

$$p(x | B) = \int p(x | B, u) p(u) du \quad (2.66)$$

Note that a prior model $p(u)$ is now required. Typically, the prior is defined using a Laplace distribution

$$p(u) \approx \exp(-\lambda |u|) \quad (2.67)$$

Because the integral is intractable to work with, the MAP approximation below

is used

$$\begin{aligned} p(x | B) &\approx p(x | B, u^*)p(u^*) \\ u^* &= \arg \max_u \ln p(x | B, u) + \ln p(u) \end{aligned} \quad (2.68)$$

Thus, the full probabilistic expression of $p(x | B)$ is

$$p(x | B) = \arg \max_u -\frac{1}{2\sigma^2} \sum_{n=1}^N \left(x_n - \sum_{k=1}^K B_k u_{nk} \right)^2 - \lambda \sum_{k=1}^K |u_{nk}| \quad (2.69)$$

The earlier probability model of $p(x | B)$ is conditional on B . For the marginal distribution of $p(x)$, it can be obtained by

$$p(x) = \int \int p(x | B, u) p(u) p(B) du dB \quad (2.70)$$

Taking noninformative prior for $p(B)$, the expression is reduced to

$$\begin{aligned} p(x) &= \arg \max_B \ln p(x | B) \\ &= \arg \max_{B, u} -\frac{1}{2\sigma^2} \sum_{n=1}^N \left(x_n - \sum_{k=1}^K B_k u_{nk} \right)^2 - \lambda \sum_{k=1}^K |u_{nk}| \end{aligned} \quad (2.71)$$

The full probabilistic expression of $p(x)$ can be re-expressed as

$$\min_{B, u} \sum_{n=1}^N \left\| x_n - \sum_{k=1}^K B_k u_{nk} \right\|^2 + \lambda \sum_{k=1}^K |u_{nk}| \quad (2.72)$$

Thus, an identical form to sparse coding using a probabilistic approach is now obtained in the above expression. This section also serve as a basis for chapter 5.

2.6 Bayesian Mixture Model based Fisher Vector

In the era of deep learning [55, 100, 99, 89, 86, 36, 98, 93, 35, 51], Fisher Vector (FV) [88, 96] remains one of the contemporary method in image categorization and retrieval for a couple of good reasons. One of the few reasons is the

low computation cost involved and it does not require large amount of training dataset by comparison. Some authors [36, 89, 99, 100, 16] have also looked into complementing FV with deep learning. FV represents an image by sum-pooling the gradients of GMM parameters. The main advantage of such representation is comparative or better performance to a Bag-of-Word vector while keeping the GMM computation cost low. Because FV computes a high dimensional representation, VLAD [50] is proposed to reduce FV's computation cost by keeping the model parameters and dimension to a minimal. The Bayesian approach to GMM defines additional prior distributions [6]. Cinbis et al [16] took this opportunity to further extend FV's model parameters with additional hyperparameters. Recently, Liu et al [70] replaces the GMM model in FV with a Sparse Coding model [82].

2.6.1 GMM based Fisher Vector

Fisher Vector [88, 96] was the leading technique in the PASCAL VOC challenge [24] before deep ConvNet [55] appeared in 2012.

There is a close working relationship between Fisher Vector and GMM. The origin of the Fisher Vector is traced back to Fisher Kernel as follows [16, 88, 96]

$$K(x, x') = g(x)^T I^{-1} g(x')$$

where $g(x) = \nabla_{\theta} p(x)$ is the Fisher score or the gradient of the data log-likelihood, $p(x)$ and $I = E_x [g(x)g(x)^T]$ is the Fisher information matrix. Due to the convenience of feature mapping as inner product space, a simpler definition for Fisher Kernel can be used as follows

$$\phi(x) = I^{-\frac{1}{2}} g(x) \tag{2.73}$$

Then by dropping I for tractability, Fisher Kernel is approximately given as

$$\phi(x) \approx \nabla_{\theta} p(x) \quad (2.74)$$

Next, for a GMM with diagonal covariance to model $p(x)$, the definition for the Fisher kernel or the maximum likelihood of GMM for the mean and variance parameter $\theta \in \{\mu, \sigma\}$ is as follows

$$\mu_k = \sum_{n=1}^N p(z_k | x_n) \cdot \frac{x_n - u_k}{\sigma_k} \quad (2.75)$$

$$\sigma_k = \sum_{n=1}^N p(z_k | x_n) \cdot \left[\frac{(x_n - u_k)^2}{\sigma_k^2} - 1 \right] \quad (2.76)$$

It is also mentioned in literature that the above Fisher Kernel represents the first and second order statistics. For a complete representation aka Fisher Vector it is simply a vector concatenation of the following

$$F = [\cdots \mu_k \cdots, \cdots \sigma_k \cdots]^T \quad (2.77)$$

VLAD [50] is in fact a simple case of Fisher Vector with only first order statistics for F and hard assignment for $p(z_k | x_n)$. The normalization of Fisher Vector or VLAD is straightfoward and not discussed here. The estimation of the cluster assignment $p(z_k | x_n)$ is further discussed in the GMM related topics in the thesis.

2.6.2 Extending Fisher Vector with Prior Models

It has been shown in experiements that only the first and second order statistics of Fisher Vector are used $\theta \in \{w, \mu, \sigma\}$ as seen below.

∇	MAP (in %)
w	46.9
μ	57.9
σ	59.6
$\mu\sigma$	61.8
w μ	58.1
w σ	59.6
w $\mu\sigma$	61.8

Table 2.1: Accuracy of using the zero, first and second (w, μ, σ respectively) order statistics of Fisher Vector on the PASCAL VOC 2007 dataset [96].

A recent study known the as Latent Mixture of Gaussian (LatMoG) [16] extends the statistics to include another 5 statistics for a Fisher Vector as seen in Table 2.1. These are the hyperparameters $\vartheta = \{\alpha, a, b, m, \beta\}$ of the Dirichlet, Gaussian and Gamma prior models for the variational posteriors of $q(\pi), q(\mu), q(\lambda)$.

The explanation of variational posteriors is left to chapters 3 and 4 of this thesis and the reader can refer to [16] for the gradient estimation of the hyperparameters. Thus, the LatMoG based Fisher Vector is given as

$$F = [\pi, \mu, \lambda, \alpha, a, b, m, \beta]^T \quad (2.78)$$

In their experiments, it is reported that the LatMoG based Fisher Vector significantly outperforms MoG based Fisher Vector for all cluster sizes below in Table 2.2 which also shows that models using a Bayesian approach is superior to Maximum Likelihood approach.

Method	32	64	128	256	512	1024
MoG	49.1	51.4	53.1	54.3	55.0	55.9
LatMoG	52.9	55.9	56.6	58.6	59.5	60.2

Table 2.2: Accuracy comparison between MoG based Fisher Vector and the LatMoG based Fisher Vector on the PASCAL VOC 2007 dataset [16].

2.6.3 When Fisher Vector meets ConvNet

The neural network based netVLAD [2] is actually a diversion from the application of BMM to Fisher Vector. Nevertheless, the idea is worth studying. Before discussing about netVLAD, lets revisit VLAD and ConvNet [55] separately.

In ConvNet there are some basic concepts such as convolutional layer, max pooling layer, ReLu and fully connected layer. In particular, the max pooling layer employs a simple operation known as max pooling in general as seen in Figure 2.7 below.

Figure 2.7: Example of max pooling in ConvNet. After the max operation, it is followed by a downsampling.

The main idea of Fisher Vector using first order statistics or equivalently VLAD or is simply to take the total metric difference between each k^{th} cluster and all the training samples

$$v_k = p(z_k | x_n) \cdot \sum_{n=1}^N (x_n - c_k) \quad (2.79)$$

as seen in Figure 2.8 below

Figure 2.8: Illustration of VLAD computation. μ_k refers to the cluster center and the blue dash refers to the metric distance of each sample from μ_k .

Where $p(z_k | x_n) \in \{1, 0\}$ is the membership of sample x_i being inside cluster c_k . Next, by concatenating each vector from each cluster, the full VLAD representation of an image is obtained

$$V = [v_1 \cdots, v_k, \cdots v_K]^T \quad (2.80)$$

Instead of max pooling in ConvNet, netVLAD uses VLAD pooling. Also, the hard assignment a_k can be re-expressed using a Gaussian soft assignment

$$p(z_k | x_n) = \frac{\exp(-\alpha \|x_n - c_k\|^2)}{\sum_{k'} \exp(-\alpha \|x_n - c_{k'}\|^2)} \quad (2.81)$$

After some simplification, the expression can take the form below

$$p(z_k | x_n) = \frac{\exp(w_k^T x_n + b_k)}{\sum_{k'} \exp(w_{k'}^T x_n + b_{k'})} \quad (2.82)$$

Where $w_k = 2\alpha c_k$ and $b_k = -\alpha \|c_k\|^2$ refers to the weights and bias of a convolutional filter. Effectively, the new expression of $p(z_k | x_n)$ decouples w_k and b_k from c_k . This also means that for a pretrained VLAD model, the decoupling allows the flexibility of fine tuning on other datasets. The learning of w_k and b_k uses Stochastic Gradient Descend (SGD) while c_k still rely on the standard EM or K-means algorithm. The experimental result in [2] claims that netVLAD

outperforms standard ConvNet features by 50% accuracy on place recognition.

2.7 Strategies for Modeling Image using Local Structure

Without access to huge computational power and millions of training images for training a deep ConvNet, past works often rely on human expertise to come up with ad-hoc strategies to improve image modeling. In this section, some non BMM related works are discussed that address preserving local structure information in image modeling.

Several works have addressed the loss of image spatial information in a histogram based representation. The classical method in this context is the spatial pyramid representation (SPR) [59] by partitioning an image into smaller quadrants each represented by a separate Bag-of-Word (BoW). SPR is considered a global partitioning approach since the input to the unsupervised learner remains unchanged from BoW. For local partitioning, pairwise methods such as SVN-KNN [111] uses a SVM classifier with a kernel function in terms of pairwise distance between query feature and nearest training feature. Thus, the SVM classifier would favour predictions with lesser spatial distance between query and training feature. Another work that uses a similar idea is [68]. Each feature is now represented by a word and its location in the image. A new descriptor is computed based on these two information by modeling the occurrence of every possible word pairing.

All the above are considered post dictionary learning approach of capturing spatial information. In order to overcome scalability issue inherent in these approach, the pairwise codebook method [75] starts from feature level by concatenating descriptors of neighboring image patch and treating them as a new image feature. Working from a post dictionary learning approach is the same as partitioning a single feature codebook, whereas, learning directly from feature level [75] allows distribution of pairwise feature to be learnt. The disadvantage is

that this approach is too naive and suffers from a significant doubling increases of descriptor length.

Two recent works are reviewed below to highlight the importance of modeling image using local structure.

2.7.1 A Local Pairwise Codebook

The well cited work Spatial Pyramid Representation (SPR) [59] is proven to globally preserve spatial information in image representation. For preserving local spatial information, a naive approach is to build a pairwise codebook [19] that treat each pair of spatially close descriptors as a training sample.

There are two main issues according to the authors of the Local Pairwise Codebook (LPC) approach [75]. i) when the codebook size increases, the possible pair of spatially close descriptors grows quadratically. ii) Most pairwise codebook do not exploit the distribution of pairs, since each descriptor in the pairing use the same universal codebook. This is seen in Figure 2.9.

Figure 2.9: Clustering on spatially close descriptors. (left) A naive pairwise codebook approach. (right) The LPC approach. LPC exhibit a more data distribution guided clustering than the grid like clustering used by a naive approach.

LPC overcome the two issues by concatenating each pair of spatially close descriptors d_i, d_j as a single joint descriptor. Then a clustering on such dataset is used to form a codebook. The LPC feature is further computed with image

coordinates x, y as follows

$$f(i, j) = \left(\frac{x_i + x_j}{2}, \frac{y_i + y_j}{2}, [d_i, d_j] \right) \quad (2.83)$$

The scene categorization experimental results of LPC is compared with SPR in terms of accuracy as follows in Table 2.3

		15 Scenes	67 Indoors
Methods	K		
Sgl	400	74.75±0.66	23.89±0.92
	800	75.62±0.58	26.11±0.98
	1600	76.67±0.61	27.67±1.03
	3200	76.13±0.51	28.77±1.85
LPC	800	76.78±0.53	27.30±1.44
	1600	78.42±0.23	29.71±0.70
	3200	79.76±0.38	32.35±0.68

Table 2.3: Preserving spatial information, local (LPC) vs global (Sgl).

From the results, it is evident that preserving local spatial information significantly improve image categorization.

2.7.2 Detecting Spatially Close Keypoints

An interesting problem is how to detect a pair of spatially close descriptors or keypoints. This is addressed in the Spatially Stable Interest Point Pairs (SSIPP) work of [113] which build upon every SIFT keypoint detection. In simple terms, SSIPP models the co-occurrence of spatially close keypoints within a predefined spatial radius. It is visualized below in Figure 2.10

Figure 2.10: Spatially Stable Interest Point Pairs [113]. A radius is used to determine the interest points that are spatially close.

The the scale invariant radius is defined using the scale of each interest point S_{center} and a co-occurence control parameter λ .

$$Dis = S_{center} \cdot \lambda \quad (2.84)$$

S_{center} is obtained from the scale output of each SIFT keypoint as $P(S, D, O)$ where they refer to scale, descriptor and orientation respectively. [113] employ a metric for comparing two SSIPP using appearance similarity. This metric is used to cluster SSIPPs such that their co-occurence can be found. The experimental results of the SSIPP method in Figure 2.11 shows that the traditional image model using traditional codebooks (A1) is poorer than methods that employs pairwise codebooks (A2, A3). Moreover, the proposed method A2 is as efficient as A1 despite losing out mAP wise to A3.

Figure 2.11: Mean Average Precision on authors' dataset of image re-ranking (A1) Traditional visual word tree. (A2) pair-wise visual word tree. (A3) Visual Rank algorithm [52]

2.7.3 Deep Learning Approach

The more recent development in modeling local structure is in fact a deep learning based approach known as the recurrent neural network (RNN) and its variant the Long Short Term Memory. In RNN [39], multi-layer perceptrons are used in loops to store memories as seen in the figure below. Essentially, the output of the next MLP rely on information or hidden units from the previous MLP and the current input.

In image modeling, RNN has been applied to scan across the entire image to learn local structure information [104].

Figure 2.12: Each multi-layer perceptron is used in loop to form a neural network known as the recurrent neural network. The purpose of such a model is the ability to store memories.

3 A New Method for Learning Bayesian Mixture Model

3.1 Introduction

In [109, 57], the authors suggest that there is a trade-off between computational efficiency and accuracy in the various objective functions for learning variational posteriors in variational inference. The maximization-maximization (MM) objective is the most efficient to compute while the Expectation-Expectation (EE) objective is the most accurate. Inspired by [109, 57], the MM objective has very recently been applied to various Bayesian mixture models such as Gaussian mixture [67], Field-of-Gaussian based Gaussian mixture [64], and sparse coding based Gaussian mixture [66]. Despite the recent success of MM, it suffers from several problems: i) MM is not as widely known as EE. ii) it is very rare to find any theoretical study comparing MM to EE. iii) the computational efficiency and accuracy of MM is seldom compared to EE. Hence it is difficult to justify or convince the use of MM over a mainstream learner such as EE or even Gibbs sampling.

In this chapter: i) The algorithms for computing EE and MM on a simple Bayesian GMM case are given in detail. ii) A theoretical comparison of MM with EE is also given and it is shown that the analytical solution of MM is in fact identical to EE despite their difference in approach. Thus, in theory MM should not be inferior to EE in terms of accuracy. iii) Two set of experiments are ran to compared the efficiency and accuracy of MM and EE. iv) A discussion on how a Bayesian approach using MM compares to a maximum likelihood approach such as K-means and EM is made. This is also shown on two sets of experiments. v) An explanation on how a Bayesian approach can be related to a maximum likelihood approach using MM is made. vi) Finally, a discussion on how to check the convergence of MM guaranteed via lower bound is given.

In the first experiment, MM is used to perform unsupervised classification on the Caltech [28] and Scene Categories [59] datasets. The efficiency and accuracy of MM with EE is compared. It is observed that MM is typically faster to compute but have slightly poorer accuracy than EE. Both methods are also comparable to state-of-the-art result on these datasets. In the second experiment, MM is used to perform unsupervised feature learning on two object classification datasets, Multi-view Car [84] and Caltech [28]. Experimental results show that a Bayesian approach (such as MM) consistently obtain better mean average precision score than a maximum likelihood approach (such as K-means and EM) at unsupervised feature learning for all dictionary sizes.

3.2 Related Work

In [67] the authors used MM to learn a Bayesian GMM for computing BoW and VLAD. It is compared with a non Bayesian approach such as Kmeans. Their finding on two image classification datasets shows that in general, the MM learner outperforms kmeans at unsupervised feature learning for all dictionary sizes. This experiment is repeated using the same setup. The main difference is that the Bayesian GMM here is not restricted to shared precision. Hence, the best results in this work at 0.74 outperforms the 0.68 result reported in [67].

MM can also be applied to more a complex model such as the Field-of- Gaussian mixture in [64]. In this case, instead of learning random image patches for an object category dataset, the authors use random local neighborhood of patches. Each neighborhood contains several adjacently connected image patches. The approach is effectively taking a product of individual likelihood for MM learning. The additional complexity due to the product of likelihoods (Gaussian) can also be straightforward learnt using MM. The Field-of- Gaussian was also found to outperform the standard GMM.

Another recent work using MM is the sparse coding based GMM in [66].

In this work, the standard GMM is extended to include an additional sparse coefficient variable. Instead of using an off-the-shelf learner such as the feature-sign algorithm, MM is used to perform learning for the hidden variables of sparse coding based GMM. The learning performance of MM is reported to be at least on par or better than the feature-sign algorithm on the sparse coding based Fisher vector [70] representation as reported in their experiments.

3.3 Background

3.3.1 Gaussian Mixture Model

GMM models a set of N observed variable denoted as $x = \{x_n\}_{n=1}^N \in \mathbb{R}^D$ with a set of hidden variables, $\theta = \{\mu, \tau, z, \pi\}$. The dimension of each observed instance is denoted D . The cluster assignment is denoted $z = \{z_n\}_{n=1}^N$ where z_n is a $1 - of - K$ binary vector, subjected to $\sum_{k=1}^K z_{nk} = 1$ and $z_{nk} \in \{0, 1\}$. Mixture component is denoted $\pi = \{\pi_k\}_{k=1}^K \in \mathbb{R}^D$. Diagonal covariance $\Sigma = \sigma^2 I$ is assumed and precision is denoted as $\tau = \frac{1}{\sigma^2}$ where $\sigma = \{\sigma_k\}_{k=1}^K \in \mathbb{R}^D$. Cluster mean is denoted as $\mu = \{\mu_k\}_{k=1}^K \in \mathbb{R}^D$.

3.3.2 Bayesian Approach

In a Bayesian approach to GMM (Figure 3.1), each hidden variable is now modeled by a prior distribution as follows [6, 74]

$$\begin{aligned}
x \mid z, \mu, \tau &\sim \mathcal{N}(\mu, \tau^{-1})^z \\
\mu \mid \tau &\sim \mathcal{N}(m_0, (\lambda_0 \tau)^{-1}) \\
\tau &\sim \text{Gam}(a_0, b_0) \\
z \mid \pi &\sim \text{Mult}(\pi) \\
\pi &\sim \text{Dir}(\alpha_0)
\end{aligned} \tag{3.1}$$

Figure 3.1: Graph model of Bayesian GMM. The unshaded circles in boxes are hidden variables while the variables outside the boxes are hyperparameters. The shaded circle refers to the observed variable. The arrows show the relationships between variables and hyperparameters.

and their joint distribution can be factorized as

$$p(x, z, \pi, \mu, \tau) = p(x \mid z, \mu, \tau)p(z \mid \pi)p(\pi)p(\mu \mid \tau)p(\tau) \quad (3.2)$$

The hyperparameters $\alpha_0, a_0, b_0, m_0, \lambda_0$ are usually empirically determined or assumed noninformative.

3.3.3 Variational Inference and Learning Objective

Variational inference [6, 18] approximates the intractable integral of the marginal distribution

$$p(x) = \int p(x \mid \theta)p(\theta)d\theta \quad (3.3)$$

This approximation can be decomposed as a sum

$$\ln p(x) = \mathcal{L} + KL_{divergence} \quad (3.4)$$

where $\mathcal{L}$ is a lower bound on the joint distribution between observed and hidden variable. A tractable distribution $q(\theta)$ is used to compute $\mathcal{L}$ and $KL_{divergence}$

$$\mathcal{L} = \int q(\theta) \ln \left(\frac{p(x, \theta)}{q(\theta)} \right) d\theta \quad (3.5)$$

$$KL_{divergence} = - \int q(\theta) \ln \left(\frac{p(\theta|x)}{q(\theta)} \right) d\theta \quad (3.6)$$

When $q(\theta) = p(\theta|x)$, the Kullback-Leibler divergence is removed and $\mathcal{L} = \ln p(X)$. The tractable distribution $q(\theta)$ is also called the variational posterior distribution and assumes the following factorization

$$q(\theta) = \prod_i q(\theta_i) \quad (3.7)$$

Variational log posterior distribution is expressed as

$$\ln q(\theta_j) = E_{i \neq j} [\ln p(x, \theta_i)] + \text{const.} \quad (3.8)$$

Welling and Kurihara [109, 57] categorized a family of alternating learners for variational posteriors that are analytically intractable to learn. They can be generalized as variants of variational inference and can be classified into Expectation Expectation (EE), Expectation Maximization (EM), Maximization Expectation (ME) and Maximization Maximization (MM). They have a tradeoff between computational speed and accuracy.

The objective function of EE is

$$E[z] \propto \exp(E_{\Theta} [\ln p(x, \Theta, z)]) \leftrightarrow E[\Theta] = \int \Theta p(\Theta; \bar{z}) d\Theta \quad (3.9)$$

where $\bar{z}$ refers to the hyperparameters computed using the local statistics of $E[z]$.

The main interest here is the MM algorithm due to its simplicity. In MM,

the goal is to alternatively compute the MAP estimates for $\ln q[\Theta]$ and $\ln q[z]$. The objective function is

$$E[z] \approx \arg \max_z E_{\Theta} [\ln p(x, \Theta, z)] \leftrightarrow E[\Theta] \approx \arg \max_{\Theta} E_z [\ln p(x, \Theta, z)] \quad (3.10)$$

In the following sections, a theoretical and experimental comparisons between the EE and MM approach in eqn (3.9), (3.10) for learning a Bayesian GMM is made. It shall be seen that between the two learning objective, EE is the most accurate while MM is the fastest to compute.

3.3.4 Variational Learning of Bayesian GMM

In variational inference the learner is categorized as the EE algorithm. Learning is performed by updating the expectation of its variational posterior for every iteration till convergence. Usually, conjugate priors are assumed to conveniently exploit moments of well-defined prior distributions (ie. no need to compute expectations). Before computing these moments, the hyperparameters of the conjugate priors must be updated by the sample estimates of $E[z]$ for each iteration.

The main objective of variational inference is to train a model by computing the expectation of each variational posterior distribution for each iteration until convergence. Defining $\Theta = \{\mu, \tau, \pi\} \in \theta$, the expectation of $q(z)$ is

$$\begin{aligned} E[z_{nk}] &\propto \exp \{E_{\Theta} [\ln p(x, \Theta, z)]\} \\ &= \frac{1}{\mathbb{Z}} \exp \left\{ E_{\mu_k, \tau_k, \pi_k} [\ln p(x_n | \mu_k, \tau_k, z_{nk}) + \ln p(z_{nk} | \pi_k)] \right\} \\ &= \frac{1}{\mathbb{Z}} \exp \left\{ \frac{1}{2} E[\ln \tau_k] - \frac{1}{2} E[\tau_k] (x_n - E[\mu_k])^2 + E[\ln \pi_k] \right\} \end{aligned} \quad (3.11)$$

(where $\mathbb{Z}$ is a normalization term). For each iteration of $E[z_{nk}]$, it is required to compute the expectations of other variational distributions from previous iteration. Like most EE algorithms, prior distributions are restricted to exponential

family. Doing so, EE avoids directly deriving the expectations by conveniently using statistical moments of well defined prior distributions instead. In the case of Gamma, Gaussian and Dirichlet, their moments are given as

$$E[\ln \tau_k] = \psi(a_k) - \ln(b_k) \quad (3.12)$$

$$E[\tau_k] = \frac{a_k}{b_k} \quad (3.13)$$

$$E[\mu_k] = m_k \quad (3.14)$$

$$E[\ln \pi_k] = \psi(\alpha_k) - \psi\left(\sum_{j=1}^K \alpha_j\right) \quad (3.15)$$

(ψ refers to digamma function). Before computing these moments, their hyperparameters need to be updated using previous iteration value of $E[z_{nk}]$ as follows [6, 16]

$$a_k = a_0 + \frac{1}{2} \sum_{n=1}^N E[z_{nk}] \quad (3.16)$$

$$b_k = b_0 + \frac{1}{2} \left(\sum_{n=1}^N (x_n - m_k)^2 E[z_{nk}] + \lambda_0 (m_k - m_0)^2 \right) \quad (3.17)$$

$$\lambda_k = \lambda_0 + \sum_{n=1}^N E[z_{nk}] \quad (3.18)$$

$$m_k = \frac{1}{\lambda_k} \left(\lambda_0 m_0 + \sum_{n=1}^N E[z_{nk}] x_n \right) \quad (3.19)$$

$$\alpha_k = \alpha_0 + \sum_{n=1}^N E[z_{nk}] \quad (3.20)$$

(Note that the definition of b_k here is different from [16]). The EE objective for learning a Bayesian GMM [6] is summarized in Algorithm 3.1.

Algorithm 3.1 Variational inference of Bayesian GMM [6]

Input: x

Output: $E[\ln \pi_k], E[\mu_k], E[\ln \tau_k], E[\tau_k], E[z_{nk}]$

Initialization:

1. Randomly initialize the expectations $E[z_{nk}]$
2. Initialize the hyperparameters $m_0, \lambda_0, a_0, b_0, \alpha_0$

Compute until convergence:

1. Update the hyperparameters:

$$\begin{aligned} a_k &= a_0 + \frac{1}{2} \sum_{n=1}^N E[z_{nk}] \\ b_k &= b_0 + \frac{1}{2} \left(\sum_{n=1}^N (x_n - m_k)^2 E[z_{nk}] + \lambda_0 (m_k - m_0)^2 \right) \\ \lambda_k &= \lambda_0 + \sum_{n=1}^N E[z_{nk}] \\ m_k &= \frac{1}{\lambda_k} \left(\lambda_0 m_0 + \sum_{n=1}^N E[z_{nk}] x_n \right) \\ \alpha_k &= \alpha_0 + \sum_{n=1}^N E[z_{nk}] \end{aligned}$$

2. Update the expectations:

$$\begin{aligned} E[\ln \tau_k] &= \psi(a_k) - \ln(b_k) \\ E[\tau_k] &= \frac{a_k}{b_k} \\ E[\mu_k] &= m_k \\ E[\ln \pi_k] &= \psi(\alpha_k) - \psi(\sum_{j=1}^K \alpha_j) \\ E[z_{nk}] &= \frac{1}{Z} \exp \left\{ \frac{1}{2} E[\ln \tau_k] - \frac{1}{2} E[\tau_k] (x_n - E[\mu_k])^2 \right. \\ &\quad \left. + E[\ln \pi_k] \right\} \end{aligned}$$

End

3.4 Proposed Method, MM-GMM

The proposed method for learning GMM using the MM objective function in eqn (3.10) is known as the MM-GMM. The three main steps are as follows:

- i) define variational posteriors for all hidden variables
- ii) moments approximated using mode of distribution (MAP), $E[\theta] \approx \arg \max_{\theta} \ln q(\theta)$. $E[z_{nk}]$ is MAP estimated as well.
- iii) use of lower bound for the moments, $\ln(E[\Theta]) \geq E[\ln \Theta]$ for computing cluster assignment, $E[z_{nk}]$.

Further details of the above steps are given in the following sections 3.3.1 to 3.3.3. The proposed method for learning Bayesian GMM is summarized in Algorithm 3.2.

Algorithm 3.2 Proposed method, MM-GMM.

Input: x

Output: $E[\mu_k], E[\tau_k], E[\pi_k], E[z_{nk}]$

Initialization:

1. Randomly initialize the expectations $E[\mu_k], E[z_{nk}], E[\pi_k]$
2. Initialize the hyperparameters $m_0, \lambda_0, a_0, b_0, \alpha_0$

Compute until convergence

$$E[\tau_k] = \frac{\frac{1}{2} \sum_{n=1}^N E[z_{nk}] + (a_0 - 1)}{b_0 + \frac{1}{2} \left\{ \sum_{n=1}^N (x_n - E[\mu_k])^2 E[z_{nk}] + \lambda_0 (E[\mu_k] - m_0)^2 \right\}}$$

$$E[\mu_k] = \frac{\sum_{n=1}^N x_n E[z_{nk}] + \lambda_0 m_0}{\sum_{n=1}^N E[z_{nk}] + \lambda_0}$$

$$E[z_{nk}] = \arg \max_{z_{nk}} \left\{ \ln E[\pi_k] + \frac{1}{2} \ln E[\tau_k] - \frac{1}{2} E[\tau_k] (x_n - E[\mu_k])^2 \right\} z_{nk}$$

$$E[\pi_k] = \frac{\sum_{n=1}^N E[z_{nk}] + (\alpha_0 - 1)}{\sum_{j=1}^K \sum_{n=1}^N E[z_{nj}]}$$

End

3.4.1 Defining Variational Posteriors

The first task common to EE and MM is to assume the following mean field factorization for tractability [6]

$$q(z, \pi, \mu, \tau) = \prod_{n=1}^N \prod_{k=1}^K q(z_{nk}) q(\pi_k) q(\mu_k, \tau_k) \quad (3.21)$$

The variational posterior of $\ln q(\theta)$ is found by keeping terms related to θ in $p(x, z, \pi, \mu, \tau)$ while the rest are absorbed into a constant

$$\begin{aligned} \ln q(\tau_k) = E_{z_{nk}, \mu_k} [\ln p(x_n | z_{nk}, \mu_k, \tau_k) + \ln p(\mu_k | \tau_k) \\ + \ln p(\tau_k)] + \text{const.} \end{aligned} \quad (3.22)$$

$$\begin{aligned} \ln q(\mu_k) = E_{z_{nk}} [\ln p(x_n | z_{nk}, \mu_k, \tau_k) + \ln p(\mu_k | \tau_k)] \\ + \text{const.} \end{aligned} \quad (3.23)$$

$$\ln q(\pi_k) = E_{z_{nk}} [\ln p(z_{nk} | \pi_k) + \ln p(\pi_k)] + \text{const.} \quad (3.24)$$

$$\begin{aligned} \ln q(z_{nk}) = E_{\mu_k, \tau_k, \pi_k} [\ln p(x_n | z_{nk}, \mu_k, \tau_k) + \ln p(z_{nk} | \pi_k)] \\ + \text{const.} \end{aligned} \quad (3.25)$$

In variational inference, all the above expressions are not used except for the discrete hidden variable, $\ln q(z_{nk})$. However, in the proposed method these expressions are required as explained in the next section.

3.4.2 Approximating Expectation of Variational Posteriors

The EE algorithm in variational inference mainly rely on already defined statistical moments of prior distributions such as Gaussian, Gamma and Dirichlet. Due to the mathematical convenience of conjugacy, the statistical moments (or expectations) of the above variational posteriors expression in eqn (3.22) to (3.24) takes similar functional forms as the priors. This has been explained earlier

in section 3.3.4. Thus, the expectation of variational posteriors in variational inference are not directly derived but “borrowed”.

MM can be seen as a variant of EE where instead of “borrowing” the expectation of variational posterior, the mode of variational posterior is directly used to derive analytical solutions as follows

$$E[\theta] \approx \arg \max_{\theta} \ln q(\theta) \quad (3.26)$$

MAP estimation approximate the expectations of variational log posteriors. Their closed-form expressions can be obtained as follows (detailed in section 3.5.1)

$$\begin{aligned} E[\mu_k] &\approx \arg \max_{\mu_k} \ln q(\mu_k) \\ &= \frac{\sum_{n=1}^N x_n E[z_{nk}] + \lambda_0 m_0}{\sum_{n=1}^N E[z_{nk}] + \lambda_0} \end{aligned} \quad (3.27)$$

$$\begin{aligned} E[\mu_k] &\approx \arg \max_{\mu_k} \ln q(\mu_k) \\ &= \frac{\sum_{n=1}^N x_n E[z_{nk}] + \lambda_0 m_0}{\sum_{n=1}^N E[z_{nk}] + \lambda_0} \end{aligned} \quad (3.28)$$

$$\begin{aligned} E[\pi_k] &\approx \arg \max_{\pi_k} \ln q(\pi_k) \text{ s.t. } \sum_k \pi_k = 1 \\ &= \frac{\sum_{n=1}^N E[z_{nk}] + (\alpha_0 - 1)}{\sum_{j=1}^K \sum_{n=1}^N E[z_{nj}]} \end{aligned} \quad (3.29)$$

$$\begin{aligned} E[\tau_k] &\approx \arg \max_{\tau_k} \ln q(\tau_k) \\ &= \frac{\frac{1}{2} \sum_{n=1}^N E[z_{nk}] + a_0 - 1}{b_0 + \frac{1}{2} \left(\sum_{n=1}^N (x_n - \mu_k)^2 E[z_{nk}] + \lambda_0 (\mu_k - m_0)^2 \right)} \end{aligned} \quad (3.30)$$

3.4.3 Jensen's Inequality for Cluster Assignment

In EE, the cluster assignment is computed as $E[z_{nk}] \propto \exp\{\ln q(z_{nk})\}$. But for MM, $E[z_{nk}]$ is now estimated using MAP estimation

$$\begin{aligned} E[z_{nk}] &\approx \arg \max_{z_{nk}} \ln q(z_{nk}) \\ &\approx \arg \max_{z_{nk}} \left\{ E[\ln \pi_k] + \frac{1}{2} E[\ln \tau_k] - \frac{1}{2} E[\tau_k] (x_n - E[\mu_k])^2 \right\} z_{nk} \end{aligned} \quad (3.31)$$

Jensen's inequality is used to simplify $E[z_{nk}]$ as follows

$$\begin{aligned} E[\ln \tau_k] &\geq \ln E[\tau_k] \\ E[\ln \pi_k] &\geq \ln E[\pi_k] \end{aligned} \quad (3.32)$$

Thus, $E[z_{nk}]$ can now be computed in terms of $E[\tau_k], E[\pi_k], E[\mu_k]$ as seen below

$$\arg \max_{z_{nk}} \left\{ \ln E[\pi_k] + \frac{1}{2} \ln E[\tau_k] - \frac{1}{2} E[\tau_k] (x_n - E[\mu_k])^2 \right\} z_{nk} \quad (3.33)$$

3.5 Proposed Theoretical Analysis

3.5.1 Proof to Analytical Solution of Proposed Method

Here, the derivation for $\hat{\tau}_{dk}$ and $\hat{\mu}_{dk}$ is detailed. For the purpose of brevity, the subscript in τ_{dk} and μ_{dk} are dropped. Given the variational posterior distribution, $\ln q(\tau)$:

$$\begin{aligned} \ln q(\tau) &= E_z[\ln p(X | \mu, \tau, Z) + \ln \mathcal{N}(\mu | \mu_0, (\lambda\tau)^{-1}) \\ &\quad + \ln \mathcal{G}(\tau | a_0, b_0)] + \text{const.} \\ &= \sum_{n=1}^N \frac{1}{2} E[z_n] \ln(\tau) - \sum_{n=1}^N \frac{\tau}{2} (x_n - \mu)^2 E[z_n] - \frac{(\mu - \mu_0)^2}{2(\lambda\tau)^{-1}} \\ &\quad + (a_0 - 1) \ln(\tau) - b_0 \tau + \text{const.} \end{aligned} \quad (3.34)$$

Next, the derivative of $\ln q(\tau)$ is as follows

$$\begin{aligned} \frac{\partial \ln q(\tau)}{\partial \tau} &= \frac{1}{2} \sum_{n=1}^N \frac{E[z_n]}{\tau} - \frac{1}{2} \sum_{n=1}^N (x_n - \mu)^2 E[z_n] \\ &\quad - \frac{1}{2} \lambda (\mu - \mu_0)^2 + \frac{a_0 - 1}{\tau} - b_0 \end{aligned} \quad (3.35)$$

and by setting $\frac{\partial \ln q(\tau)}{\partial \tau} = 0$ and re-arranging, the following solution is obtained as seen in Algorithm 3.2

$$\hat{\tau} = \frac{\frac{1}{2} \sum_{n=1}^N E[z_n] + a_0 - 1}{b_0 + \frac{1}{2} \left(\sum_{n=1}^N (x_n - \mu)^2 E[z_n] + \lambda (\mu - \mu_0)^2 \right)} \quad (3.36)$$

Likewise, for variational posterior distribution $\ln q(\mu)$

$$\begin{aligned} \ln q(\mu) &= E_z [\ln p(X \mid \mu, \tau, Z) + \ln \mathcal{N}(\mu \mid \mu_0, (\lambda \tau)^{-1})] \\ &\quad + \text{const.} \\ &= \frac{\tau}{2} \left(- \sum_{n=1}^N (x_n - \mu)^2 E[z_n] - \lambda (\mu - \mu_0)^2 \right) \\ &\quad + \text{const.} \end{aligned} \quad (3.37)$$

By taking the derivative of $\ln q(\mu)$,

$$\frac{\partial \ln q(\mu)}{\partial \mu} = \tau \left(\sum_{n=1}^N (x_n - \mu) E[z_n] - \lambda (\mu - \mu_0) \right) \quad (3.38)$$

and finally by setting $\frac{\partial \ln q(\mu)}{\partial \mu} = 0$ and re-arranging, the next solution in Algorithm 3.2 is obtained below

$$\hat{\mu} = \frac{\sum_{n=1}^N x_n \cdot E[z_n] + \lambda \mu_0}{\sum_{n=1}^N E[z_n] + \lambda} \quad (3.39)$$

3.5.2 Convergence of Proposed Method

In the proposed approach for solving Bayesian GMM using the MM algorithm, an emphasis is on the computational efficiency. For completeness, a discussion on how to obtain convergence by checking the lower bound for each iteration is

as follows

$$\begin{aligned}
\mathcal{L} &\geq \int \int \int \int q(z, \mu, \tau, \pi) \ln \left[\frac{p(x, z, \mu, \tau, \pi)}{q(z, \mu, \tau, \pi)} \right] dz d\mu d\tau d\pi \\
&= E [\ln p(x, z, \mu, \tau, \pi)] - E [\ln q(z, \mu, \tau, \pi)] \\
&= E [\ln p(x | z, \mu, \tau)] + E [\ln p(z | \pi)] + E [\ln p(\pi)] \\
&\quad + E [\ln p(\mu, \tau)] - E [\ln q(z)] - E [\ln q(\pi)] - E [\ln q(\mu, \tau)]
\end{aligned} \tag{3.40}$$

In the lower bound expression above, each expectation function is taken with respect to all of the hidden variables and is simplified using Jensen's inequality as follows

$$\begin{aligned}
&E [\ln p(x | z, \mu, \tau)] \\
&= E \left[\sum_{k=1}^K \sum_{n=1}^N \ln \mathcal{N}(x_n | \mu_k, \tau_k^{-1})^{z_{nk}} \right] \\
&= \sum_{k=1}^K \sum_{n=1}^N \left\{ \frac{1}{2} E [\ln \tau_k] - \frac{E[\tau_k]}{2} (x_n - E [\mu_k])^2 \right\} E [z_{nk}]
\end{aligned} \tag{3.41}$$

$$\begin{aligned}
E [\ln p(z | \pi)] &= E \left[\sum_{k=1}^K \sum_{n=1}^N z_{nk} \ln \pi_k \right] \\
&= \sum_{k=1}^K \sum_{n=1}^N E [z_{nk}] E [\ln \pi_k]
\end{aligned} \tag{3.42}$$

$$\begin{aligned}
E [\ln p(\pi)] &= E \left[\sum_{k=1}^K \ln \text{Dir}(\pi_k | \alpha_0) \right] \\
&= (\alpha_0 - 1) \sum_{k=1}^K E [\ln \pi_k]
\end{aligned} \tag{3.43}$$

$$\begin{aligned}
&E [\ln p(\mu, \tau)] \\
&= E \left[\sum_{k=1}^K \ln \mathcal{N}(\mu_k | m_0, (\lambda_0 \tau_k)^{-1}) \text{Gam}(\tau_k | a_0, b_0) \right] \\
&= \sum_{k=1}^K \left(-\frac{(E[\mu_k] - m_0)^2}{2(\lambda_0 E[\tau_k])^{-1}} + (a_0 - 1) E [\ln \tau_k] - b_0 E [\tau_k] \right)
\end{aligned} \tag{3.44}$$

$$\begin{aligned}
E [\ln q(z)] &= \sum_{n=1}^N \sum_{k=1}^K \left\{ \ln E [\pi_k] + \frac{1}{2} \ln E [\tau_k] \right. \\
&\quad \left. - \frac{E[\tau_k]}{2} (x_n - E [\mu_k])^2 \right\} E [z_{nk}]
\end{aligned} \tag{3.45}$$

$$\begin{aligned}
E [\ln q(\pi)] &= E \left[\sum_{n=1}^N \sum_{k=1}^K E [z_{nk}] \ln \pi_k \right. \\
&\quad \left. + (\alpha_0 - 1) \sum_{k=1}^K \ln \pi_k \right] \\
&= \sum_{n=1}^N \sum_{k=1}^K E [z_{nk}] \ln E [\pi_k] + (\alpha_0 - 1) \sum_{k=1}^K \ln E [\pi_k]
\end{aligned} \tag{3.46}$$

$$\begin{aligned}
& E [\ln q(\mu, \tau)] \\
&= E \left[\sum_{k=1}^K \left(\sum_{n=1}^N \left\{ \frac{\ln \tau_k}{2} - \frac{\tau_k}{2} (x_n - E[\mu_k])^2 \right\} E[z_{nk}] \right. \right. \\
&\quad \left. \left. - \frac{(E[\mu_k] - m_0)^2}{2(\lambda_0 \tau_k)^{-1}} + (a_0 - 1) \ln \tau_k - b_0 \tau_k \right) \right] \tag{3.47} \\
&= \sum_{k=1}^K \left(\sum_{n=1}^N \left\{ \frac{E[\ln \tau_k]}{2} - \frac{E[\tau_k]}{2} (x_n - E[\mu_k])^2 \right\} E[z_{nk}] \right. \\
&\quad \left. - \frac{(E[\mu_k] - m_0)^2}{2(\lambda_0 E[\tau_k])^{-1}} + (a_0 - 1) E[\ln \tau_k] - b_0 [E\tau_k] \right)
\end{aligned}$$

In practice, it is not necessary to compute lower bound as the convergence can be visualized by checking the changes in weight of each cluster using the equation below, as seen in the experiments later on

$$\pi_k = \frac{\sum_{n=1}^N E[z_{nk}]}{\sum_{j=1}^K \sum_{n=1}^N E[z_{nj}]} \tag{3.48}$$

3.5.3 Comparison to Variational Inference

In MM, the mode of $\ln q(\theta)$ is used for updating $E[\theta]$ instead of computing the actual expectation function. By doing so, the stochastic behavior for the hidden variables (especially $E[z_{nk}]$) is lost. However, computation load is also reduced. Under certain models eg. exponential family of distribution, the expectation of the distribution is also identical to the mode, MM is identical to EE. This naive assumption is possibly the main difference between EE and MM. The generic algorithm for learning MM is $E[z] \leftrightarrow E[\Theta]$.

Perhaps the most interesting finding in this chapter is the fact that both MM and EE are theoretically related. This is despite the differences in their analytical approaches (MM rely on MAP approximation, while EE computes statistical moments via updated hyperparameters). This is true for the Bayesian GMM study in this thesis.

Comparison of $E[\mu_k]$:

A sharp reader may have noticed the similarity between the hyperparameter update in Algorithm 3.1

$$m_k = \frac{1}{\lambda_k} \left(\lambda_0 m_0 + \sum_{n=1}^N E[z_{nk}] x_n \right) \quad (3.49)$$

and the MAP approximated moment in Algorithm 3.2 below.

$$E[\mu_k] = \frac{\sum_{n=1}^N x_n E[z_{nk}] + \lambda_0 m_0}{\sum_{n=1}^N E[z_{nk}] + \lambda_0} \quad (3.50)$$

By substituting the hyperparameter update below to m_k , an identical expressions for both m_k and $E[\mu_k]$ is obtained.

$$\lambda_k = \lambda_0 + \sum_{n=1}^N E[z_{nk}] \quad (3.51)$$

Comparison of $E[\pi_k]$:

A Dirichlet distribution is given as

$$\pi_k \sim Dir(\alpha_k) \quad (3.52)$$

The statistical moment is given as

$$E[\pi_k] = \frac{\alpha_k}{\sum_{j=1}^K \alpha_j} \quad (3.53)$$

The corresponding hyperparameter update in Algorithm 3.1 is given as

$$\alpha_k = \alpha_0 + \sum_{n=1}^N E[z_{nk}] \quad (3.54)$$

If the hyperparameter update is substituted to $E[\pi_k]$, the following is obtained

$$E[\pi_k] = \frac{\alpha_0 + \sum_{n=1}^N E[z_{nk}]}{\sum_{j=1}^K \left(\alpha_0 + \sum_{n=1}^N E[z_{nj}] \right)} \quad (3.55)$$

If α_0 is constrained to zero in the denominator in the above expression, then the EE derived solution is almost identical to the below MM derived solution in

Algorithm 3.2

$$E[\pi_k] = \frac{\sum_{n=1}^N E[z_{nk}] + (\alpha_0 - 1)}{\sum_{j=1}^K \sum_{n=1}^N E[z_{nj}]} \quad (3.56)$$

Comparison of $E[\tau_k]$:

For a Gamma distribution shown below

$$\tau_k \sim \text{Gam}(a_k, b_k) \quad (3.57)$$

the statistical moment is given as

$$E[\tau_k] = \frac{a_k}{b_k} \quad (3.58)$$

Now for Algorithm 3.1, the hyperparameters are

$$a_k = a_0 + \frac{1}{2} \sum_{n=1}^N E[z_{nk}] \quad (3.59)$$

$$b_k = b_0 + \frac{1}{2} \left(\sum_{n=1}^N (x_n - m_k)^2 E[z_{nk}] + \lambda_0 (m_0 - m_k)^2 \right) \quad (3.60)$$

By expressing the statistical moment in terms of the hyperparameter updates, EE can derive the following solution

$$E[\tau_k] = \frac{a_0 + \frac{1}{2} \sum_{n=1}^N E[z_{nk}]}{b_0 + \frac{1}{2} \left(\sum_{n=1}^N (x_n - m_k)^2 E[z_{nk}] + \lambda_0 (m_k - m_0)^2 \right)} \quad (3.61)$$

By substituting $E[\mu_k] = m_k$, it is observed that the above expression has an identical form to the corresponding expression found in Algorithm 3.2 below

$$E[\tau_k] = \frac{\frac{1}{2} \sum_{n=1}^N E[z_{nk}] + (a_0 - 1)}{b_0 + \frac{1}{2} \left(\sum_n (x_n - E[\mu_k])^2 E[z_{nk}] + \lambda_0 (E[\mu_k] - m_0)^2 \right)} \quad (3.62)$$

Comparisons of $E[z_{nk}]$, $E[\ln \tau_k]$ and $E[\ln \pi_k]$:

Theoretically, since $z_{nk} \in \{1, 0\}$ the expected value of z_{nk} and the maximum value of z_{nk} should be identical when the dataset has sufficient statistics as sample size, $N \rightarrow \infty$.

Recall that in the case of MM algorithm, by taking Jensen's inequality

$$\begin{aligned} E[\ln \tau_k] &\geq \ln E[\tau_k] \\ E[\ln \pi_k] &\geq \ln E[\pi_k] \end{aligned} \tag{3.63}$$

additional computation for the moments approximation can be to skipped.

From the analysis above, when applying the MM algorithm to infer a Bayesian model, it turns out that the solution of the MM algorithm is somehow combining both the hyperparameter updates and the statistical moments of the EE algorithm as one entity. This phenomenal is a coincidence as clearly no hyperparameters updates were involved and no knowledge of any statistical moments (of prior distributions) were ever used in the MM algorithm.

3.5.4 Comparison to Maximum Likelihood

An intuitive way to explain the relationship between a Bayesian approach and a ML approach to Bayesian GMM using the MM algorithm is as follows. MM is governed by a prior belief λ_0 which determines how much trust is placed on the prior models in a Bayesian approach. For zero prior belief $\lambda_0 = 0$, MM reduces to the ML approach of GMM with $b_0 = 0$ as follows

$$\begin{aligned} E[\mu_k] &= \frac{\sum_{n=1}^N x_n E[z_{nk}] + \lambda_0 m_0}{\sum_{n=1}^N E[z_{nk}] + \lambda_0} \\ &\Rightarrow \frac{\sum_{n=1}^N x_{nd} E[z_{nk}]}{\sum_{n=1}^N E[z_{nk}]} \end{aligned} \tag{3.64}$$

$$E[\tau_k] = \frac{\frac{1}{2} \sum_{n=1}^N E[z_{nk}] + (a_0 - 1)}{b_0 + \frac{1}{2} \left\{ \sum_{n=1}^N (x_n - E[\mu_k])^2 E[z_{nk}] + \lambda_0 (E[\mu_k] - m_0)^2 \right\}}$$

$$\Rightarrow \frac{\sum_{n=1}^N E[z_{nk}]}{\sum_{n=1}^N (x_n - E[\mu_k])^2 E[z_{nk}]} \quad (3.65)$$

Also, a larger prior belief leads to fewer mixture components that contribute to the dataset while a smaller belief allows more Gaussians to capture the dataset. This can be seen mathematically in the equation above when λ_0 increases, a smaller $E[\tau_k]$ or larger variance ($\sigma^2 = \frac{1}{\tau}$) is obtained. Hence, the estimated clusters will end up with larger within-cluster distance and vice versa.

3.6 Experiments

The MM algorithm for learning Bayesian GMM is compared on two different problems.

In the first problem, Bayesian GMM is used to model class separation. One inherent characteristic of Bayesian GMM is its ability to predict class number. This means that class label is not required for learning the class separation. This problem is known as model selection or unsupervised classification. The objective is to compare the performance of MM for learning a Bayesian model with state-of-the-art results.

In the second problem, Bayesian GMM is used to build an image dictionary for a Bag-of-Word image representation. This is commonly known as unsupervised feature learning. The goal is to compare the learning performance of Bayesian approach and maximum likelihood (ML) approach such as K-means and the Expectation-Maximization algorithm (EM). Lastly, the Bayesian approach is compared with a ML approach and it is further explained how the analytical form of MM algorithm allows easy interpretation of how both Bayesian and ML approaches are related to each other.

3.6.1 Unsupervised Image Classification on Object Categories

A Bayesian GMM is trained as a classifier for both the object and scene classification datasets as seen in Table 3.1. The learner of variational inference (EE) is compared to MM-GMM (MM) using criteria such as the predicted model, accuracy and convergence time. K refers to the truncation level also set in [25]. $K = 15$ is fixed in both datasets.

In the Caltech dataset [28], the train and test are setup according to [25]. For the attribute, a BoW image representation followed by a PCA reduction is used. The K-means used in the BoW is 1000 and it is reduced to a 256 image vector using PCA. This setup is ran for both the EE and MM algorithm. In a practical scenario, a finite number of iterations is limited for both methods. In this experiment, this number is set to 20 as in the tabulated results. In Table 3.2, EE and MM are given 10 runs and the average is taken. Also, the best hyperparameters that appeared in Algorithm 3.1 and 3.2 are empirical selected.

It is observed that in Table 3.2, that EE is only marginally more accurate than MM but takes twice as long to compute which is within expectation. The confusion matrices for EE and MM are shown in Table 3.3 and 3.4 respectively. However, the model prediction of EE is somehow poorer than MM. The reason is mainly due to insufficient learning, resulting in poorer convergence for EE. The value of $E[z_n]$ in EE shows that it tend to contain insignificant but non-zero values for most of the clusters. In comparison, MM only compute one non-zero value for $E[z_n]$ and many insignificantly weighted clusters can be pruned away early on. However, for EE the problem can be elevated by increasing the number of iterations eg. 100 as shown in Figure 3.2. For MM, increasing the number of iterations only improve the convergence slightly. For unsupervised classification, a threshold is set on the cluster weights generated by both EE and MM to predict model. In Caltech4, the common threshold is fixed to 0.05.

Also, in the bottom row of Figure 3.2, the two plots computes the negative

lower bound of MM in eqn (3.40) and EE [6] for each iteration . It is seen that the negative lower bound of both methods does not drop and approach saturation after around 10 iterations. In fact, for this particular dataset, the accuracy of MM does not differ much after 10 iterations. This is also true for EE.

In the experiments baseline methods using infinite generalized Dirichlet mixture model (lnGD), infinite Gaussian mixture model (lnGau), KNN and SVM as published in [25] are used to obtain their classification accuracies on Caltech dataset. However, their computational times are not available in [25]. In comparison, it is found that the accuracy for EE and MM is better than lnGau despite using a finite GMM and only slightly poorer than lnGD. However, the predicted model using finite mixture model is poorer than the infinite based mixture models. Also, it is seen that unsupervised classifiers are generally on par with supervised classifiers eg. KNN and SVM for this particular dataset.

Table 3.1: Dataset used in unsupervised classification I

	train	test	attribute	class truth	K
Caltech	1079	1077	256	4	15

Table 3.2: Performance evaluation using predicted model, accuracy and computational time for 20 iterations of EE and MM. (10 average re-runs). The accuracy of lnGD, lnGau, KNN and SVM are cited in [25]. However, their computational times are not given. The best result is in bold. The proposed method is slightly poorer than EE. EE is slightly poorer than the best method.

	lnGD	lnGau	KNN	SVM	EE	MM
Model	3.85	3.80	-	-	7.60	5.40
Accuracy	88.03	84.19	84.32	88.46	87.38	86.47
Time	-	-	-	-	9.98	5.66

Table 3.3: EE confusion matrix (1 arbitrary run)

	Airplane	Car	Face	Motobike
Airplane	288	96	16	0
Car	0	61	0	0
Face	0	0	217	0
Motobike	0	0	11	388

Table 3.4: MM confusion matrix (1 arbitrary run)

	Airplane	Car	Face	Motobike
Airplane	270	128	2	0
Car	0	61	0	0
Face	0	0	217	0
Motobike	0	3	0	396

Figure 3.2: Visualizing the convergence of EE and MM. The true number of clusters is 4. **Top left and center left:** Sorted cluster weights of MM after 20 and 100 iterations respectively. **Top right and center right:** Sorted cluster weights of EE after 20 and 100 iterations. **Bottom row:** The number of iterations vs the computed negative lower bound for 40 iterations, using MM and EE respectively.

3.6.2 Unsupervised Image Classification on Scene Categories

Next, an experiment on the Scene Categories dataset [59] is ran for further comparison of EE and MM. The training setup is identical to [5] for Scene Categories as seen in Table 3.5. There is no dataset split into train and test images here. All the available images from the 4 classes are used for training a Bayesian GMM. The same evaluation criteria as the previous experiment is used. For the attribute, a 200 BoW vector and another 2-by-2 Spatial Pyramid Representation or a 800 SPR vector is sued. The number of iterations is limited to 30 to obtain the results for EE and MM in Table 3.6. An average result of 10 re-runs is used. The state-of-the-art comparison used is the infinite generalized inverted Dirichlet mixture model (GID Var) in [5].

EE is found to slightly outperforms MM in terms of accuracy and predicted model but takes longer to compute. When using the 800 SPR vector as inputs, EE met with numerical errors despite several best attempt. The problem is traced to $E[z_n]$ outputing NaN in Matlab for most of the samples after the 1st iteration despite that values of the other expectations appear to be normal. Such problem was also encountered when using the 200 BoW vector but it only occurred for several samples which were manually removed from training. Further normalizing the dataset also did not seems to help. It is speculated such issue could be due to that EE may not work for some datasets due to insufficient statistics.

Fortunately, MM does not have the same problem as EE. Moreover, it was able to obtain comparable accuracy result to GID Var, and it outperforms both EE and MM using BoW vector with over 10% improvement as well as closer to ground truth class prediction. When increasing the vector by 4 times for MM, the time taken from also increases linearly by 4 times, from 10.22s to 40.89s in Table 3.6. The confusion matrices for EE and MM are shown in Table 3.7, 3.8 and 3.9.

Table 3.5: Dataset used in unsupervised classification II

	train	test	attribute		class truth	K
			BoW	SPR		
Scene Categories	1252	-	200	800	4	15

Table 3.6: Performance evaluation using predicted model, accuracy and computational time for 30 iterations of EE and MM. (10 average re-runs). The best result is in bold. The proposed method using SPR vector (800) is close to the best result. EE is unable to compute the SPR vector. The accuracy of GID VAR is cited from [5] but the computational times was not given.

	GID VAR	EE		MM	
Attribute	81	200	800	200	800
Model	5	9	-	10.33	6.30
Accuracy	80.19	69.32	-	68.59	79.53
Time	-	16.17	-	10.22	40.89

Table 3.7: EE confusion matrix (1 arbitrary run)

	Forest	Highway	InsideCity	TallBuilding
Forest	219	14	0	95
Highway	0	224	3	33
InsideCity	0	41	179	88
TallBuilding	1	84	26	245

Table 3.8: MM confusion matrix (1 arbitrary run)

	Forest	Highway	InsideCity	TallBuilding
Forest	214	108	3	3
Highway	0	235	15	10
InsideCity	0	7	191	110
TallBuilding	2	61	73	220

Table 3.9: MM confusion matrix using SPR vector (1 arbitrary run)

	Forest	Highway	InsideCity	TallBuilding
Forest	262	0	3	63
Highway	0	217	18	25
InsideCity	0	3	261	44
TallBuilding	3	4	95	254

The convergence of EE and MM are visualized on the Scene Categories dataset in Figure 3.3. For MM, at 30 iterations (using SPR vector), there appear

to be the first four dominantly weighted clusters although the model selection is not as distinct as in the Caltech dataset case earlier. At 120 iterations, between cluster 1-7 and cluster 8-15 there is a large offset. If a threshold at 0.06 is allowed, then according to MM, the best model selection is all the clusters from 1-7. For EE (using BoW vector as EE cannot converge when using a SPR vector), when increasing the iterations from 30 to 120, the weights of most clusters are suppressed while the first 3 clusters appear more dominantly weighted than the rest. If a larger threshold is taken say 0.08, there are 3 dominant clusters observed. For a 4 class ground truth, it seems that EE was not able to pick up one of the important cluster. Regardless, for a 30 iterations computed result for EE in Table 3.6, a common threshold (with MM) at 0.06 is used for convenience.

In the bottom row of Figure 3.3, the two plots shows the negative lower bound of each method computed each iteration for 40 iterations. The negative lower bounds of both methods are found to increase until saturation at around 9 iterations for MM and about 17 iterations for EE.

Figure 3.3: Visualizing the convergence of EE and MM. The true number of clusters is 4. **Top left and center left:** Sorted cluster weights of MM after 30 and 120 iterations respectively. **Top right and center right:** Sorted cluster weights of EE after 30 and 120 iterations. **Bottom row:** The number of iterations vs the computed negative lower bound for 40 iterations, using MM and EE respectively.

3.6.3 Unsupervised Feature Learning For Multiview Car Pose and Object Classification

This chapter is concerned is how well MM models the data when compared to traditional learners such as K-means and EM. The performance of MM is evaluated on the Multi-View Car 8-bin-poses classification [84, 67] and the Caltech

10-objects classification [30, 63, 28] datasets.

Table 3.10: Dataset used in unsupervised learning

	train	test	attribute	class truth
Car Pose	400	1040	$100 \leq k \leq 1000$	8
Caltech	400	2644	$100 \leq k \leq 1000$	10

The first dataset contains 2299 image sequences of 20 different cars on a rotating platform (first 10 cars for training and next 10 cars for testing). 400 of 1259 training images and 1040 testing images are used in total. The second dataset has 400 training images and 2644 testing images. Both experiments use object bounding box to extract features. HoG feature is extracted every 8 pixels with a 2x2 partition outputting a 1x32 descriptor with L2 normalization. In both datasets, a random selection of 2000 and 2500 HoG descriptors is used per class respectively totalling 20,000 training set for unsupervised visual dictionary learning.

The baseline comparison are K-means and EM. Both K-means and EM are based on maximum likelihood approach. For EM, only 10 iterations are used as computation is very demanding when the number of Gaussian increases, eg. $k = 200$ takes 3.9 hrs and $k = 700$ takes 46 hrs for Caltech10. In comparison, MM takes 0.3 hrs for $k = 200$. A cluster sizes of $100 \leq k \leq 1000$ is used for all methods. 10 iterations are used to train MM as the learning usually converges after 10 iterations for both datasets.

Each image of a dataset contains its own set of features. After unsupervised learning with GMM or K-means, an image is represented by vector quantization. The most popular method is Bag-of-Word as follows

$$BoW = \sum_{n=1}^N E[z_n] \quad (3.66)$$

Two new constants C and D are introduced [6]. It is shown later on that they allow MM to be easily control with a single value λ_0 by empirically fixing

these two constants. They are related to the hyperparameters of Bayesian GMM as follows

$$\begin{aligned} m_0 &= \frac{C}{\lambda_0} \\ a_0 &= 1 + \frac{\lambda_0}{2} \\ b_0 &= D - \frac{C^2}{2\lambda_0} \end{aligned} \tag{3.67}$$

For $C = 4$ and $D = 6$ and a belief of $\lambda_0 = 2$, the hyperparameters are worked out to be $m_0 = 2$, $a_0 = 2$, $b_0 = 2$. This means that the priors $\tau \sim \text{Gam}(a_0, b_0)$ and $\mu \mid \tau \sim \mathcal{N}(m_0, (\lambda_0 \tau)^{-1})$ are governed by the hyperparameters values, which in turn will affect the estimation of $E[\mu_k]$ and $E[\tau_k]$ in Algorithm 3.2. Unfortunately, there is no optimal step for obtaining the hyperparameter values. This is an **inherent** problem in Bayesian approach and hyperparameter optimization is beyond the scope of this thesis. Instead, the two constants C, D are fixed and the prior belief λ_0 is varied in the experiments to empirically select the best result MM can offer.

Linear SVM is used for classification for all methods. Classification performance is measure in mean Average Precision (mAP) where Average Precision is the area under the precision-recall curve.

$$\begin{aligned} \textit{Precision} &= TP / (TP + FP) \\ \textit{Recall} &= TP / (TP + FN) \\ \textit{mAP} &= \frac{1}{|J|} \sum_{j=1}^J \textit{Average Precision} \end{aligned} \tag{3.68}$$

Where $|J|$ is the number of classes. The mAP results is averaged from ten **re-runs** for each method. The behavior of the methods for different cluster sizes are shown in Figure 3.4 and Figure 3.5 for Multi-view Car and Caltech-10 respectively.

Figure 3.4: Performance comparison for Multi-view Car bin-pose classification

Figure 3.5: Performance comparison for Caltech-10 object classification

In the Multi-view Car experiment in Figure 3.4, MM outperforms both K-means and EM for all cluster size. MM performs less optimal when λ_0 is either too large or zero. MM with smaller value of λ_0 outperforms having $\lambda_0 = 0$. The best mAP using $\lambda_0 = 5$ for MM is 0.7418 at $k = 200$ compared to K-means of 0.66516 at $k = 300$, and EM of 0.67785 at $k = 500$. The MM gain over best baseline is at 6.54% in Table 3.11.

In the Caltech-10 experiment in Figure 3.5, using prior belief $\lambda_0 \leq 1$ works

best for MM. The best mAP for MM using $\lambda_0 = 0.5$ is 0.89608 at $k = 300$ compared to K-means of 0.87769 at $k = 1000$, and EM of 0.86046 at $k = 500$. The MM gain over best baseline is at 1.84% in Table 3.12. In both dataset, it can be seen that even with no prior belief MM outperforms best result of K-means and EM at $k = 300$.

On both dataset, the prior belief is empirically found to be optimal at $\lambda_0 \leq 2$ to outperformed both K-means and EM for all cluster size.

In comparison with recent works (Table 3.13) such as SA [30], SDLM [63] and GEMP [29] on Caltech-10, all three methods SA, SDLM and GEMP are supervised learner for feature learning while MM is an unsupervised learner. The MM best performance is slightly better than SDLM and SA but worse than GEMP. An improvement could be to consider combining supervised learning for MM in future.

Table 3.11: Average Precision (AP) evaluation on the Multi-view Car dataset for 8 bin-pose.

Bin-Pose Class	K-means	EM	MM*	MM
	$k = 300$	$k = 500$	$k = 300$	$k = 200$
Front	0.99437	0.97332	0.99824	0.98534
Front-Left	0.62753	0.45492	0.86211	0.51241
Left	0.69789	0.63366	0.78453	0.89611
Rear-Left	0.45773	0.57537	0.31384	0.64906
Rear	0.87559	0.95798	0.96399	0.83582
Rear-Right	0.58441	0.5265	0.42318	0.65707
Right	0.33067	0.49751	0.37113	0.50651
Front-Right	0.75306	0.80357	0.73772	0.90359
mean AP	0.66516	0.67785	0.68184	0.74324

*(MM using shared precision in [67])

Table 3.12: Average Precision (AP) evaluation on the Caltech-10 dataset.

Object Classes	K-means	EM	MM
	$k = 1000$	$k = 500$	$k = 300$
Airplane	0.99439	0.98236	0.98774
Bonsai	0.4286	0.30479	0.63457
Car	0.98918	0.99026	0.99026
Chandelier	0.89538	0.88957	0.90423
Faces	0.98578	0.9809	0.988
Hawksbill	0.68685	0.79228	0.58625
Ketch	0.84959	0.83017	0.8814
Leopards	0.98226	0.96713	0.98856
Motorbikes	0.96803	0.98379	0.99302
Watch	0.99688	0.88339	1
mean AP	0.87769	0.86046	0.89608

Table 3.13: Comparison of Mean Average Precision on Caltech10

	SA	SDL	GEMP	K-means	EM	MM
k	400	200	200	1000	500	300
mAP	0.875	0.893	0.929	0.878	0.860	0.896

3.7 Conclusion

A new parameter estimation for Bayesian GMM using the MM algorithm is presented in this chapter. The approach is closely related to the variational inference framework. The main difference is that while variational inference rely on computing statistical moments of prior distributions for learning, MM rely on MAP approximation of variational posteriors for computational efficiency purpose. Although both methods offer analytical solution, only the proposed method provide analytical solution in closed-form. This is seen by comparing variational inference to the MM algorithm.

To verify the effectiveness of MM, it is evaluated on two unsupervised image classification datasets. Experimental results shows that MM is more efficient to compute but slightly poorer than variational inference in terms of accuracy and predicted model. Also, both MM and variational inference are comparable to

published state-of-the-art results. Secondly, the maximum likelihood approach (K-means and EM) is compared with a Bayesian approach (using MM) on two unsupervised feature learning datasets. It is found that given empirically determined optimal hyperparameter settings, Bayesian approach significantly outperforms maximum likelihood approach at modeling datasets.

A possible future work would be to extend the MM algorithm to areas where variational inference has received success, particularly in Bayesian nonparametric models, hierarchical models and deep learning. Also, Fisher Vector traditionally rely on maximum likelihood approach for learning GMM. It is possible to extend Fisher Vector using a Bayesian approach to further improve the result for unsupervised feature learning.

4 Learning Bayesian Mixture Model with Dirichlet Process

4.1 Introduction

There are two key observations on fast DPM algorithms: i) Most works do not consider variational inference as mainstream due to the difficulty in implementation and high computation cost. Their works are largely based on MAP estimate of the true posterior for cluster assignment. However, they face intractable posterior and in return they usually degenerate the conditional likelihood (e.g. SVA) to overcome the inefficiency. ii) Most works using variational inference rely on a two-step learning approach (i.e. first compute hyperparameter updates then compute expectation of variational posteriors e.g. Algorithm 4.1). Also, computing the expectation for the cluster assignment variational posterior is more expensive than a MAP estimate. This two-step approach mainly constitutes to the high cost involved.

Inspired by the challenges of fast DPM algorithms, this chapter contributes in the followings: i) The expectations of variational posteriors are directly approximated using MAP estimation. ii) It is shown that the use of lower bounds for computing expectation functions greatly reduce the analytical complexity involved. iii) Variational learning is now completed in a single step in Algorithm 4.2. iv) Cluster assignment is also approximated using MAP estimation on the posterior but within a variational framework. ii) A revisit to the concepts of variational inference is made and it is observed that the analytical solution of expectations obtained by the proposed method are in fact very similar to variational inference. vi) Convergence of the proposed method is also briefly discussed. vii) Lastly, a comparison of the proposed work is made with DP-means and variational inference on several UCI datasets. Empirical results showed that the proposed method obtained the best overall clustering accuracy and model

selection and significantly faster convergence than variational inference.

4.2 Related Work

In DPM-EM [43, 44], Heinzl et al used the EM algorithm for the inference of DPM with linear mixture model. This work mainly differs from DPM-EM in several ways. Firstly, they only considered a maximum likelihood approach to the mixture model whereas in the proposed work, a Bayesian approach is used. Secondly, their proposed solution is based on the EM algorithm, whereas here the MM algorithm is used within a variational inference framework. Lastly, while linear mixture model is more expressive than GMM is used to compute longitudinal data, it comes at an extra computational cost. Thus, linear mixture model may be less desirable when it comes to application that require fast DPM computation. One key similarity is that both expression for the cluster weight variable, V have identical closed-form expression despite both method employing different mixture models. This is possible since the cluster assignment path is disjointed from the mixture model path in the graph model of DPM in Figure 4.1. Thus, inference of V is essentially the same problem for both DPMs despite using different mixture models and approach.

Recently, in [66, 65] MM was proposed to successfully perform learning for the variational inference of GMM [65] and the Sparse Coding based GMM [66]. The main difference between this work and MM-GMM [65] is that the theoretical comparison between MM and variational inference is now extended to stochastic process. Due to having a more complex model, there is greater difficulty computing expectations of variational posteriors here.

4.2.1 Fast DPM Algorithms

A brief introduction to some recently proposed fast Bayesian nonparametrics algorithms is given below. These methods are mainly used as baseline compar-

ison with the proposed method.

X-means: The most fundamental approach to model selection is to use the Bayesian Information Criteria (BIC). It is given as

$$BIC = -2 \ln p(x \mid \hat{\theta}, k) + \ln(N)$$

where the likelihood function is defined as $p(x \mid \hat{\theta}, K)$. $\hat{\theta}_k$ refers to the optimal parameter for each k . e.g. $\hat{\theta}_2 = [1.3, 4.2]$ and $\{x_n\}_{n=1}^N \in \mathbb{R}$. When the k^{th} likelihood function is a good fit to the dataset, $p(x \mid \hat{\theta}, k)$ will be close to 1 and $-2 \ln p(x \mid \hat{\theta}, k) \rightarrow 0$. Hence, BIC will output the largest value for the k^{th} model.

A widely used likelihood function for BIC is the Kmeans objective

$$p(x \mid \hat{\theta}, k) = \sum_{n=1}^N \sum_{k=1}^K -\frac{1}{2} (x_n - \hat{\theta}_k)^2$$

This BIC-Kmeans combination is also known as the X-means algorithm [87]. Although, X-means performs model selection, it is required to guess K ahead and test every k model. A more elegant approach is to turn to Bayesian non-parametrics such as DPM.

DP-means: In clustering, it is required beforehand the number of clusters K to use. Nonparameteric clustering does not have this issue as it can learn K directly from a dataset. DPmeans [56] is a recently proposed nonparameteric clustering. By assuming noninformative prior for mean, a closed-form Gibbs probability of starting a new cluster or being assigned to an existing cluster can be obtained. Next, by assuming small variance assumption $\sigma \rightarrow 0$, the probability of starting a new cluster is now dependent on a single parameter Ω . The DPmeans objective is found to be identical to Kmeans except for the addition of Ω . Initially, DPmeans starts with one cluster and Ω controls when new cluster are introduced.

$$j_n = \begin{cases} \eta_{T+1} = x_n & \sum_t \|x_n - \eta_t\|^2 > \Omega \\ \arg \max_{j_n} \sum_{t=1}^T \ln \mathcal{N}(x_n | \eta_t) \cdot j_{nt} & \text{otherwise} \end{cases}$$

Where j_n, T, η_t, x_n refers to the cluster assignment of sample x_n , total no. of clusters, mean of cluster t and n sample of input x respectively. Ω is chosen either heuristically or manually for each dataset [56, 92]. η_t is re-computed after every iteration. Experimental results on UCI datasets show that DP-means is at least on par or better than Kmeans [56].

MAP-DPM: While SVA greatly reduce the complexity in [13, 56], it also rob away the “rich get richer” property in DPM. MAP-DPM [92] overcame the SVA reliance in DPM [13] (or hard clustering) by exploiting conjugacy in the exponential family. In MAP-DPM [92], the probability of assigning a new sample to a new cluster or to an existing set of clusters is centered around a decision from two Student-T distributions. The link between DPM and Student-T comes from the fact that in a Gibbs sampler, cluster parameters of the Gaussian-Gamma prior are integrated out for efficiency, leading to a Student-T distribution.

Student-T is basically a 3 parameter Gaussian with an additional parameter for degree of freedom. This parameter a_0 (or a_k) allows Student-T to flex between heavy tail and Gaussian like. When a_0 is large enough, Student-T is approximately a Gaussian distribution. The two Student-Ts compete for maximum likelihood score (or minimum negative likelihood). The first Student-T $p(x_{n+1} | z)$ is computed with means and precisions from each existing cluster and the other Student-T $p(x_{n+1} | \theta_0)$ takes the mean and precision of the new sample.

$$p(x_{n+1} | z) = St \left(x_{n+1} | m_k, \frac{a_k c_k}{b_k (c_k + 1)}, 2a_k \right)$$

$$p(x_{n+1} \mid \theta_0) = St \left(x_{n+1} \mid m_0, \frac{a_0 c_0}{b_0(c_0 + 1)}, 2a_0 \right)$$

For each sample, the parameters are updated through a set of rules similar to those in [6, 77, 109]

$$\begin{aligned} m_k &= \frac{c_0 m_0 + N_k \bar{x}_k}{c_0 + N_k} \\ c_k &= c_0 + N_k \\ a_k &= a_0 + \frac{1}{2} N_k \\ b_k &= b_0 + \frac{1}{2} \sum_{j: z_n=k} (x_j - \bar{x}_k)^2 + \frac{c_0 N_k (\bar{x}_k - m_0)^2}{2(c_0 + N_k)} \end{aligned}$$

The Gibbs sampler cluster assignment probabilities are given as

$$q_{i,k} = p(z_{n+1} = k \mid z, \theta_k) = N_k p(x_{n+1} \mid z)$$

$$q_{i,K+1} = p(z_{n+1} = K + 1 \mid \omega, \theta_0) = \omega p(x_{n+1} \mid \theta_0)$$

A parameter ω affects the cluster assignment of starting a new cluster. Like Ω in DP-means, ω can also be hand selected for convenience. Finally, MAP-DPM compute MAP estimate on the probabilities

$$z = \arg \max_{k \in \{1, \dots, K, K+1\}} \ln q_{i,k}$$

Unlike DP-means and MAP-DPM the proposed approach does not have to refer to ground truth on class size to specifically hand select a certain parameter value for each dataset. MAP-DPM is very similar to another DPM algorithm which has also appear in [41] for background subtraction.

4.3 Background

4.3.1 Dirichlet Process Mixture

Figure 4.1: Graph model of DPM

The mixture model in DPM (Figure 4.1) is assumed Gaussian distributed. DPM models a set of N observed variable denoted as $x = \{x_n\}_{n=1}^N \in \mathbb{R}^D$ with a set of hidden variables, $\theta = \{\mu, \tau, z, v\}$. The total dimension of each observed instance is denoted D . Diagonal covariance $\Sigma_k = \sigma_k^2 I$ is assumed and precision $\tau_k = \frac{1}{\sigma_k^2}$ is denoted $\tau = \{\tau_k\}_{k=1}^K \in \mathbb{R}^D$. The mean is denoted $\mu = \{\mu_k\}_{k=1}^K \in \mathbb{R}^D$. The cluster assignment is denoted $z = \{z_n\}_{n=1}^N$ where z_n is a 1-of- K binary vector, subjected to $\sum_{k=1}^K z_{nk} = 1$ and $z_{nk} \in \{0, 1\}$. The truncated stick-breaking [97] process to DPM with truncation level is denoted K . Starting with a unit length stick, the proportional length of each broken off piece (w.r.t remainder of stick) is a random variable v drawn from a Beta distribution, denoted $v = \{v_k\}_{k=1}^K \in \mathbb{R}$.

All broken off pieces add up to a unit length of the full stick. This value can be seen as the cluster weight similar to GMM and is given as

$$\pi_k = v_k \prod_{l=1}^{k-1} (1 - v_l) \quad (4.1)$$

The joint distribution between the observed and hidden variables of DPM, can be factorized as $p(x, z, v, \mu, \tau) = p(x | z, \mu, \tau) p(\mu | \tau) p(\tau) p(z | v) p(v)$. In a Bayesian

approach, each variable is modeled with a distribution as follows [6, 9, 16]

$$\begin{aligned}
p(x \mid z, \mu, \tau) &= \prod_{n=1}^N \prod_{k=1}^K \mathcal{N}(x_n \mid \mu_k, \tau_k^{-1})^{z_{nk}} \\
p(\mu, \tau) &= \prod_{k=1}^K \mathcal{N}(\mu_k \mid m_0, (\lambda_0 \tau_k)^{-1}) \text{Gam}(\tau_k \mid a_0, b_0) \\
p(\nu) &= \prod_{k=1}^K \text{Beta}(\nu_k \mid 1, \alpha_0) \\
p(z \mid v) &= \prod_{n=1}^N \prod_{k=1}^K (w_k)^{z_{nk}}
\end{aligned} \tag{4.2}$$

where $\mathcal{N}, \text{Gam}, \text{Beta}$ refers to the Gaussian, Gamma, and Beta distribution respectively. The hyperparameters are denoted $a_0, b_0, m_0, \lambda_0, \alpha_0$.

4.3.2 Variational Posteriors

To obtain the marginal distribution of DPM, the joint hidden variables are integrated out as follows [16]

$$\begin{aligned}
p(x) &= \sum_z \int \int p(x \mid z, \mu, \tau) p(\mu \mid \tau) p(\tau) \\
&\quad p(z \mid v) p(v) d\mu d\tau dv
\end{aligned} \tag{4.3}$$

The intractable form which arise from the expression, $p(\theta \mid x) = \frac{p(x|\theta)p(\theta)}{p(x)}$ and is known as the **true posterior**.

When seeking expectation of the true posterior, it may be too complex to work with directly. Instead, variational inference approximates the true posterior by using **variational posterior** which is analytically tractable [6, 9]. The expression is obtained using $\ln q(\theta_j) = E_{i \neq j} [\ln p(X, \theta_i)] + \text{const.}$ and mean field factorization $q(z, v, \mu, \tau) = \prod_{n=1}^N \prod_{k=1}^K q(z_{nk})q(v_k)q(\mu_k, \tau_k)$. The “unnormalized” variational posteriors of DPM is defined as follows (for brevity, the subscripts

are omitted)

$$\begin{aligned}
\ln q(\mu) &= E_{\tau,z} [\ln p(x \mid z, \mu, \tau) p(\mu \mid \tau)] + \text{const.} \\
\ln q(\tau) &= E_{\mu,z} [\ln p(x \mid z, \mu, \tau) p(\mu \mid \tau) p(\tau)] + \text{const.} \\
\ln q(v) &= E_z [\ln p(z \mid v) p(v)] + \text{const.} \\
\ln q(z) &= E_{\mu,\tau,v} [\ln p(x \mid z, \mu, \tau) p(z \mid v)] + \text{const.}
\end{aligned} \tag{4.4}$$

4.3.3 Problem Formulation

Variational posteriors are optimized by iteratively computing their expectations until convergence [26, 71, 5, 16]. The expectations of variational posteriors for DPM are introduced below

$$\begin{aligned}
E[\mu] &= \int \mu q(\mu) d\mu \\
E[\tau] &= \int \tau q(\tau) d\tau \\
E[\ln \tau] &= \int \ln \tau q(\ln \tau) d \ln \tau \\
E[\ln v] &= \int \ln v q(\ln v) d \ln v \\
E[\ln(1-v)] &= \int \ln(1-v) q(\ln(1-v)) d \ln(1-v) \\
E[z] &\propto \exp \{ \ln q(z) \}
\end{aligned} \tag{4.5}$$

These expectations are analytically intractable except for $E[z]$, which refers to the cluster assignment variational posterior expectation. The first five expectations are revealed as elements used to compute $E[z]$ in Algorithm 4.1. Variational inference do not explicitly solve for eqn (4.5) as seen in the next section. However in this case, it is possible to directly solve for eqn (4.5) using an optimization route. Note that in eqn (4.5), the “normalized” variational posteriors are defined as $q(\theta) = \frac{\exp(\ln q(\theta))}{C}$, where $C = \int \exp(\ln q(\theta)) d\theta$ is the normalizing constant.

4.3.4 Learning Algorithm of Variational Inference

Expectation: When conjugate prior is assumed, the variational posterior distribution will follow the same distribution as the prior. In the DPM case, the variational posterior distribution of the continuous hidden variables in eqn (4.4) are the Gaussian-Gaussian, Gaussian-Gamma and Multinomial-Beta. Their equivalent conjugate distributions are the Gaussian, Gamma and Beta distributions. Due to conjugacy, expectations of the variational posteriors can be defined by the statistical moments of conjugate distribution. For example in Table 4.1, the log-expectation of the variational posterior $q(\tau_k) \sim \text{Gam}(a_k, b_k)$ is $E[\ln \tau_k] = \psi(a_k) - \ln(b_k)$. ψ refers to the digamma function.

Table 4.1: In most variational inference algorithms, expectations of variational posterior distribution are typically obtained by exploiting statistical moments of well-defined conjugate prior.

variational posterior distribution	expectation	log-expectation
$q(\mu_k) \sim \mathcal{N}(m_k)$	$E[\mu_k] = m_k$	-
$q(\tau_k) \sim \text{Gam}(a_k, b_k)$	$E[\tau_k] = a_k/b_k$	$E[\ln \tau_k] = \psi(a_k) - \ln(b_k)$
$q(v_k) \sim \text{Beta}(\gamma_{k,1}, \gamma_{k,2})$	-	$E[\ln v_k] = \psi(\gamma_{k,1}) - \psi(\gamma_{k,1} + \gamma_{k,2})$ $E[\ln(1 - v_k)] = \psi(\gamma_{k,2}) - \psi(\gamma_{k,1} + \gamma_{k,2})$

Hyperparameter: Before computing the expectations of the variational posteriors, their hyperparameters, $a_k, b_k, \lambda_k, m_k, \gamma_{k,1}, \gamma_{k,2}$ must be updated. The reader can refer to [6, 9, 16, 71, 76] for a detailed discussion on the hyperparameters expressions obtained in Table 4.2.

Table 4.2: Lookup table for updating variational posterior hyperparameters

likelihood	conjugate prior	variational posterior hyperparameters
Gaussian	Gaussian	$m_k = \frac{1}{\lambda_k} \left(\lambda_0 m_0 + \sum_{n=1}^N E[z_{nk}] x_n \right)$ $\lambda_k = \lambda_0 + \sum_{n=1}^N E[z_{nk}]$
Gaussian	Gamma	$a_k = a_0 + \frac{1}{2} \sum_{n=1}^N E[z_{nk}]$ $b_k = b_0 + \frac{1}{2} \left(\sum_{n=1}^N (x_n - m_k)^2 E[z_{nk}] + \lambda_0 (m_k - m_0)^2 \right)$
Multinomial	Beta	$\gamma_{k,1} = 1 + \sum_{n=1}^N E[z_{nk}]$ $\gamma_{k,2} = \alpha_0 + \sum_{j=k+1}^K \sum_{n=1}^N E[z_{nj}]$

In this work, the variational inference of DPM (VB-DPM) has been thoroughly studied from several sources [9, 6, 16] and is summarized in Algorithm 4.1. Note that $\mathbb{Z}$ is a normalization term in $E[z_{nk}]$.

Algorithm 4.1 VB-DPM [9, 6, 16]

- 1) Input: x
- 2) Output: $E[z_{nk}]$
- 3) Initialization: $E[z_{nk}], m_0, \alpha_0, \lambda_0, a_0, b_0$
- 4) Repeat until max iterations or convergence,
 - i) Update the **hyperparameters**:

$$\begin{aligned}
 a_k &= a_0 + \frac{1}{2} \sum_{n=1}^N E[z_{nk}] \\
 b_k &= b_0 + \frac{1}{2} \left(\sum_{n=1}^N (x_n - m_k)^2 E[z_{nk}] + \lambda_0 (m_k - m_0)^2 \right) \\
 \lambda_k &= \lambda_0 + \sum_{n=1}^N E[z_{nk}] \\
 m_k &= \frac{1}{\lambda_k} \left(\lambda_0 m_0 + \sum_{n=1}^N E[z_{nk}] x_n \right) \\
 \gamma_{k,1} &= 1 + \sum_{n=1}^N E[z_{nk}] \\
 \gamma_{k,2} &= \alpha_0 + \sum_{j=k+1}^K \sum_{n=1}^N E[z_{nj}]
 \end{aligned}$$

- ii) Update the **expectations** (statistical moments):

$$\begin{aligned}
 E[\tau_k] &= a_k / b_k \\
 E[\ln \tau_k] &= \psi(a_k) - \ln(b_k) \\
 E[\mu_k] &= m_k \\
 E[\ln v_k] &= \psi(\gamma_{k,1}) - \psi(\gamma_{k,1} + \gamma_{k,2}) \\
 E[\ln(1 - v_k)] &= \psi(\gamma_{k,2}) - \psi(\gamma_{k,1} + \gamma_{k,2}) \\
 E[z_{nk}] &= \frac{1}{\mathbb{Z}} \exp \left\{ E[\ln v_k] + \sum_{l=1}^{k-1} E[\ln(1 - v_l)] \right. \\
 &\quad \left. + \frac{1}{2} E[\ln \tau_k] - \frac{1}{2} E[\tau_k] (x_n - E[\mu_k])^2 \right\}
 \end{aligned}$$

End

4.4 Proposed Method, MM-DPM

4.4.1 Expectation of Variational Posteriors

In comparison to variational inference: i) A revisit to eqn (4.5) by approximating expectations $E[\theta]$ using MAP estimation of the variational posteriors (multinomial-beta, normal-normal, normal-gamma distributions) is made. ii)

New closed-form expressions are obtained for the expectation of variational posteriors. iii) No hyperparameter updates are required for computing expectations. iv) Some derived expressions for expectations are in fact identical to the ones obtained by variational inference using statistical moments.

MAP estimation to directly approximate the integrals in eqn (4.5) are as follows [70]

$$\begin{aligned}
E[\mu] &= \int \mu q(\mu) d\mu \approx \arg \max_{\mu} q(\mu) \\
E[\tau] &= \int \tau q(\tau) d\tau \approx \arg \max_{\tau} q(\tau) \\
E[\ln \tau] &= \int \ln \tau q(\ln \tau) d \ln \tau \approx \arg \max_{\ln \tau} q(\ln \tau) \\
E[\ln v] &= \int \ln v q(\ln v) d \ln v \approx \arg \max_{\ln v} q(\ln v) \\
E[\ln(1-v)] &= \int \ln(1-v) q(\ln(1-v)) d \ln(1-v) \\
&\approx \arg \max_{\ln(1-v)} q(\ln(1-v))
\end{aligned} \tag{4.6}$$

There are two difficulties here: i) Based on eqn (4.6), there are log-expectations to solve. For MAP estimation, it means that there are undefined variational posteriors $q(\ln \tau)$, $q(\ln v)$ and $q(\ln(1-v))$ to solve. ii) This increases the analytical complexity over variational inference. Thus, the direct use of this approach may seem inelegant. An effective way to **reduce** the analytical complexity in eqn (4.6) is to take the lower bound of the following $E[\ln \tau] \geq \ln E[\tau]$, $E[\ln v] \geq \ln E[v]$ and $E[\ln(1-v)] \geq \ln(1-E[v])$. This means that only the integral of the variational posteriors that originally appeared in eqn (4.4) are required to be approximated

$$\begin{aligned}
E[\mu] &= \int \mu q(\mu) d\mu \approx \arg \max_{\mu} E_{\tau,z} [\ln p(x | z, \mu, \tau) p(\mu | \tau)] \\
E[\tau] &= \int \tau q(\tau) d\tau \approx \arg \max_{\tau} E_{\mu,z} [\ln p(x | z, \mu, \tau) p(\mu | \tau) p(\tau)] \\
E[v] &= \int v q(v) dv \approx \arg \max_v E_z [\ln p(z | v) p(v)] \\
E[z] &= \int z q(z) dz \approx \arg \max_z E_{\mu,\tau,v} [\ln p(x | z, \mu, \tau) p(z | v)]
\end{aligned} \tag{4.7}$$

In the above expressions, it is interesting to note that since only the optimal value of θ that satisfy the objective function $\arg \max_{\theta} \frac{\exp(\ln q(\theta))}{C}$ is of interest, further computational expenses can be reduced by i) skipping the normalization step required and ii) directly using the un-normalized form, $\arg \max_{\theta} \ln q(\theta)$. The reason for the latter is that it is easier to work with $\ln q(\theta)$ and since the logarithm function is monotonic, it will not affect the optimal value of θ estimated. Expressions in eqn (4.7) are further simplified using the lower bound of the following, $\arg \max_{\theta_j} E_{i \neq j} [f(\theta_i)] \geq \arg \max_{\theta_j} f(E_{i \neq j} [\theta_i])$.

Fortunately, closed-form solutions to the MAP estimation in eqn (4.7) can be directly derived by hand (refer to section 4.4.2). The closed-form approximation to the expectations of the variational posteriors are summarized below:

$$E[\mu_k] \approx \frac{\sum_{n=1}^N E[z_{nk}]x_n + \lambda_0 m_0}{\sum_{n=1}^N E[z_{nk}] + \lambda_0} \quad (4.8)$$

$$E[\tau_k] \approx \frac{\frac{1}{2} \sum_{n=1}^N E[z_{nk}] + a_0 - 1}{b_0 + \frac{1}{2} (\sum_{n=1}^N (x_n - E[\mu_k])^2 E[z_{nk}] + \lambda_0 (E[\mu_k] - m_0)^2)} \quad (4.9)$$

$$E[v_k] \approx \frac{\sum_{n=1}^N E[z_{nk}]}{\sum_{n=1}^N \sum_{j=k+1}^K E[z_{nj}] + \alpha_0 - 1} \quad (4.10)$$

$$E[z_{nk}] \approx \arg \max_{z_{nk}} \left\{ \ln E[v_k] + \sum_{l=1}^{k-1} \ln(1 - E[v_l]) + \frac{1}{2} \ln E[\tau_k] - \frac{1}{2} E[\tau_k] (x_n - E[\mu_k])^2 \right\} z_{nk} \quad (4.11)$$

4.4.2 Proof to Analytical Solution of Expectations

The proof to eqn (4.8), (4.9), (4.10), (4.11) are given in this section. Since diagonal covariance is assumed, the dimension d of each hidden variable of the mixture component can be treated independently. In the proof below, the reader can refer to [81, 6] to find out how to obtain expressions for the conjugate priors of the exponential family.

Derivation of $E[\mu]$:

The variational posterior, $\ln q(\mu)$ is found by keeping terms related to μ in

eqn (4.4), while the rest are absorbed into constant

$$\ln q(\mu) = E_{z,\tau} [\ln p(x | z, \mu, \tau) + \ln p(\mu | \tau)] + \text{const.} \quad (4.12)$$

The likelihood term is Gaussian distributed

$$\ln p(x | z, \mu, \tau) \propto \sum_{d=1}^D \sum_{k=1}^K \sum_{n=1}^N \left(-\frac{\tau_{dk}(x_{nd} - \mu_{dk})^2}{2} \right) z_{nk} \quad (4.13)$$

and the conjugate prior is also Gaussian distributed

$$\ln p(\mu | \tau) \propto \sum_{d=1}^D \sum_{k=1}^K -\frac{(\mu_{dk} - m_0)^2}{2(\lambda_0 \tau_{dk})^{-1}} \quad (4.14)$$

The MAP estimate of $\ln q(\mu)$ is

$$\begin{aligned} E[\mu] &\approx \arg \max_{\mu} E_{z,\tau} [\ln p(x | z, \mu, \tau) + \ln p(\mu | \tau)] \\ &= \arg \max_{\mu} \sum_{d=1}^D \sum_{k=1}^K \sum_{n=1}^N \left(-\frac{E[\tau_{dk}](x_{nd} - \mu_{dk})^2}{2} \right) E[z_{nk}] \\ &\quad - \frac{(\mu_{dk} - m_0)^2}{2(\lambda_0 E[\tau_{dk}])^{-1}} \end{aligned} \quad (4.15)$$

Taking the derivative of the log posterior w.r.t μ , a closed-form MAP estimated expression for $\ln q(\mu)$ is derived for an independent assumption of d and k

$$E[\mu_{dk}] \approx \frac{\sum_{n=1}^N E[z_{nk}] \cdot x_{nd} + \lambda_0 m_0}{\sum_{n=1}^N E[z_{nk}] + \lambda_0} \quad (4.16)$$

The parameters λ_0 and m_0 refer to the prior belief and hyperparameter respectively. When $\lambda_0 = 0$, $E[\mu_{dk}]$ is identical to the maximum likelihood obtained point estimate for μ .

Derivation of $E[\tau]$:

Reading off the factorized terms related to precision τ in eqn (4.4), the log

posterior is

$$\ln q(\tau) = E_{\mu,z} [\ln p(x | z, \mu, \tau) + \ln p(\mu | \tau) + \ln p(\tau)] + \text{const.} \quad (4.17)$$

For a Gaussian observation, the likelihood is

$$\ln p(x | z, \mu, \tau) \propto \sum_{d=1}^D \sum_{k=1}^K \sum_{n=1}^N \left(\frac{1}{2} \ln(\tau_{dk}) - \frac{\tau_{dk}(x_{nd} - \mu_{dk})^2}{2} \right) z_{nk} \quad (4.18)$$

The conjugate prior $p(\mu | \tau)$ is already previously defined, while the conjugate prior for precision τ is assumed Gamma distributed

$$\ln p(\tau) \propto \sum_{d=1}^D \sum_{k=1}^K (a_0 - 1) \ln(\tau_{dk}) - b_0 \tau_{dk} \quad (4.19)$$

The full expression for the MAP inference of $\ln q(\tau)$ is

$$\begin{aligned} E[\tau] &\approx \arg \max_{\tau} E_{\mu,z} [\ln p(x | z, \mu, \tau) + \ln p(\mu | \tau) + \ln p(\tau)] \\ &= \arg \max_{\tau} \sum_{d=1}^D \sum_{k=1}^K \sum_{n=1}^N \left(\frac{1}{2} \ln(\tau_{dk}) - \frac{\tau_{dk}(x_{nd} - E[\mu_{dk}])^2}{2} \right) E[z_{nk}] \\ &\quad - \frac{(E[\mu_{dk}] - m_0)^2}{2(\lambda_0 \tau_{dk})^{-1}} + (a_0 - 1) \ln(\tau_{dk}) - b_0 \tau_{dk} \end{aligned} \quad (4.20)$$

After optimization and taking an independent assumption of d and k for τ , a closed-form expression can be obtained as follows

$$E[\tau_{dk}] \approx \frac{\frac{1}{2} \sum_{n=1}^N E[z_{nk}] + a_0 - 1}{b_0 + \frac{1}{2} \left(\sum_{n=1}^N (x_{nd} - E[\mu_{dk}])^2 E[z_{nk}] + \lambda_0 (E[\mu_{dk}] - m_0)^2 \right)} \quad (4.21)$$

Derivation of $E[v]$

The log posterior of cluster weight, v can be found from eqn (4.4)

$$\ln q(v) = E_z [\ln p(z | v) + \ln p(v)] + \text{const.} \quad (4.22)$$

The likelihood in terms of a stick-breaking representation of mixture component

is

$$\ln p(z \mid v) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \left[\ln(v_k) + \sum_{l=1}^{k-1} \ln(1 - v_l) \right] \quad (4.23)$$

An equivalent form to the above due to Blei and Jordan [9] is

$$\ln p(z \mid v) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \ln(v_k) + \sum_{j=k+1}^K z_{nj} \ln(1 - v_k) \quad (4.24)$$

Given the Beta distributed prior is

$$\ln p(v; 1, \alpha_0) = \sum_{k=1}^K (\alpha_0 - 1) \ln(1 - v_k) \quad (4.25)$$

Thus, the MAP estimation for $\ln q(v)$ can be written as follows

$$\begin{aligned} E[v] &\approx \arg \max_v E_z [\ln p(z \mid v) + \ln p(v)] \\ &= \arg \max_v \sum_{k=1}^K \sum_{n=1}^N E[z_{nk}] \ln(v_k) \\ &\quad + \sum_{j=k+1}^K E[z_{nj}] \ln(1 - v_k) + \sum_{k=1}^K (\alpha_0 - 1) \ln(1 - v_k) \end{aligned} \quad (4.26)$$

After optimization, a closed-form expression for $E[v]$ for each independent assumption of k is obtained below

$$E[v_k] \approx \frac{\sum_{n=1}^N E[z_{nk}]}{\sum_{n=1}^N \sum_{j=k+1}^K E[z_{nj}] + \alpha_0 - 1} \quad (4.27)$$

Note that α_0 is a hyperparameter of $\ln q(v)$

Derivation of $E[z]$

Lastly, the cluster assignment is inferred. The log posterior is defined as follows

$$\ln q(z) = E_{\eta, \tau, v} [\ln p(x \mid z, \mu, \tau) + \ln p(z \mid v)] + \text{const.} \quad (4.28)$$

The MAP estimate of $\ln q(z)$ is found by

$$E[z] = \arg \max_z E_{\mu, \tau, v} [\ln p(x | z, \mu, \tau) + \ln p(z | v)] \quad (4.29)$$

For each independent assumption of μ and k , the cluster assignment is given as

$$E[z_{nk}] = \arg \max_{z_{nk}} \left[\ln E[v_k] + \sum_{l=1}^{k-1} \ln(1 - E[v_l]) + \sum_{d=1}^D \left(\frac{\ln E[\tau_{dk}]}{2} - \frac{E[\tau_{dk}]}{2} (x_{nd} - E[\mu_{dk}])^2 \right) \right] z_{nk} \quad (4.30)$$

The notation for $E[z_{nk}]$ in eqn (4.30) should be interpreted as finding the k^{th} cluster and its associated parameters e.g. $E[\tau_{dk}]$, $E[v_k]$, $E[\mu_{dk}]$ that gives the largest value for the RHS of eqn (4.30) and setting $E[z_{nk}] = 1$ for this particular k (the remaining $1, \dots, K$ for $E[z_{nk}] = 0$).

4.4.3 Convergence of Proposed Method

The proposed method learns DPM by updating expectations of variational posteriors for each iteration similarly to variational inference. The update is related to an objective function

$$\arg \max_z E_{\Theta} [\ln p(x, \Theta, z)] \leftrightarrow \arg \max_{\Theta} E_z [\ln p(x, \Theta, z)] \quad (4.31)$$

known as the maximization-maximization (MM) algorithm [78, 102, 57, 65]. It can be categorized under the EM algorithm and the objective function seeks to converge to the global maximum of the data.

The argument for convergence of MM is briefly as follows:

- i) By restricting the mixture component to exponential family, the RHS of eqn (4.31) is a convex function, hence a unique maximum exist.
- ii) In the LHS of eqn (4.31), for independent data points $x_1, \dots, x_N$

with corresponding cluster assignment $z_1, \dots, z_N$, given sufficient statistics each data point should only fall under one cluster, meaning that for LHS of eqn (4.31) there is indeed a unique maximum [6].

It is further discussed in [78, 102, 6] that the above function monotonic increases for each iteration and will lead to convergence. The MM objective function can also be expressed as a lower bound and the convergence of Algorithm 4.2 can be monitored by checking that the lower bound does not decrease for each iteration [71, 72, 73, 26, 25, 115, 45]. The expressions used to compute the lower bound of the proposed method is given below.

$$\begin{aligned}
\mathcal{L} &\geq \sum_z \int q(z, \Theta) \ln \left[\frac{p(x, z, \Theta)}{q(z, \Theta)} \right] d\Theta \\
&\geq \sum_z \int \int \int q(z, \mu, \tau, v) \ln \left[\frac{p(x, z, \mu, \tau, v)}{q(z, \mu, \tau, v)} \right] d\mu d\tau dv \\
&= E [\ln p(x, z, \mu, \tau, v)] - E [\ln q(z, \mu, \tau, v)] \\
&= E [\ln p(x | z, \mu, \tau)] + E [\ln p(z | v)] + E [\ln p(v)] \\
&\quad + E [\ln p(\mu, \tau)] - E [\ln q(z)] - E [\ln q(v)] - E [\ln q(\mu, \tau)]
\end{aligned} \tag{4.32}$$

In the expression above, each expectation function are detailed below as follows

$$\begin{aligned}
E [\ln p(x | z, \mu, \tau)] &= E \left[\sum_{k=1}^K \sum_{n=1}^N \ln \mathcal{N}(x_n | \mu_k, \tau_k^{-1})^{z_{nk}} \right] \\
&= \sum_{k=1}^K \sum_{n=1}^N \left\{ \frac{1}{2} E [\ln \tau_k] - \frac{E[\tau_k]}{2} (x_n - E[\mu_k])^2 \right\} E [z_{nk}]
\end{aligned} \tag{4.33}$$

$$\begin{aligned}
E [\ln p(z | v)] &= E \left[\sum_{n=1}^N \sum_{k=1}^K \ln v_k E [z_{nk}] + \sum_{l=1}^{k-1} \ln(1 - v_l) E [z_{nk}] \right] \\
&= \sum_{n=1}^N \sum_{k=1}^K \ln E [v_k] E [z_{nk}] + \sum_{l=1}^{k-1} \ln(1 - E [v_l]) E [z_{nk}]
\end{aligned} \tag{4.34}$$

$$E [\ln p(v)] = E \left[\sum_{k=1}^K (\alpha_0 - 1) \ln(1 - v_k) \right] = (\alpha_0 - 1) \sum_{k=1}^K \ln(1 - E[v_k]) \quad (4.35)$$

$$\begin{aligned} E [\ln p(\mu, \tau)] &= E \left[\sum_{k=1}^K \ln \mathcal{N}(\mu_k \mid m_0, (\lambda_0 \tau_k)^{-1}) \text{Gam}(\tau_k \mid a_0, b_0) \right] \\ &= \sum_{k=1}^K \left(-\frac{(E[\mu_k] - m_0)^2}{2(\lambda_0 E[\tau_k])^{-1}} + (a_0 - 1)E[\ln \tau_k] - b_0 E[\tau_k] \right) \end{aligned} \quad (4.36)$$

$$E [\ln q(z)] = \sum_{n=1}^N \sum_{k=1}^K \left\{ \ln E[\pi_k] + \frac{1}{2} \ln E[\tau_k] - \frac{E[\tau_k]}{2} (x_n - E[\mu_k])^2 \right\} E[z_{nk}] \quad (4.37)$$

$$\begin{aligned} E [\ln q(v)] &= E \left[\sum_{n=1}^N \sum_{k=1}^K \ln v_k E[z_{nk}] \right. \\ &\quad \left. + \sum_{l=1}^{k-1} \ln(1 - v_l) E[z_{nk}] + \sum_{k=1}^K (\alpha_0 - 1) \ln(1 - v_k) \right] \\ &= \sum_{n=1}^N \sum_{k=1}^K \left\{ \ln E[v_k] + \sum_{l=1}^{k-1} \ln(1 - E[v_l]) \right\} E[z_{nk}] \\ &\quad + (\alpha_0 - 1) \sum_{k=1}^K \ln(1 - E[v_k]) \end{aligned} \quad (4.38)$$

$$\begin{aligned} E [\ln q(\mu, \tau)] &= E \left[\sum_{k=1}^K \left(\sum_{n=1}^N \left\{ \frac{\ln \tau_k}{2} - \frac{\tau_k}{2} (x_n - E[\mu_k])^2 \right\} E[z_{nk}] \right. \right. \\ &\quad \left. \left. - \frac{(E[\mu_k] - m_0)^2}{2(\lambda_0 \tau_k)^{-1}} + (a_0 - 1) \ln \tau_k - b_0 \tau_k \right) \right] \\ &= \sum_{k=1}^K \left(\sum_{n=1}^N \left\{ \frac{E[\ln \tau_k]}{2} - \frac{E[\tau_k]}{2} (x_n - E[\mu_k])^2 \right\} E[z_{nk}] \right. \\ &\quad \left. - \frac{(E[\mu_k] - m_0)^2}{2(\lambda_0 E[\tau_k])^{-1}} + (a_0 - 1)E[\ln \tau_k] - b_0 [E\tau_k] \right) \end{aligned} \quad (4.39)$$

4.4.4 Theoretical Comparision to Variational Inference

Some expectations obtained by the proposed method are in fact very similar to variational inference. Below, some theoretical relationship between variational inference in Algorithm 4.1 and the proposed method in Algorithm 4.2 for the continuous hidden variables of DPM are shown.

Comparison of $E[\mu_k]$

By substituting λ_k to m_k in Algorithm 4.1, it is straightforward to show that $E[\mu_k]$ is identical for both algorithms as follows

$$\frac{1}{\lambda_k} \left(\lambda_0 m_0 + \sum_{n=1}^N E[z_{nk}] x_n \right) = \frac{\sum_{n=1}^N E[z_{nk}] x_n + \lambda_0 m_0}{\sum_{n=1}^N E[z_{nk}] + \lambda_0} \quad (4.40)$$

Comparison of $E[\tau_k]$

In eqn (4.9), if $a_0 > 1$ is set in the RHS of eqn (4.41) below, then it can be seen that $E[\tau_k]$ is identical for both algorithms

$$\frac{a_k}{b_k} = \frac{\frac{1}{2} \sum_{n=1}^N E[z_{nk}] + a_0 - 1}{b_0 + \frac{1}{2} \left(\sum_{n=1}^N (x_n - m_k)^2 E[z_{nk}] + \lambda_0 (m_k - m_0)^2 \right)} \quad (4.41)$$

Comparison of $E[v_k]$

In [9], the authors use the conjugacy between a multinomial likelihood and a Beta prior distribution to obtain hyperparameter updates. Thus, the variational posterior should have a similar form to a Beta prior. Given that the expectation of a Dirichlet prior is

$$E[v_k] = \frac{\gamma_{k,1}}{\gamma_{k,1} + \gamma_{k,2}} \quad (4.42)$$

where $v_k \sim \text{Beta}(\gamma_{k,1}, \gamma_{k,2})$. For $z \mid v \sim \text{Mult}(v)$, the multinomial-Beta hyperparameter update is

$$\begin{aligned} \gamma_{k,1} &= 1 + \sum_{n=1}^N E[z_{nk}] \\ \gamma_{k,2} &= \alpha_0 + \sum_{n=1}^N \sum_{j=k+1}^K E[z_{nj}] \end{aligned} \quad (4.43)$$

Note that in [9] and in eqn (4.25), the Beta prior is $v_k \sim \text{Beta}(1, \gamma_{k,2})$. Since $E[\ln v_k]$ cannot be directly compared for both methods, $E[v_k]$ shall be compared instead. The statistical moment of $E[v_k]$ is approximate to $E[v_k]$ in Algorithm 4.2 when using the following expression

$$\frac{1}{1+\gamma_{k,2}} \approx \frac{\sum_{n=1}^N E[z_{nk}]}{\sum_{n=1}^N \sum_{j=k+1}^K E[z_{nj}] + \alpha_0 - 1} \quad (4.44)$$

$$\frac{1}{1+\alpha_0+\sum_{n=1}^N \frac{1}{\sum_{j=k+1}^K E[z_{nj}]}} \approx N_k \left(\frac{1}{\alpha_0 - 1 + \sum_{n=1}^N \frac{1}{\sum_{j=k+1}^K E[z_{nj}]}} \right)$$

On both sides of eqn (4.44), if α_0 is assumed significantly large, then the RHS of eqn (4.44) can be seen as a scaled factor of the LHS of eqn (4.44).

4.4.5 Proposed Algorithm, MM-DPM

The proposed algorithm which is called MM-DPM as shown in Algorithm 4.2.

The learning objective of MM-DPM is to iteratively update $E[\mu_k], E[\tau_k], E[v_k], E[z_{nk}]$ until convergence.

Initialization: The initialization of $E[z_{nk}], m_0, \alpha_0, \lambda_0, a_0, b_0, K$ are discussed here. The values of $E[z_{nk}]$ are randomly set using a multinomial random generator. The truncation level, K is fixed to a much larger value than ground truth such that it is independent of the ground truth (eg. $K = 50$ when the ground truth is < 10 , or $K = 100$ when the ground truth is > 30). Due to lack of domain knowledge, it is preferred to avoid making decision on the hyperparameters values. They are usually set to $\alpha_0 = N, m_0 = 1, \lambda_0 = 1, a_0 = 1.5, b_0 = 0.5$ for most experiments.

Cluster Pruning: In order to speed up the algorithm, at the end of each iteration a run on the threshold check, $E[v_k] < THR$ for $k = 1, \dots, K$ is made. When $E[v_k]$ does not meet the threshold, the k^{th} cluster and its associated expectations, $E[\mu_k], E[\tau_k], E[v_k]$ are discarded. Thus, there is 1 lesser cluster to compute for each case of $E[v_k] < THR$ being true at the next iteration. The value of THR to use can take cue by plotting the graph of $E[v_k]$ for all K after a maximum number of iterations has passed. In the early iterations, there are dominant clusters and some insignificant ones. However, a typical setting is to use $THR > 0$ and let the algorithm run till it converge. The lower bound is computed using eqn (4.32) to check the saturation level. Another way

is to monitor when the cluster size no longer changes after sufficient iterations is made.

Cluster Ordering: Another change that is introduced is to re-arrange $E[v_k]$ according to descending value. After re-ordering $E[v_k]$, it is necessary to also align $E[\mu_k], E[\tau_k]$ before computing $E[z_{nk}]$. This is due to the definition of the stick-breaking process, $\sum_{l=1}^{k-1} \ln(1 - E[v_l])$ appearing in eqn (4.30). Ideally, more emphasis is given to clusters with larger values of $E[v_k]$ which due to more frequent appearance rather than to emphasise on random appearance. This is also discussed in [58]. It is found that for most datasets, performing this re-ordering does improve the result.

Algorithm 4.2 Proposed Algorithm, MM-DPM

- 1) Input: x
- 2) Output: $E[z_{nk}]$
- 3) Initialization: $E[z_{nk}], m_0, \alpha_0, \lambda_0, a_0, b_0, K$
- 4) Repeat until max iteration or convergence,
 - i) Update the **expectations** (MAP of variational posterior):

$$E[\tau_k] \approx \frac{\frac{1}{2} \sum_{n=1}^N E[z_{nk}] + a_0 - 1}{b_0 + \frac{1}{2} (\sum_{n=1}^N (x_n - E[\mu_k])^2 E[z_{nk}] + \lambda_0 (E[\mu_k] - m_0)^2)}$$

$$E[\mu_k] \approx \frac{\sum_{n=1}^N E[z_{nk}] x_n + \lambda_0 m_0}{\sum_{n=1}^N E[z_{nk}] + \lambda_0}$$

$$E[v_k] \approx \frac{\sum_{n=1}^N E[z_{nk}]}{\sum_{n=1}^N \sum_{j=k+1}^K E[z_{nj}] + \alpha_0 - 1}$$
 - ii) Sort $E[v_k]$ for $k = 1, \dots, K$ according to descending order. Use the re-ordering to sort $E[\mu_k]$ and $E[\tau_k]$ accordingly.
 - iii) Compute the cluster assignment:

$$E[z_{nk}] \approx \arg \max_{z_{nk}} \left\{ \ln E[v_k] + \sum_{l=1}^{k-1} \ln(1 - E[v_l]) + \frac{1}{2} \ln E[\tau_k] - \frac{1}{2} E[\tau_k] (x_n - E[\mu_k])^2 \right\} z_{nk}$$
 - iv) Prune $E[\mu_k], E[\tau_k], E[v_k], E[z_{nk}]$ for all $E[v_k] < THR$
 - v) Compute lower bound, $\mathcal{L}$ if necessary

End

4.5 Proposed Future Work

There are three basic steps in the proposed method for learning the hidden variables, $E[\theta]$ in DPM as follows:

- i) Establish the expression for the variational log-posterior distribution of each hidden variable, $\ln q(\theta_j) = E_{\theta_{i \neq j}} [\ln p(x, \theta_j)] + \text{const.}$
- ii) Use Jensen's inequality, $E[\ln \theta] \geq \ln E[\theta]$ to simplify all expectations within the variational log-posterior distribution.
- iii) Take the MAP estimation of the variational log-posterior distribution, $E[\theta] \approx \arg \max_{\theta} \ln q(\theta)$.

4.5.1 Modeling Hyperparameter in DPM

Figure 4.2: DPM with new hidden variable β

Yet, there are two main concerns that remain. Firstly, whether the current method can be extended to other exponential family. Secondly, which hyperparameter values should be used in experiments? Here, these two concerns are solved by extending DPM with a hyperprior using the three basic steps in the generalized proposed method above. One criticism faced early on was how to select values for the mixture model hyperparameters $m_0, \alpha_0, \lambda_0, a_0, b_0$ in DPM or GMM. This is an inherent problem in most Bayesian approach. In Algorithm 4.2, the naive approach is to simply treat the hyperparameters as noninformative, e.g. $m_0 = \lambda_0 = a_0 = b_0 = 1$ and $\alpha_0 = N$.

For discussion purpose, a new hidden variable β is introduced to the joint posterior of DPM to account for the Bayesian inference of hyperparameters b_0 , as seen in Figure 4.2. The actual purpose of β will be explained later on. First, the factorized terms are defined as

$$\begin{aligned} p(X, Z, V, \eta, \tau, \beta) &= p(X | Z, \eta, \tau) p(Z | V) p(V) \\ &\quad p(\eta | \tau) p(\tau | \beta) p(\beta) \end{aligned} \quad (4.45)$$

Inference of β :

Since $E[\tau]$ contains four hyperparameters, $\lambda_0, a_0, b_0, \eta_0$ which can affect the estimation of $E[\tau]$ and in turn $E[Z]$, an ideal way is to automatically infer optimal values for these constants from the data. In Bayesian Regularization [83], the authors suggested reducing the number of hyperparameters to only b_0 while fixing the other hyperparameters $\lambda_0 = 0, a_0 = 1, m_0 = 0$, as in this case. Instead of empirically finding a suitable b_0 , another way is to model it as a hidden variable, β , by defining its hyperposterior as

$$\ln q(\beta) = E_\tau [\ln p(\tau | \beta) + \ln p(\beta)] + \text{const.} \quad (4.46)$$

A Chi-square distribution is used to model the hyperprior distribution for β

$$\ln p(\beta) \propto \left(\frac{r}{2} - 1\right) \ln(\beta) - \frac{\beta}{2} \quad (4.47)$$

where r is the degree of freedom or the hyperhyperparameter. A Gamma likelihood for $p(\tau | \beta)$ is used and the MAP estimate is

$$\begin{aligned} E[\beta] &\approx \arg \max_{\beta} E_\tau [\ln p(\tau | \beta) + \ln p(\beta)] \\ &= \arg \max_{\beta} \sum_{d=1}^D \sum_{k=1}^K (a_0 - 1) \ln E[\tau_{dk}] - \beta E[\tau_{dk}] + \left(\frac{r}{2} - 1\right) \ln(\beta) - \frac{\beta}{2} \end{aligned} \quad (4.48)$$

A closed-form MAP expression can be obtained for $E[\beta]$ in terms of each inde-

pendent assumption of k

$$E[\beta_k] = \frac{r-2}{2 \sum_{d=1}^D \tau_{dk} + 1} \quad (4.49)$$

An intuitive explanation for the choice of hyperprior model using the Chi-square distribution is to consider the role of β . In [83], β is treated as a regularization term and is typically ranged in the lower numerical region of $0 < \beta \leq 1$. Thus, it is intuitive to use an exponential distribution. Instead, a more expressive exponential distribution with a degree of freedom, r is used. When $r = 2$, Chi-square reduces to the exponential distribution and $\beta = 0$. A smaller value of hyperhyperparameter r of the Chi-square distribution allow higher probability of β being in the lower numerical range and vice versa. Also, the hyperprior allows a closed form expression for the MAP estimate of $E[\beta_k]$.

It is noted that this variant of DPM is only used for discussion and not actually used in the experiments.

4.6 Experiments

4.6.1 Synthetic Dataset

The performance of the proposed method, VB-DPM and DP-means are demonstrated on a synthetic case. Five separate clusters of Gaussian distributed 2D points with some overlap are randomly generated as seen in Figure 4.3. The parameter values μ, τ, π used to generate the 4 Gaussian datasets are shown in Table 4.4. The 4 datasets are arranged according to Table 4.3 where the dimension $d = 2$ is fixed but the number of mixture components varies from 2, 3, 4, 5. Each dataset has N instance size with equal sample contribution from each cluster.

Goal 1) Model Selection: A fixed upper limit of truncation level, $K = 15$ is used for all datasets. The number of clusters with $E[v_k] > 0$ after convergence

refer to the model selected. It is observed that the estimated mixture sizes are correctly found for all the 4 datasets as seen in Figure 4.4.

Goal 2) Parameter Estimation: In Table 4.4, a comparison on the parameters μ, τ, π used to generate the 4 datasets as well as the estimated parameters $E[\mu], E[\tau], \pi$ is made. Note that the proposed method use $\pi_k = E[v_k] \prod_{l=1}^{k-1} (1 - E[v_l])$ since $E[v_k]$ is generated, while for VB-DPM and DP-means, they empirically compute $\hat{\pi} = \frac{\sum_{n=1}^N E[z_{nk}]}{\sum_{k=1}^K \sum_{n=1}^N E[z_{nk}]}$. For DP-means, the cluster centers $\hat{\mu}$ are re-estimated after every iterations (refer to related works on DP-means). It is observed that in general, the estimation is very close to ground truth for all methods. However, for dataset #4 both Figure 4.3 and Table 4.4 reveals that DP-means is unable to correctly identify the clusters except for 3 out of 5 clusters. Whereas the proposed method and VB-DPM were mostly able to identify the clusters correctly. It is likely that this is due to DP-means lacking the variance parameter.

Goal 3) Convergence via Lower Bound Check: In Figure 4.5, the convergence of the proposed method is tested by checking its negative lower bound using eqn (4.32) for each iteration. In theory, the computed lower bound values should not decreases. Usually, in the first few iterations there are some redundant mixture components that contain zero elements or zero cluster weight. These components must be pruned out or it will cause the lower bound to compute NaN. Subsequently, after pruning the computed lower bound values either increases or saturates at a value. The number of iterations required by the proposed method to converge for each dataset are shown in Table 4.3. A 2.6GHz laptop is used to perform all experiments on Matlab. It is notable that the proposed method is cheap to run and converges very fast due to available closed-form solution.

Figure 4.3: A randomly generated dataset of 5 Gaussian distributed clusters of 2D points. The Gaussian mixture parameters for generating the plots are given in Table 4.4 (ground truth). Note that the order of clustering is not important.

Figure 4.4: Proposed method correctly identifying the number of mixture components remaining (non-zeros cluster weights) after learning has converged. The initial truncation size is 15.

Figure 4.5: Experiment II: The number of iterations vs the computed lower bound for the synthetic dataset experiment.

Table 4.3: Computation time and iterations needed for proposed method to convergence for synthetic datasets #1,...,#4.

dataset #	N	D	ground truth	K	iterations
1	400	2	2	15	5
2	600	2	3	15	7
3	800	2	4	15	10
4	1000	2	5	15	12

Table 4.4: Ground truth vs estimated hidden variables for the synthetic datasets of Gaussian mixtures. For the array that appear in each box, the rows refers to K no. of mixtures while the columns are D no. of dimensions.

ground truth						proposed method					
#	μ		σ		π	$E[\mu]$		$1/\sqrt{E[\tau]}$		$\pi(E[v])$	
1	2.3	4.5	2.2	4.4	0.50	1.6	4.1	1.8	4.4	0.67	
	10.5	5.5	2.1	5.8	0.50	10.1	6.0	2.6	5.9	0.33	
2	2.3	4.5	2.2	4.4	0.33	2.1	4.3	2.1	4.2	0.32	
	10.5	5.5	2.1	5.8	0.33	10.4	5.1	2.2	5.2	0.35	
	15.5	19.5	3.1	4.8	0.33	15.5	19.9	3.2	4.9	0.33	
3	2.3	4.5	2.2	4.4	0.25	2.1	4.2	2.1	4.1	0.24	
	10.5	5.5	2.1	5.8	0.25	10.0	2.4	2.1	4.3	0.17	
	15.5	15.5	4.1	3.8	0.25	14.3	14.8	3.8	4.1	0.35	
	5.5	19.5	3.1	4.8	0.25	4.8	19.2	2.6	4.3	0.24	
4	2.3	4.5	2.2	4.4	0.20	2.0	4.2	2.1	3.8	0.20	
	10.5	5.5	2.1	5.8	0.20	10.4	4.8	2.1	4.4	0.19	
	15.5	15.5	4.1	3.8	0.20	15.2	15.1	3.9	2.9	0.21	
	5.5	19.5	3.1	4.8	0.20	5.3	19.0	2.5	3.6	0.17	
	10.5	30.5	4.1	3.8	0.20	10.3	30.1	4.5	3.9	0.23	

VB-DPM						DP-means		
#	$E[\mu]$		$1/\sqrt{E[\tau]}$		$\hat{\pi}$	$\hat{\mu}$	-	$\hat{\pi}$
1	3.3	4.7	1.2	5.0	0.30	2.5	4.0	0.50
	10.7	4.8	2.1	6.1	0.70	10.7	5.4	0.50
2	3.3	6.7	1.5	4.4	0.20	2.5	4.0	0.34
	10.6	6.2	2.0	6.2	0.43	11.1	4.7	0.31
	15.2	19.3	3.3	5.0	0.37	15.2	20.0	0.35
3	3.1	4.4	1.7	2.1	0.15	2.2	4.4	0.25
	10.5	4.0	2.1	4.5	0.26	10.6	3.5	0.20
	15.3	15.1	3.9	4.0	0.36	15.1	14.7	0.30
	5.0	19.6	3.0	5.4	0.23	5.8	20.0	0.25
4	2.6	5.3	2.4	2.5	0.17	5.0	15.4	0.16
	10.4	3.9	2.4	5.3	0.2	5.8	3.1	0.32
	15.1	15.4	4.4	3.8	0.21	15.1	14.5	0.24
	5.7	19.8	2.7	6.0	0.23	6.0	25.1	0.13
	10.9	29.4	4.0	4.9	0.19	11.9	31.7	0.15

4.6.2 Expectation of Variational Posteriors: Estimated Lower Bound vs Actual

In the proposed method, the lower bound of variational posteriors expectations i.e. $E[\ln \theta] \geq \ln E[\theta]$ is used to lower the algorithm's computational cost. Earlier a theoretical comparison between the proposed method and variational inference mainly was given for non log-expectation case i.e. $E[\theta]$ in section 4.4.4. In this section, the log-expectations case i.e. $E[\ln \theta]$ using empirical results will be examined.

Using the parameters in Table 4.4 to generate data #2, the values of the expectations and log-expectations obtained using Algorithm 4.1 and 4.2 are recorded after their convergence. Their values are shown in Table 4.5. Similarly, in Table 4.6, the experiment is repeated on the UCI Iris dataset.

It is observed that for $E[\mu], E[\tau], E[\ln \tau]$, the proposed method obtained very close values to variational inference. For $E[\ln v], E[\ln(1 - v)]$, the proposed method generates values that correlates with variational inference. But the values have some discrepancy for some values of $E[\ln v], E[\ln(1 - v)]$. It is likely due to the re-ordering and pruning process used exclusively in Algorithm 4.2.

What is shown in Table 4.5 and 4.6 under VB-DPM are the actual generated values for $E[\ln v], E[\ln(1 - v)]$, while $E[v_k]$ was never generated. Alternatively, $E[v_k] = N_k \left(\frac{1}{1 + \gamma_{k,2}} \right)$ can be also computed using eqn (4.44) after VB-DPM converges. For synthetic dataset #2, $E[v_k] = \begin{bmatrix} 0.12492 \\ 0.14455 \\ 0.16756 \end{bmatrix}$ is computed and if it is substituted to compute the values of $E[v], \ln E[v], E[\ln(1 - v)]$, they are comparable to its corresponding log-expectation generated values.

Table 4.5: Expectations and log-expectations of the variational posteriors computed on synthetic dataset #2 (generated using Table 4.4) using Algorithm 4.1 and 4.2 (proposed). The asterisk beside each row refers to corresponding expectations that were generated by each algorithm.

VB-DPM				proposed method			
$E[\mu]$	15.4 3.8 10.3	19.7 6.3 4.2	*	$E[\mu]$	15.5 2.1 10.3	20.0 4.5 4.4	*
$1/\sqrt{E[\tau]}$	3.4 2.2 2.3	4.6 3.9 4.7	*	$1/\sqrt{E[\tau]}$	3.4 2.3 2.2	4.4 4.5 4.6	*
$E[\ln \tau]$	-2.5 -1.6 -1.6	-3.1 -2.8 -3.1	*	$\ln E[\tau]$	-2.4 -1.6 -1.6	-2.9 -3.0 -3.0	
-	-	-		$E[v]$	0.15 0.16 0.19		*
$E[\ln v]$	-2.1 -2.8 -2.1		*	$\ln E[v]$	-1.9 -1.8 -1.6		
$E[\ln(1 - v)]$	-0.13 -0.064 -0.13		*	$\ln(1 - E[v])$	-0.17 -0.18 -0.21		

Table 4.6: Expectations and log-expectations of the variational posteriors computed on UCI Iris dataset using Algorithm 4.1 and 4.2 (proposed). The asterisk beside each row refers to corresponding expectations that were generated by each algorithm.

VB-DPM					proposed method				
$E[\mu]$	3.0 3.4 2.7	5.4 1.5 4.2	1.9 0.30 1.3	*	$E[\mu]$	3.0 3.4 2.7	5.4 1.5 4.2	2.0 0.26 1.3	*
$1/\sqrt{E[\tau]}$	0.43 0.53 0.44	0.86 0.27 0.72	0.38 0.24 0.29	*	$1/\sqrt{E[\tau]}$	0.43 0.52 0.40	0.85 0.23 0.69	0.32 0.20 0.23	*
$E[\ln \tau]$	1.7 1.2 1.64	0.29 2.6 0.65	1.9 2.8 2.4	*	$\ln E[\tau]$	1.8 1.3 1.7	0.76 2.9 0.32	2.9 3.2 2.3	
-	-	-			$E[v]$	0.21 0.33 0.25			*
$E[\ln v]$	-1.7 -1.9 -1.7			*	$\ln E[v]$	-1.6 -1.1 -1.4			
$E[\ln(1 - v)]$	-0.2 -0.16 -0.21			*	$\ln(1 - E[v])$	-0.23 -0.41 -0.28			

4.6.3 UCI Datasets Cited in [56, 92]

A total 8 UCI datasets which appeared in [56, 92] were used. For baseline, X-means [87], DP-means [56], MAP-DPM [92] and VB-DPM [6, 9] are included. All codes are self-implemented for all methods except MAP-DPM. In order to avoid implementing MAP-DPM, their results are cited. In Table 4.8, 4.9 and 4.10 the average results from 10 re-runs for all the methods are recorded.

Truncation / Parameters: All methods here rely on a truncation or parameter (e.g. K, Ω, ω). In X-means [87], a likelihood for every K is ran and then select the model with the optimal score. $K = 10$ is used for X-means. DP-means requires setting a value for the cluster penalty parameter Ω so that it can converge close to ground truth. Similarly for MAP-DPM the setting of ω is crucial. For simplicity, Ω is manually set. However, these values can automatically be found heuristically [56, 92]. For VB-DPM and proposed method, it is much simpler as K only needs to be larger than ground truth for each datasets. For VB-DPM and proposed method, in order to demonstrate that the truncation is independent from the ground truth, a larger $K = 50$ is used for all the datasets here.

Goal 1) Clustering Accuracy: Cluster accuracy is measured using normalized mutual information (NMI). In Table 4.8, the proposed method is consistently on par with DP-means and in some cases outperform DP-means but slightly poorer than VB-DPM. It is found that overall the proposed method is identical to VB-DPM except for Wine dataset. In terms of average NMI, VB-DPM achieved the best score of 0.462 followed by the proposed method at 0.454 and MAP-DPM’s average cited result at 0.445 (excluding #3 and #5), DP-means at 0.409 and lastly X-means at 0.304. Also for #4 and #7, DP-means outperforms all methods. But both VB-DPM and the proposed method have close results. One possible reason could be due to the natural structure of these two datasets which perhaps favors methods with equal cluster variance sizes.

Goal 2) Model Selection: (Pls refer to [87, 18, 6] for a formal discussion on model selection) Win/Tie/Lose is used to summarize the average estimated result for model selection. The ground truth is provided in Table 4.7. The best result amongst all methods for each dataset is in bold. In Table 4.9, the result for DP-means rely on manually setting the threshold Ω until an estimated cluster size is obtained close to ground truth. Thus DP-means has the best model selection result. As model selection is not reported in [92], it is assumed that MAP-DPM can heuristically obtained optimal parameter ω to estimate the best cluster size. It is found that both VB-DPM and the proposed method could obtain similar cluster size for all methods except for #5. X-means performed poorest but the estimated cluster size is not far from VB-DPM and reasonable.

Goal 3) Computational Efficiency: For the proposed method and VB-DPM, the lower bound is computed for each iteration until it reached saturation for each dataset. The number of iterations it takes to converge is reported in Table 4.10. For MAP-DPM, the reported number of iterations it takes to converge is cited from [92]. For the implementation of DP-means, convergence is met when additional cluster is no longer added for the remaining iterations. Overall, VB-DPM took the longest to converge and DP-means is the fastest, followed by MAP-DPM. The proposed method took about 1x to 2x slower than DP-means took and the proposed method is about 2x faster than VB-DPM.

In Figure 4.6, the convergence of the proposed method using the negative lower bound of eqn (4.32) for 10 re-runs on each dataset were graphically shown. The importance is to check whether the lower bound drops during each iterations. In some datasets, pruning may cause the lower bound to drop slightly in the early iterations. It is noted that for dataset #5, the proposed method may occasionally fail as seen in the graph.

Table 4.7: 8 UCI dataset cited in [56, 92].

#	dataset	instance	attribute	class
1	Wine	178	13	3
2	Iris	150	3	3
3	Soybean	47	35	4
4	Vehicle	846	18	4
5	Balance Scale	625	4	3
6	Parkinson's	195	23	2
7	Breast Cancer	699	10	2
8	Pima	768	8	2

Table 4.8: Normalized Mutual Information (using average score from 10 re-runs)

#	X-means	DP-means	MAP-DPM	VB-DPM	Proposed
1	0.39	0.42	0.86	0.75	0.68
2	0.70	0.84	0.76	0.84	0.84
3	0.64	0.80	-	0.87	0.88
4	0.16	0.20	0.15	0.19	0.18
5	0.13	0.12	-	0.13	0.12
6	0.16	0.13	0.12	0.16	0.17
7	0.21	0.73	0.71	0.69	0.69
8	0.044	0.03	0.07	0.071	0.072
avg. NMI	0.304	0.409	0.445	0.462	0.454

Table 4.9: Model selection

#	X-means	DP-means	MAP-DPM	VB-DPM	Proposed
1	4.2	3	3	3	3.7
2	3.7	3	3	3	2.9
3	5.2	5	-	4.6	3.9
4	4.9	4	4	4.4	5.8
5	4.6	6	-	7.8	4.4
6	5.9	2	2	4	4.6
7	4.7	2	2	3	3.3
8	5.4	2	2	4.2	4.3
W/T/L	0/0/8	6/6/2	6/6/2	2/0/6	2/1/6

Table 4.10: Number of iterations to convergence

#	X-means	DP-means	MAP-DPM	VB-DPM	Proposed
1	-	15	11	26	10
2		7	5	33.2	16
3		5	-	7.8	10
4		23	9	31.6	20
5		8	-	78.4	10
6		15	13	23.8	12
7		6	8	48.2	22
8		17	17	54.4	35
W/T/L	-	4/1/4	3/1/5	0/0/8	2/0/6

4.6.4 Challenging UCI Datasets

For scalability test, another 5 newer and more challenging datasets from UCI were selected as shown in Table 4.11. These datasets range from financial, audio, text to medical. They are more challenging than the UCI datasets in Table 4.7 in terms of instance, attribute and class sizes.

The baseline now includes X-means, DP-means, MM-GMM [65] and VB-DPM. MM-GMM is a Bayesian GMM employed to perform model selection in [65]. In Table 4.12, 4.13 and 4.14 the average results from 10 re-runs for all the baseline methods are implemented and recorded. For dataset #2 and #5, their class ground truth are continuous values instead of discrete value. However, class number are given. Hence, kmeans is used to generate cluster centers for these datasets. For dataset # 4 and #5, only random samples of 2000 and 5000 are used for MM-DPM, VB-DPM and proposed method as they are quite slow to compute.

Goal 1) Clustering Accuracy: In Table 4.12, the NMI scores for all the methods are recorded using an average of 10 re-runs for each datasets. Immediately, from the avg. NMI it is seen that both VB-DPM and the proposed method have very similar score but the former is slighter superior. This is followed by MM-GMM at 0.516 and X-means at 0.404. It is not possible to compute the avg. NMI for DP-means since it cannot converge for dataset #1 and #2.

Goal 2) Model Selection: Table 4.13 shows the estimate cluster size for all methods. The truncation level or parameter used to compute each method is included. Generally, a truncation value for K that is sufficiently larger than ground truth in Table 4.11 is used, but not too large such that it causes a large overhead. The same setting is mostly used for all methods. However, for VB-DPM it cannot converge for some datasets unless K is smaller or larger.

It is found that both VB-DPM and the proposed method computes similar estimated cluster sizes. MM-GMM is usually slightly poorer than both methods

Figure 4.6: The number of iterations vs the computed lower bound for 10 re-runs per dataset, using the proposed method: 8 UCI datasets cited in [56, 92].

but better than X-means. While DP-means obtains best model selection for dataset #3 and #4, its corresponding NMI are poorer.

Goal 3) Computational Efficiency:

In Table 4.14, it is found that MM-DPM takes the least iterations to converge. Both methods are mostly faster than DP-means. An important oversight in common employed UCI datasets is how scalability play a role. When the number of instances, attributes and classes is large, it can affect the computational efficiency a lot. In this case, DP-means appears to suffer from scalability issues. Also, VB-DPM is the slowest to converge despite marginally better avg. NMI than the proposed method. It is observed that the proposed method is about 2x to 7x faster than VB-DPM and the proposed method is comparable to MM-GMM.

It is also shown the convergence graphically of the proposed method in Figure 4.7. It is noted that the proposed method may occasionally fail for dataset #1. Apart from that, there are no major issues repeating the proposed method for all the datasets.

Table 4.11: 5 challenging UCI dataset.

#	year	dataset	instance	attribute	class
1	2014	Dow Jones Index [14]	720	13	30
2	2014	Music by Geography [114]	1059	116	33
3	1998	Pen Handwritten Digit Recog. [1]	10992	16	10
4	1991	Letter Recognition [31]	20000	16	26
5	2011	Location of CT Slices [37]	53500	384	12

Table 4.12: Normalized Mutual Information (using average score from 10 runs)

#	X-means	DP-means	MM-GMM	VB-DPM	Proposed
1	0.46	-	0.70	0.73	0.71
2	0.26	-	0.23	0.23	0.24
3	0.48	0.67	0.65	0.70	0.70
4	0.29	0.34	0.37	0.38	0.37
5	0.53	0.63	0.63	0.64	0.63
avg. NMI	0.404	-	0.516	0.536	0.530

Table 4.13: Model selection (truncation or parameter in bracket)

#	X-means	DP-means	MM-GMM	VB-DPM	Proposed
1	38.1(100)	-	38.3 (150)	31 (300)	32.5(150)
2	34.5 (70)	-	46 (100)	35 (150)	35.8(70)
3	25(50)	10 ($8.9e^4$)	17 (50)	13.3 (20)	11.9(50)
4	35.9(70)	31 ($8.29e^2$)	32.3 (100)	37 (50)	37.2(70)
5	24(50)	17($4.25e^4$)	16.6 (50)	13.5 (50)	13.3 (50)
W/T/L	1/0/4	2/0/1	0/0/5	2/2/3	1/1/4

Table 4.14: Number of iterations to convergence

#	X-means	DP-means	MM-GMM	VB-DPM	Proposed
1	-	-	20	152	22
2		-	16	150	25
3		24	20	118	40
4		45	32	191.5	35
5		69	35	86	42
W/T/L	-	0/0/5	5/0/0	0/0/5	0/0/5

Figure 4.7: The number of iterations vs the computed lower bound for 10 re-runs per dataset using the proposed method on 5 challenging UCI datasets.

4.6.5 Real Image Datasets

Three popular image categorization datasets (Multiview Car Pose [84], Caltech101 [28] and Scene Categories [59]) are included in this experiment. Illustrations of the datasets are shown in Figure 4.8, 4.9 and 4.10. For each image or instance, the recently proposed sparse coding based fisher vector (SC-FV) [66, 70] is used to compute the feature or attribute of each image as seen in Table 4.15. SC-FV is computed by using

$$v = \sum_{n=1}^N \nabla_B \ln p(x_n) = \sum_{n=1}^N (x_n - B\hat{u}_n) \hat{u}_n \quad (4.50)$$

where, $\hat{u}_n$ and B are solved by using an off-the-shelf sparse coding solver [62]. In this case, x_n refers to HoG features uniformly extracted across an image. The dimension of HoG is 1x32 and the cluster size of SC-FV used is 64. Thus a SC-FV of dimension 1x32x64=1x2048 is obtained.

The goal is to estimate the number of classes from each image categorization dataset. The experimental procedure in section 5.4 is used.

Goal 1) Clustering Accuracy: In Table 4.16, VB-DPM has the best avg. NMI at 0.342. The proposed method achieved better NMI than DP-means on both dataset #1 and #3. But performed poorer than DP-means and VB-DPM on dataset #2. It is possibly because the value of THR at 0.01 for pruning at each iteration may not be suitable. MM-GMM lags behind the proposed method but performed better than X-means.

Goal 2) Model Selection: In Table 4.17, the proposed method and DP-means are found to achieve best 2 out of 3 model selection. VB-DPM did not perform as well as the proposed method on dataset #2 and #3 possibly due to slower convergence for larger datasets. MM-DPM generally performed poorer than the proposed method but is better than X-means.

Goal 3) Computational Efficiency: Both the proposed method and MM-

GMM are comparable and about 2x faster than DP-means in terms of convergence speed. VB-DPM is unable to fully converge after more than 200 iterations which took overnight to compute.

Figure 4.11 shows the convergence of the proposed method using negative lower bound for 10 re-runs on the 3 datasets. The proposed method appear to be quite stable when repeating the runs.

Figure 4.8: Two different car model (first 5 column and next 5 column) of the Multiview Carpose dataset arranged into 8 view poses or classes (row). Each class has a minor view change of 5 image sequences for each car model. There are 20 different car models in total.

Figure 4.9: 10 object categories or classes of Caltech 101 dataset. Each row shows an object category and each column is an object instance. There are at least 100 images per class.

Figure 4.10: 15 classes from Scene Categories dataset. Each image shows an indoor or outdoor scene such as bedroom, sub-urban, industrial, kitchen and etc.

Table 4.15: Image Categorization Datasets

#	dataset	images	feature dim.	class
1	Multiview Carpose [84]	800	2048	8
2	Caltech101 [28]	3044	2048	10
3	Scene Categories [59]	4485	2048	15

Table 4.16: Normalized Mutual Information (larger score is better)

#	X-means	DP-means	MM-GMM	VB-DPM	Proposed
1	0.214	0.249	0.246	0.264	0.250
2	0.356	0.418	0.383	0.446	0.398
3	0.307	0.316	0.311	0.317	0.323
avg. NMI	0.292	0.328	0.313	0.342	0.324

Table 4.17: Model selection (truncation or parameter in bracket)

#	X-means	DP-means	MM-GMM	VB-DPM	Proposed
1	25.5 (50)	10 (2069.5)	21.9 (50)	12 (50)	15.4 (50)
2	29 (50)	12 (2053)	13.1 (50)	23 (50)	12.5 (50)
3	30.6 (50)	19 (2062.5)	22.8 (50)	41 (50)	17.6 (50)
W/T/L	0/0/3	2/1/1	0/0/3	0/0/3	2/2/1

Table 4.18: Number of iterations to convergence

#	X-means	DP-means	MM-GMM	VB-DPM	Proposed
1	-	41	16	200+	15
2		48	26	200+	30
3		36	19	200+	16
W/T/L	-	0/0/3	2/1/1	0/0/3	2/1/1

Figure 4.11: The number of iterations vs the computed lower bound for 10 re-runs per dataset, using the proposed method: 3 real datasets.

4.7 Conclusion

Variational inference is widely used in Bayesian nonparametrics models such DPM. It mainly rely on computing expectations of variational posterior to perform iterative learning. However, in the experiments it is found that there are issues with VB-DPM [9] such as higher computational cost and scale poorly to larger datasets. Thus, a new framework is proposed by applying MAP estimation of variational posteriors for learning. To further lower the computational cost, the use of lower bounds to reduce the analytical complexity of the MAP estimation is made. Also a discussion the similarities between the proposed method and variational inference is made. Experimental results shows that the proposed method outperformed variational inference in terms of lower computational cost and comparable in terms of clustering accuracy and model selection.

5 Learning Bayesian Mixture Model with Sparse Coding

5.1 Introduction

There are three main issues with Fisher Vector (FV). Firstly, when GMM is used to model a high dimensional feature vector (e.g. 4096-dim), it will lead to poor fitting. Even as the number of mixture increases, the fit does not improve much [70]. Secondly, very few works have looked into using a different model to improve the performance of FV. A unique case is Liu et al’s Sparse Coding based FV [70], which leads to the third issue. In Liu et al’s approach, an off-the-shelve solver [62] is required. Most Sparse Coding algorithms only consider the problem as a convex optimization [62, 12]. From a non-probabilistic perspective, it is not easy to see Sparse Coding’s relationship with FV [70]. The proposed work is strongly influenced by Liu et al’s work [70]. Although, Sparse Coding based FV was originally proposed to solve the first and second problem ie. address poor fitting and a new model for FV, the proposed work mainly addresses the third problem, ie. using a probabilistic solver for Sparse Coding.

Unlike most Sparse Coding algorithms which consider the problem as a convex optimization, in this chapter Sparse Coding is interpreted as a probabilistic model loosely based on GMM. The new model is called the Sparse Coding based GMM (sGMM). A major contribution is that the proposed algorithm is based on variational method to approximate the hidden variables of sGMM. Then, the learnt sGMM parameters are used to compute a Sparse Coding based FV for image representation. Empirically results show that this Sparse Coding based FV significantly improves the performance over GMM based FV. In comparison with convex optimization solver, using a variational method achieves similar learning performance as far as Sparse Coding based FV is concerned.

5.2 Related Works

As of 2017, there is no existing works that propose using the maximization-maximization algorithm within a variational inference machinery for learning Sparse Coding based Fisher Vector [70]. The most common solvers are the convex optimization based solvers such as [62, 12] used in the sparse coding of Bag-of-Words models [110] (and their extended versions, the spatial-pyramid representations) and used in Sparse Coding based Fisher Vector [70].

The difference between Bag-of-Words and Fisher Vector is that in the former, the output is a count of re-occurrence based on the cluster assignment. For Fisher Vector, the output is a computed vector based on the aggregated sum of difference between each cluster and all the samples. For the sparse coding of Bag-of-Words, it will minimize the reconstruction error of the image representation [110]. For the sparse coding of Fisher vector, it also minimizes the representation error especially when the feature dimension is huge [70].

5.3 Background

5.3.1 GMM based Fisher Vector (GMM-FV)

In its simplest form, GMM based Fisher Vector (GMM-FV) is defined as the gradient of $\ln p(x, \theta)$ over B only [50, 96]. The mean is denoted as $B = \{B_k\}_{k=1}^K \in \mathbb{R}^D$ and the total number of Gaussian clusters in GMM is K .

Although FV [88] was originally defined using a Maximum Likelihood approach, it can alternatively use a Maximum a Posteriori (MAP) approach here. Both approaches are identical when noninformative prior for B is assumed as follows

$$\begin{aligned}
v_k &= \sum_{n=1}^N \nabla_{B_k} \ln p(x_n, B_k, z_{nk}, \pi_k) \\
&= \sum_{n=1}^N \nabla_{B_k} (\ln p(x_n | B_k, z_{nk}) + \ln p(B_k) + \text{const.}) \\
&= \sum_{n=1}^N (x_n - B_k) z_{nk}
\end{aligned} \tag{5.1}$$

In the above, when maximizing $\ln p(x, \theta)$, the factorial terms of $\ln p(x, \theta)$ not related to B are absorbed into a constant.

To further simplify FV, for a shared covariance assumption, ie. $\sigma_k = 1$ and $\frac{\sum_{n=1}^N z_{nk}}{\sum_{n=1}^N \sum_{k=1}^K z_{nk}} = \frac{1}{K}$, each cluster centered at B_k has equal sample size. Thus, z_{nk} can be treated as a constant and GMM-FV can be obtained as follows

$$v_k = \sum_{n=1}^N (x_n - B_k) \tag{5.2}$$

The training of B_k require offline learning using an approximation such as the EM algorithm or variational Bayes [6].

5.3.2 Learning Sparse Coding with Convex Solver

The probabilistic model of Sparse Coding is [70, 82]

$$\begin{aligned}
p(x) &= \int p(x | u, B) p(u) du \\
x &\sim \mathcal{N}(Bu) \\
p(u) &\propto \exp(-\Omega |u|)
\end{aligned} \tag{5.3}$$

The sparse coefficient u is drawn from a zero mean Laplace distribution while x is drawn from a Gaussian distribution with mean Bu .

Instead of computing the integral, Liu et al [70] approximate $p(x)$ using a MAP estimate

$$\begin{aligned}
p(x) &\approx p(x | u^*, B) p(u^*) \\
u^* &= \arg \max_u \ln p(x | u, B) + \ln p(u)
\end{aligned} \tag{5.4}$$

In [70], u^* and B are solved by using an off-the-shelf sparse coding solver [12]

such as the feature-sign search algorithm [62].

5.3.3 Sparse Coding based Fisher Vector (SC-FV)

Similarly to GMM-FV earlier, a Sparse Coding based Fisher Vector (SC-FV) [70] is defined as the gradient of $p(x)$ over B

$$\begin{aligned} v &= \sum_{n=1}^N \nabla_B \ln p(x_n) \\ &= \sum_{n=1}^N (x_n - Bu_n^*) u_n^* \end{aligned} \tag{5.5}$$

5.4 Proposed Model, sGMM

The advantages of proposing a variational inference based solver for SC-FV over a convex optimization solver are:

- i) The theoretical relationship between Sparse Coding and Fisher Vector can now be examined.
- ii) Closed-form solutions can be obtained for SC-FV (abeit under certain constraint). Analytical solutions are often superior to numerical approaches both in terms of accuracy and speed.
- iii) The Bayesian approach proposed here models the hidden parameters as probability distributions. When there exist probability models that closely model the actual dataset, good sGMM modeling will be guaranteed.
- iv) The original work by Liu et al only proposed using a single hidden variable B for computing SC-FV. However, the sGMM model proposed in this work provides a framework for future SC-FV expansion to other hidden variables such as π and u .

The disadvantages of this work over a convex optimization solver are:

- i) there is only one non-zero element used for u_n rather than a small number of non-zero elements.

Figure 5.1: Probabilistic model of Sparse Coding based GMM

- ii) instead of using a Laplace model for enforcing sparsity on u_n (a.k.a L1-norm regularization), a Gaussian model is used for tractability.

5.4.1 Sparse Coding based GMM (sGMM)

A new GMM model known as the Sparse Coding based GMM (sGMM) is presented here as seen in Figure 5.1. The hidden variables are defined as $\phi = \{z, u, B, \pi\}$. The sparse coefficient is denoted as $u = \{u_n\}_{n=1}^N \in \mathbb{R}^K$. The joint distribution of the observed and hidden random variables $p(x, \phi)$ is

$$p(x, z, u, B, \pi) = p(x | B, u, z)p(u)p(B)p(z | \pi)p(\pi) \quad (5.6)$$

A Gaussian distributed prior is used to model u . Using Bayesian approach [6], the probability models of $p(x, \phi)$ are further defined as

$$\begin{aligned} x | u, B, z &\sim \mathcal{N}(Bu)^z \\ B &\sim \mathcal{N}(B_0, \gamma^{-1}) \\ u &\sim \mathcal{N}(u_0, \lambda^{-1}) \\ z | \pi &\sim \text{Mult}(\pi) \\ \pi &\sim \text{Dir}(\alpha_0) \end{aligned} \quad (5.7)$$

The main difference between sGMM and GMM is the additional term, $u \sim \mathcal{N}(u_0, \lambda^{-1})$. Whereas, the main difference between sGMM and Sparse Coding

is due to the fact that $u_n \in \mathbb{R}^K$ is a $1 - of - K$ vector and has a cardinality constraint, $Card(u_n) = 1$. It means that there is only one nonzero element in u_n while the rest of the elements are zero [110]. Also, in the proposed sGMM model, summation of the absolute value of each element did not receive normalization i.e. $|u_n| \neq 1$.

In the inference of $p(x, \phi)$, the main goal is learning B from dataset since FV rely on B . The variational method [6] is applied to perform inference on sGMM. For tractability, the following mean field factorization is used

$$q(z, u, B, \pi) = q(z)q(u, B, \pi) \quad (5.8)$$

The variational log posterior can be obtained by the form [6]

$$\ln q(\phi_j) = E_{i \neq j} [\ln p(X, \phi_j)] + const. \quad (5.9)$$

Using the two expressions above, the variational log posterior distribution for the hidden variables can be obtained as follows

$$\begin{aligned} \ln q(u) &= E_z [\ln p(x | u, B, z) + \ln p(u)] + const. \\ \ln q(B) &= E_z [\ln p(x | u, B, z) + \ln p(B)] + const. \\ \ln q(\pi) &= E_z [\ln p(z | \pi) + \ln p(\pi)] + const. \\ \ln q(z) &= E_{u, B, \pi} [\ln p(x | u, B, z) + \ln p(z | \pi)] + const. \end{aligned} \quad (5.10)$$

As variational posteriors cannot be learnt analytically, approximation techniques such as variational Expectation-Maximization is used [6].

5.4.2 Variational Posterior Learning

A further decomposition on the variational distribution is as follows

$$q(u, B, \pi) = q(u)q(B)q(\pi) \quad (5.11)$$

By taking MAP estimation for the variational log posterior distributions, closed-form expression can be obtained

$$\begin{aligned} u_{nk}^{\hat{}} &= \arg \max_{u_{nk}} \ln q(u_{nk}) \\ &= \frac{B_k x_n E[z_{nk}] + \lambda u_0}{B_k^2 E[z_{nk}] + \lambda} \end{aligned} \quad (5.12)$$

$$\begin{aligned} \hat{B}_k &= \arg \max_{B_k} \ln q(B_k) \\ &= \frac{\sum_{n=1}^N u_{nk} x_n E[z_{nk}] + \gamma B_0}{\sum_{n=1}^N u_{nk}^2 E[z_{nk}] + \gamma} \end{aligned} \quad (5.13)$$

$$\begin{aligned} \hat{\pi}_k &= \arg \max_{\pi_k} \ln q(\pi_k) \text{ s.t. } \sum_k \pi_k = 1 \\ &= \frac{\sum_{n=1}^N E[z_{nk}] + (\alpha_0 - 1)}{\sum_{j=1}^K \sum_{n=1}^N E[z_{nj}]} \end{aligned} \quad (5.14)$$

The hyperparameters $\lambda, u_0, \gamma, B_0, \alpha_0$ are empirically determined. More importantly, $E[z_{nk}]$ must be solved before computing the above estimation. The full expression for the inference of z is given by

$$\begin{aligned} E[z_{nk}] &\propto \underset{u, B, \pi}{E} [\ln p(x \mid u, B, z_k = 1) + \ln p(z_k = 1 \mid \pi)] \\ &= \frac{\exp \left(\ln (E[\pi_k]) - \frac{1}{2} (x_n - E[B_k] E[u_{nk}])^2 \right)}{\sum_{j=1}^K \exp \left(\ln (E[\pi_j]) - \frac{1}{2} (x_n - E[B_j] E[u_{nj}])^2 \right)} \end{aligned} \quad (5.15)$$

Instead of defining moments for variational EM, MAP estimation is used for approximating the expected values as follows

$$\begin{aligned} E[\pi_k] &\approx \hat{\pi}_k \\ E[B_k] &\approx \hat{B}_k \\ E[u_{nk}] &\approx u_{nk}^{\hat{}} \end{aligned} \quad (5.16)$$

The proposed algorithm of learning sGMM is summarized in Algorithm 5.1.

5.4.3 sGMM based Fisher Vector (sGMM-FV)

Taking the MAP of $p(x, \phi)$ over B , the sGMM based Fisher Vector (sGMM-FV) is computed as

$$\begin{aligned}
v_k &= \sum_{n=1}^N \nabla_{B_k} \ln p(x_n, z_{nk}, B_k, u_{nk}, \pi_k) \\
&= \sum_{n=1}^N \nabla_{B_k} (\ln p(x_n | B_k, u_{nk}, z_{nk}) + \ln p(B_k) + \text{const.}) \\
&= \sum_{n=1}^N z_{nk} (x_n - B_k u_{nk}) u_{nk} - \gamma(B_k - B_0)
\end{aligned} \tag{5.17}$$

The difference between sGMM-FV and SC-FV is the presence of prior belief γ , hyperparameters B_0 and cluster assignment z_{nk} . If the assumption $\frac{\sum_{n=1}^N z_{nk}}{\sum_{n=1}^N \sum_{k=1}^K z_{nk}} = \frac{1}{K}$ and $\gamma = 0$ is set to the above expression, an identical form to SC-FV is obtained.

5.4.4 Proof to Analytical Solution

Inference of u :

$$\begin{aligned}
\hat{u} &= \arg \max_u \ln q(u) \\
&= \arg \max_u E_z [\ln p(x | u, B, z) + \ln p(u)] \\
&= \arg \max_u E_z \left[-\frac{1}{2} \sum_{k=1}^K \sum_{n=1}^N (x_n - B_k u_{nk})^2 z_{nk} - \frac{\lambda}{2} (u_{nk} - u_0)^2 \right] \\
&= \sum_{k=1}^K \sum_{n=1}^N (x_n - B_k u_{nk}) B_k E[z_{nk}] - \lambda(u_{nk} - u_0) \\
\hat{u}_{nk} &= \frac{B_k x_n E[z_{nk}] + \lambda u_0}{B_k^2 E[z_{nk}] + \lambda}
\end{aligned} \tag{5.18}$$

Algorithm 5.1 Proposed Method, Sparse Coding based GMM (sGMM)

Input: x

Output: $\hat{B}, \hat{u}$

Initialization: $K, u_{nk}, E[z_{nk}], B_0, \gamma, \alpha_0, u_0, \lambda, \zeta$

For each iteration:

1) compute the cluster mean/basis

$$\hat{B}_k = \frac{\sum_{n=1}^N u_{nk} x_n E[z_{nk}] + \gamma B_0}{\sum_{n=1}^N u_{nk}^2 E[z_{nk}] + \gamma}$$

2) update the expected values

$$\begin{aligned} E[\pi_k] &= \hat{\pi}_k \\ E[B_k] &= \hat{B}_k \\ E[u_{nk}] &= \hat{u}_{nk} \end{aligned}$$

$$E[z_{nk}] = \frac{\exp(\ln(E[\pi_k]) - \frac{1}{2}(x_n - E[B_k]E[u_{nk}])^2)}{\sum_{j=1}^K \exp(\ln(E[\pi_j]) - \frac{1}{2}(x_n - E[B_j]E[u_{nj}])^2)}$$

3) compute the mixture component and sparse coefficient

$$\begin{aligned} \hat{\pi}_k &= \frac{\sum_{n=1}^N E[z_{nk}] + (\alpha_0 - 1)}{\sum_{j=1}^K \sum_{n=1}^N E[z_{nj}]} \\ \hat{u}_{nk} &= \frac{\hat{B}_k x_n E[z_{nk}] + \lambda u_0}{(\hat{B}_k)^2 E[z_{nk}] + \lambda} \end{aligned}$$

End

Inference of B :

$$\begin{aligned}
\hat{B} &= \arg \max_B \ln q(B) \\
&= \arg \max_B E_z [\ln p(x | u, B, z) + \ln p(B)] \\
&= \arg \max_B E_z \left[- \sum_{k=1}^K \sum_{n=1}^N \frac{1}{2} (x_n - B_k u_{nk})^2 z_{nk} - \frac{\gamma}{2} (B_k - B_0)^2 \right] \\
&= \sum_{k=1}^K \sum_{n=1}^N (x_n - B_k u_{nk}) u_{nk} E[z_{nk}] - \gamma (B_k - B_0) \\
\hat{B}_k &= \frac{\sum_{n=1}^N u_{nk} x_n E[z_{nk}] + \gamma B_0}{\sum_{n=1}^N u_{nk}^2 E[z_{nk}] + \gamma}
\end{aligned} \tag{5.19}$$

Inference of π :

$$\begin{aligned}
\hat{\pi} &= \arg \max_{\pi_k} \ln q(\pi_k) \text{ s.t. } \sum_k \pi_k = 1 \\
&= \arg \max_{\pi_k} E_z [\ln p(z | \pi) + \ln p(\pi)] - \zeta (\sum_k \pi_k - 1) \\
&= \arg \max_{\pi_k} E_z \left[\sum_{k=1}^K \sum_{n=1}^N \ln (\pi_k)^{z_{nk}} + \ln (\pi_k)^{(\alpha_0-1)} \right] - \zeta (\sum_k \pi_k - 1) \\
&= \sum_{k=1}^K \sum_{n=1}^N \frac{E[z_{nk}]}{\pi_k} + \frac{(\alpha_0-1)}{\pi_k} - \zeta \\
\hat{\pi}_k &\propto \sum_{n=1}^N (E[z_{nk}] + \alpha_0 - 1) \frac{1}{\zeta} \\
&= \frac{\sum_{n=1}^N E[z_{nk}] + \alpha_0 - 1}{\sum_{j=1}^K \sum_{n=1}^N E[z_{nj}] \zeta}
\end{aligned} \tag{5.20}$$

Inference of z :

$$\begin{aligned}
\hat{z} &= \arg \max_{u, B, \pi} \ln q(z_{nk}) \\
&= \arg \max_z E_{u, B, \pi} [\ln p(x | u, B, z) + \ln p(z | \pi)] \\
&= \arg \max_z E_{u, B, \pi} \left[-\frac{1}{2} (x_n - B_k u_{nk})^2 z_{nk} + \ln (\pi_k) z_{nk} \right] \\
\hat{z}_{nk} &= \arg \max_{z_{nk}} \ln (E[\pi_k]) z_{nk} - \frac{1}{2} (x_n - E[B_k] E[u_{nk}])^2 z_{nk}
\end{aligned} \tag{5.21}$$

5.5 Experiments

5.5.1 Setup

Dataset: The performance of proposed method is evaluated on Multi-View Car [84], Caltech10, Scene Categories and MIT Indoor. When coding an image from raw feature (e.g. pixel or HoG) to FV, random patches from the entire image is

extracted. This is for both case of train and test images.

Image Feature: A single-scale HoG is used as the feature. For each image patch, a 16x16 pixels uniformly extract every 8 pixels. Each patch is used to compute a 2x2 partitioned HOG outputting a 1x32 descriptor with L2 normalization.

sGMM Initialization: The following initializations $u_{nk} = 1, B_0 = 0, \gamma = 0, K \in \{64, 128\}, \alpha_0 = 1, u_0 = 0, \lambda = 1, \zeta = 1, E[z_{nk}] \sim Mult(\pi), \pi_k = \frac{1}{K}$ are used.

SC Initialization: For SC it use $\Omega = 0.4$ for all experiments. A batch size of 1000 samples and default setting according to [62] is used for the rest of the parameters.

sGMM Practical Implementation: Instead of computing the expected value $E[z_{nk}]$, the following is used to compute $E[z] \approx \hat{z} = \arg \max_z \ln q(z)$.

Iterations and Cluster Size: The number of iterations is set to 40 for all training, as there is very minor changes on the results of all methods after 10 iterations. A cluster size of $K \in \{64, 128\}$ is used to run each method. Usually, there is negligible difference in FV improvement when increasing the cluster size beyond $K = 64$.

Fisher Vector: K-means, GMM, SC and sGMM are used to train their respective variable B . Both SC and sGMM further computes $\hat{u}$. The learnt variables B and/or $\hat{u}$ are used to compute FV for each baseline and the proposed method.

Both K-means and GMM compute GMM-FV. The implementation in [67] is used to train B for GMM and K-means. Both SC and sGMM compute SC-FV. The feature-sign algorithm in [62] is used to learn both variables for SC-FV. For sGMM-FV, Algorithm 5.1 is used to learn B with initialization $u_{nk} = 1$ for computing $E[z_{nk}]$ which in turn computes $\hat{u}$. The hyperparameter values are used $u_0 = 0, \lambda = 1$ and $\sum_{n=1}^N z_{nk} = \frac{1}{K}$. Essentially, sGMM-FV is treated as

SC-FV to avoid making any decision for the hyperparameter values. The only discrimination between sGMM-FV and SC-FV is the way their parameters are computed.

Each FV is further normalized by its $L2$ norm, $v_k = \frac{v_k}{\|v_k\|_2}$. Finally, FV coding is the concatenation, $V = [\cdots v_k \cdots]^T$

Evaluation Method: Linear SVM is used for classification on all methods. Classification performance is measured in mean Average Precision (mAP) [50]. The mAP results for each method takes the average of five re-runs.

Computational Cost: Although not shown in the experimental results, the computation time for both SC-FV and sGMM-FV are on par but both are much slower than Kmeans-FV and GMM-FV due to the additional sparse coefficient involved. The computational cost is also increased when K increases. Typically, for SC-FV and sGMM-FV, it require overnight training for both Scene Categories and MIT Indoor dataset.

5.5.2 Discussion on Experimental Results

The performance of the methods for different cluster sizes are shown in Table 5.1, 5.2, 5.3, 5.4. From the experimental results it is concluded that sGMM-FV is at least on-par with SC-FV despite having a cardinality constraint on variable u . This is within expectation since sGMM-FV is solving identical problem to SC-FV albeit using a probabilistic approach. Both methods also significantly outperforms non sparse coefficient enabled FV learners such as GMM and K-means consistently for all datasets.

Multi-view Car: This is the smallest dataset, using 20K features for unsupervised learning. Object bounding box is available. There are 8 bin-pose view classes in total and 400 train images and 400 test images using the setup in [84]. It is observed that both SC-FV and sGMM-FV are comparative and they significantly outperform both GMM and K-means by a huge 15% improvement

in terms of mAP.

Caltech10: In the next dataset, 20K features are used for unsupervised learning. Object bounding box is available. The setup in [67] is used for 10 object classes and 400 train images and 2644 test images. In this dataset, the proposed method sGMM-FV significantly outperformed all baseline methods with 18% vs Kmeans-FV, GMM-FV and 7% vs SC-FV. The poorer performance of SC-FV than sGMM-FV is possible due to insufficient training data when using a convex solver for Sparse Coding.

Scene Categories: This is a well known dataset for outdoor scene recognition. The entire image is used as there is no bounding box information given. There are 75000 feature for unsupervised learning from 15 classes, and there are 1500 train and 2985 test images [59]. Both SC-FV and sGMM-FV outperformed both Kmeans-FV and GMM-FV by a wide margin of 22%.

MIT Indoor: This is another recently used and well known dataset for both indoor and outdoor scene recognition. The entire image is used as there is no bounding box information given. There are 107200 feature for unsupervised learning from 67 classes, and there are 5360 train and 1340 test images [90]. Both SC-FV and sGMM-FV outperformed both Kmeans-FV and GMM-FV by a gain of 5%.

Table 5.1: Image Categorization Datasets

dataset	images			train size	class size
	train	test	total		
Multiview Carpose [84]	400	400	800	20000	8
Caltech101 [28]	400	2644	3044	20000	10
Scene Categories [59]	1500	2985	4485	75000	15
MIT Indoor [90]	5360	1340	6700	107200	67

Table 5.2: Mean average precision of classification accuracy on 8 classes Multiview Car dataset. No. of features used for training FV is 20,000 and there are 400 test images.

mAP	Kmeans-FV	GMM-FV	SC-FV	sGMM-FV
$K = 64$	0.57404	0.57356	0.72426	0.7163
$K = 128$	0.57486	0.57709	0.71346	0.71657

Table 5.3: Mean average precision of classification accuracy on 10 classes Caltech dataset. No. of features used for training FV is 20,000 and there are 2985 test images.

mAP	Kmeans-FV	GMM-FV	SC-FV	sGMM-FV
$K = 64$	0.44432	0.44463	0.55736	0.6258
$K = 128$	0.44396	0.44205	0.55525	0.62676

Table 5.4: Mean average precision of classification accuracy on 15 classes Scene Categories dataset. No. of features used for training FV is 75,000 and there are 2644 test images.

mAP	Kmeans-FV	GMM-FV	SC-FV	sGMM-FV
$K = 64$	0.44565	0.44595	0.66347	0.65703
$K = 128$	0.44701	0.44626	0.66528	0.66591

Table 5.5: Mean average precision of classification accuracy on 67 classes MIT Indoor dataset. No. of features used for training FV is 107,200 and there are 1340 test images.

mAP	Kmeans-FV	GMM-FV	SC-FV	sGMM-FV
$K = 64$	0.14372	0.14251	0.19556	0.19634
$K = 128$	0.14270	0.14623	0.19666	0.20807

5.6 Conclusion

A recently proposed Sparse Coding based Fisher Vector is able to achieved much better image representation performance than a traditional GMM based Fisher Vector. This is due to the presence of a sparse coefficient term which enforces good reconstruction of signal. But because the authors did not propose a solution to Sparse Coding based Fisher Vector, an off-the-shelf Sparse Coding solver is suggested.

In this work, a new solution based on probabilistic modeling is proposed for such a purpose. The model is called the Sparse Coding based GMM. It is similar to GMM and mainly differs by an additional hidden variable for representing the sparse term. Model inference is performed by using variational method to obtain analytical solution to the model's hidden variables.

Experimental results on several image classification datasets show that Sparse Coding based Fisher Vector significantly outperformed a GMM based Fisher Vector. Furthermore, from a Sparse Coding based Fisher Vector perspective, the learning performance of the proposed probabilistic solver is at least on-par with off-the-shelf solver.

At the moment, only the gradient of the basis variable B is used for computing a Sparse Coding based Fisher Vector. In future, it would be interesting to find out how what performance would arise from extending the Sparse Coding based Fisher Vector to more than one parameter.

6 Learning Bayesian Mixture Model with Local Structure

6.1 Introduction

In Field of Experts (FoE) [7, 95], an image patch is a clique. Under Hammersley-Clifford theorem, each clique uses the same potential function and the maximal clique (a neighbourhood of connected cliques) is a product of all cliques. While earlier works using MRF limit their potential function to nearest neighbour relationships, [7, 95, 112] showed that parameters of the MRF potential function can be learnt from training data. Parameter of FoE is trained based on the potential function of the maximal clique. In FoE, a neighborhood of the entire image is used for MRF for image denoising but in the case of image classification, only up to a neighborhood of patches for MRF is considered. The benefit of learning a patch neighbourhood over a single patch is the additional neighborhood information gained. Specifically, a maximal clique having a 4-connectivity neighbourhood for each patch is used. In this chapter, details on how a neighbourhood of patch can be directly used to estimate GMM parameters is given.

6.2 Related Works

6.3 Background

The closest work is the Gaussian mixture MRF (GM-MRF) in [112]. Similar to the proposed work, GMM is used as a potential function in MRF. The main difference is that GMM parameter learning in [112] uses a standard EM algorithm for each image and the MRF is treated as a separate learning problem.

Fortunately, things are much simpler in the proposed method here. The original Gaussian mixture is now extended to the newly proposed Field-of-Gaussian (FoG) mixture to take local samples from its neighbors. FoG parameters esti-

mation are computed in a similar way to the MM algorithm of GMM as seen in the proposed work.

6.3.1 Gaussian mixture model

The conditional distribution term of GMM is [6]

$$p(X | Z, \mu, \Sigma) = \prod_{k=1}^K \mathcal{N}(X | \mu_k, \Sigma_k)^{Z_k} \quad (6.1)$$

The observed variable is denoted $X = \{x_1, \dots, x_N\} \in \mathbb{R}^{N \times D}$. The cluster assignment variable is $Z = \{z_1, \dots, z_N\} \in \{0, 1\}^{N \times K}$ and z_n is a 1-of-K binary vector. The parameters are $\mu = \{\mu_k\}$ and $\Sigma = \{\Sigma_k\}$. Taking a diagonal covariance assumption $\Sigma_k = \sigma_k^2 I$, the multivariate Gaussian distribution can be equally shown to be

$$N(X | \mu, \Sigma) = \prod_{d=1}^D \mathcal{N}(x_d | \mu_d, \sigma_d^2) \quad (6.2)$$

Under this assumption, the condition distribution can be described as a “product of D and K independent univariate Gaussians”,

$$p(X | Z, \mu, \Sigma) = \prod_{d=1}^D \prod_{k=1}^K \mathcal{N}(x_d | \mu_{dk}, \sigma_{dk}^2)^{Z_k} \quad (6.3)$$

GMM is further restricted to a single parameter model by using a specific case of shared diagonal covariance, $\Sigma_k = I$ i.e. $\sigma_k = 1$. The conditional distribution of GMM is now

$$p(X | Z, \mu) = \prod_{d=1}^D \prod_{k=1}^K \mathcal{N}(x_d | \mu_{dk})^{Z_k} \quad (6.4)$$

Thus, it is only necessary to learn one GMM parameter μ in terms of a univariate Gaussian for each dimension of d and k .

6.3.2 Maximization-maximization algorithm

A well known iterative technique for learning GMM parameters is the expectation-maximization (EM) algorithm. However, there are practical issues with EM such as high computation cost when the number of Gaussians is large, singularities as well as requiring good initialization to avoid poor local maxima. Recently, Kurihara et al [57] discussed another technique of interest known as the maximization-maximization (MM) algorithm. Both EM and MM belong to the same class of approximation to posterior distribution where exact expression are difficult to optimize or sampling is too slow for the application. Their main difference is how the distribution over parameters and hidden variables are alternatively estimated. MM is faster to compute than EM since the expected value of the latent variable is replaced by another maximization operation and there is no need to compute the normalization term. In the case of MM, the goal is to alternatively estimate the MAP value for θ^* and Z^* . The objective is given as [57]

$$\theta^* = \arg \max_{\theta} \ln p(Z^*, \theta, X) \leftrightarrow Z^* = \arg \max_Z \ln p(Z, \theta^*, X) \quad (6.5)$$

Assuming “flat priors” for $p(\pi)$ and $p(\mu)$, the joint posterior is now

$$p(Z, \theta, X) = p(X | Z, \mu) p(Z | \pi) \quad (6.6)$$

Where $\theta = \{\mu, \pi\}$. Taking the MAP estimation of μ^* , z_n^* and π^* [67] the followings are obtained

$$\begin{aligned} \mu_{dk}^* &= \arg \max_{\mu} \ln p(X | Z^*, \mu) \\ &= \frac{\sum_{n=1}^N z_{nk}^* \cdot x_{nd}}{\sum_{n=1}^N z_{nk}^*} \end{aligned} \quad (6.7)$$

$$\begin{aligned}
z_n^* &= \arg \max_{z_n} \ln p(x_n | z_n, \mu^*) p(z_n | \pi^*) \\
&= \arg \min_{z_n} \sum_{k=1}^K \left\{ \sum_{d=1}^D \frac{1}{2} (x_{nd} - \mu_{dk}^*)^2 \right. \\
&\quad \left. + \ln \pi_k^* \right\} \cdot z_{nk}
\end{aligned} \tag{6.8}$$

$$\pi_k^* = \frac{\sum_{n=1}^N z_{nk}^*}{\sum_{j=1}^K \sum_{n=1}^N z_{nj}^*} \tag{6.9}$$

The MM algorithm for training GMM [67] works by alternatively computing eqn (6.7), (6.8) and (6.9) for each iteration until convergence.

Figure 6.1: Illustration of a random image patch (maximal clique) as a set of connected subpatches (cliques)

6.4 Proposed Model, FoG

6.4.1 Field-of-Gaussian Mixture Model (FoG)

The concept of higher-order MRF [7, 95] is applied to GMM so that additional information from the neighbourhood can allow a more robust parameter learning. Each subpatch is defined as a clique $c \in C$ and together with its connected neighbour subpatches as a maximal clique as seen in Figure 6.1.

A Field-of-Gaussian (FoG) mixture model is modeled using the definition of maximal clique as follows

$$p(X | Z, \mu) = \frac{1}{\mathcal{Z}(\Theta)} \prod_{c \in C} p(X_c | Z, \mu) \tag{6.10}$$

$\mathcal{Z}(\Theta)$ is a partition function that can be ignored in the MM objective. Follow-

ing the MM algorithm objective in eqn (6.5), the MAP estimate of FoG mean parameter is estimated as

$$\begin{aligned}\mu_{dk}^* &= \arg \max_{\mu} p(X \mid Z^*, \mu) \\ &= \frac{\sum_{c \in C} \sum_{n=1}^N z_{nk}^* \cdot x_{cnd}}{|C| \cdot \sum_{n=1}^N z_{nk}^*}\end{aligned}\tag{6.11}$$

Where $|C|$ is the total number of cliques. The newly found value of μ_k^* in (6.11) is then used to find the MAP estimated value of FoG cluster assignment

$$\begin{aligned}z_n^* &= \arg \min_{z_n} \sum_{k=1}^K \{ \sum_{c \in C} \sum_{d=1}^D \frac{1}{2} (x_{cnd} - \mu_{dk}^*)^2 \\ &\quad + \ln(\pi_k^*) \} \cdot z_{nk}\end{aligned}\tag{6.12}$$

Likewise in eqn (6.7) and (6.8), the MM algorithm for training FoG works by alternatively computing eqn (6.9), (6.11) and (6.12) for each iteration until convergence. The extension from GMM to FoG is linear with only an additional clique summation term to take into consideration. Also, for each clique within a maximal clique, they share the same GMM parameter μ and latent variable z . The method is summarized in Algorithm 6.1.

Algorithm 6.1 Proposed Method, Field-of-Gaussian mixture model (FoG)

Input: $X = \{x_{1,1}, \dots, x_{C,N}\}$

Output: $\theta^* = \{z^*, \mu^*, \pi_k^*\}$

Initialization: z, μ, k

For $i = 1 : I$

M-step:

$$\mu_{dk}^* = \frac{\sum_{c \in C} \sum_{n=1}^N z_{nk}^* \cdot x_{cnd}}{|C| \cdot \sum_{n=1}^N z_{nk}^*}$$

$$\pi_k^* = \frac{\sum_{n=1}^N z_{nk}^*}{\sum_{j=1}^K \sum_{n=1}^N z_{nj}^*}$$

E-step:

$$z_n^* = \arg \min_{z_n} \sum_{k=1}^K \left\{ \sum_{c \in C} \sum_{d=1}^D \frac{1}{2} (x_{cnd} - \mu_{dk}^*)^2 + \ln(\pi_k^*) \right\} \cdot z_{nk}$$

End

6.4.2 Proof to Analytical Solution of FoG

Here the steps to obtain the analytical solution to eqn (6.11) is shown. The goal is to obtain an analytical expression of the following

$$\mu^* = \arg \max_{\mu} p(X \mid Z^*, \mu) \quad (6.13)$$

The partition function $\mathcal{Z}(\Theta)$ in eqn (6.10) can be ignored since only the max value is required to be found. Taking the derivative of eqn (6.10) w.r.t to μ

$$\frac{dp(X \mid Z^*, \mu)}{d\mu} = \sum_{c \in C} \sum_{n=1}^N \sum_{k=1}^K \sum_{d=1}^D (x_{cnd} - \mu_{dk}) \cdot z_{nk}^* \quad (6.14)$$

Let $\frac{dp(X \mid Z^*, \mu)}{d\mu} = 0$,

$$\sum_{c \in C} \sum_{n=1}^N \sum_{k=1}^K \sum_{d=1}^D z_{nk}^* \cdot x_{cnd} - z_{nk}^* \cdot \mu_{dk} = 0 \quad (6.15)$$

$$\sum_{c \in C} \sum_{n=1}^N \sum_{k=1}^K \sum_{d=1}^D z_{nk}^* \cdot x_{cnd} = \sum_{c \in C} \sum_{n=1}^N \sum_{k=1}^K \sum_{d=1}^D z_{nk}^* \cdot \mu_{dk} \quad (6.16)$$

After re-arranging and expressing μ in terms of d and k , the following closed form solution is obtained

$$\mu_{dk}^* = \frac{\sum_{c \in C} \sum_{n=1}^N z_{nk}^* \cdot x_{cnd}}{|C| \cdot \sum_{n=1}^N z_{nk}^*} \quad (6.17)$$

6.4.3 Hierarchical Feature Learning

In the Hubel and Wiesel model in Figure 6.2, researchers found that the visual cortex is made up of a hierarchy of simple and complex cells [94]. If only the first two layers in Figure 6.2 are considered, then the first layer contains the filter response of an image patch I_j after passing the first derivative Gaussian w_j at different orientations

$$s_j = |w_j \cdot I_j| \quad (6.18)$$

The second layer perform a MAX response function which can also be treated as a clustering operation for each orientation

$$C = \max_j s_j \quad (6.19)$$

Inspired by the simple and complex cell model in [94], next a novel hierarchical feature learner that mimic simple cell is described.

The main difference from FoG in Algorithm 6.1 is that each subpatches are now automatically labeled (j_n in Algorithm 2.1) after performing offline training using DPmeans on random subpatches x_{cn} .

Most importantly, instead of a “maximal clique of HOG”, a “maximal clique of DPmean” is used for training FoG. This means that instead of taking the feature (e.g. HoG or raw pixel) of the image subpatch, x_{cn} directly for training FoG, each subpatch now use their respective DPmeans obtained output or cluster center,

Figure 6.2: Hubel and Wiesel model.

Figure 6.3: Graphical flowchart of proposed hierarchical feature learning algorithm

η in Algorithm 2.1. This is done so as to automatically find the optimal number of clusters in the first layer a.k.a Bayesian Nonparametric learning offered by DPmeans. This is important because in the simple and complex cell model, there are very few filters in the simple cell. This is illustrated in Figure 6.3.

Thus, (6.11) and (6.12) are now slightly modified as

$$\mu_{dk}^* = \frac{\sum_{c \in C} \sum_{n=1}^N z_{nk}^* \cdot \eta_{cnd}}{|C| \cdot \sum_{n=1}^N z_{nk}^*} \quad (6.20)$$

$$z_n^* = \arg \min_{z_n} \sum_{k=1}^K \left\{ \sum_{c \in C} \sum_{d=1}^D \frac{1}{2} (\eta_{cnd} - \mu_{dk}^*)^2 + \ln(\pi_k^*) \right\} \cdot z_{nk} \quad (6.21)$$

The proposed hierarchical feature learning is summarized in Algorithm 6.2.

Algorithm 6.2 Proposed Hierarchical Feature Learning

1. Randomly extract subpatches from each image.
 2. Train a DPmeans to obtain a base visual dictionary with self-obtained T clusters.
 3. Randomly extract patches (sets of connected subpatches) from each image.
 4. Label all subpatches within each extracted patch using cluster assignment j_n in Algorithm 2.1. Represent each subpatch with a cluster mean η associated with the estimated label.
 5. Train a FoG (Algorithm 6.1) using the newly represented extracted patches in step 4 for K clusters.
 6. Directly use FoG for a BoW image categorization approach.
-

6.5 Experiments

6.5.1 Field of Gaussian mixture model

Dataset: The performance of FoG, GMM and K-means are evaluated on the Multi-View Car 2D [84] 8 classes bin-pose classification datasets. The dataset contains 2299 image sequences of 20 different cars on a rotating platform (first

10 cars for training and next 10 cars for testing). The number of training images are 400 of 1259 and the number of testing images are 1040. Object bounding box is also used.

Feature: For each clique or patch, image feature is extracted using a 2x2 partitioned HoG every 8 pixels outputting a 1x32 descriptor with L2 normalization. A maximal clique is defined as a center clique with 4 adjacent cliques, with each clique having an overlap of 8 pixels with adjacent another. There are 5 likelihoods per maximal clique in total in this configuration. Alternatively, other type of connectivity can also be considered such as a full 3x3 block or a 5x5 block in [7, 95]. A compact sized maximal clique is chosen as it is noticed that when the size is too large e.g. a 5x5 block, the performance is poorer and it will take significantly longer computation time.

Setup: The baseline comparison are K-means and GMM. A randomly selected 2000 maximal clique per class respectively totalling 20,000 training set is used for unsupervised learning. 40 iterations are used to train FoG and GMM as the cluster size usually converges after 10 iterations for both datasets. Random initialization is used for FoG and GMM. A standard BoW approach is only considered for simplicity. The cluster sizes $k = [100, 200, 300, 500, 1000]$ are used to observe the behavior of each method.

Classification: Linear SVM is used for classification on all methods. Classification performance is measured in mean Average Precision (mAP). Average mAP values from five re-runs are used to record the mAP results for each method. The behavior of the methods for different cluster sizes are shown in Figure 6.4.

Result: In the Multi-view Car experiment in Figure 6.4, FoG outperforms both K-means and GMM for all cluster sizes. All methods behave poorer when cluster size is above $k = 500$ possibly due to overfitting. Table 6.1 shows the optimal performance of all the methods at $k = 300$. The best mAP for FoG is

0.71862 compared to K-means of 0.66516 and GMM of 0.68184. The FoG gain over K-means and GMM is 5.346% and 3.678% respectively. Despite being a single parameter model, GMM outperform K-means for most cluster sizes but lose performance slightly when $k > 800$. GMM has 2.028% gain over Kmean. The mAP result shown here shows that by combining the MM algorithm estimated GMM with MRF, the performance of GMM can improve significantly by up to 3.678%.

Figure 6.4: Evaluation of proposed method, FoG vs GMM and K-means for 400 training and 1040 testing images.

Table 6.1: Best performance on the Multi-view Car dataset [84]

8 class of bin-pose	K-means	GMM	FoG (Proposed)
	$K = 300$	$K = 300$	$K = 300$
Front	0.99437	0.99824	0.99418
Front-Left	0.62753	0.86211	0.90531
Left	0.69789	0.78453	0.87939
Rear-Left	0.45773	0.31384	0.24087
Rear	0.87559	0.96399	0.96527
Rear-Right	0.58441	0.42318	0.60042
Right	0.33067	0.37113	0.46112
Front-Right	0.75306	0.73772	0.70244
mean AP	0.66516	0.68184	0.71862

6.5.2 Hierarchical Feature Learning

The next goal is to show that FoG can be used to perform hierarchical learning and is at least comparable or better than traditional single layer learning such as GMM or K-means.

The Multi-View dataset is used to evaluate the performance of the proposed hierarchical feature learner. For both testing and training only 5 image sequences are used per bin-pose for a total of 400 training and 400 testing images. Previously, 400 training images and 1040 testing images are used for FoG. Thus, the earlier result cannot be a reference here. All other settings remains as in the previous experiment.

The proposed approach can be seen as the 1st level containing a small set of frequently occurring local texture and the 2nd level comprising of a larger local texture fully described by only the 1st level predefined textures.

Based on a 1st level visual dictionary of $T = 19$ clusters obtained using DPmeans on the Multi-View dataset, a FoG is trained using only a combination of the 19 clusters. For comparison, a combination of 5 cliques given 19 choices has 11628 combination. A cluster size of $K = [100, 200, 300, 500, 1000]$ is considered.

In Figure 6.4, it is shown that when using $T = 19$ clusters to train a single layer learner, e.g. K-means, the mAP score is 0.69159. Subsequently, the same set of $T = 19$ clusters is used to learn a FoG for various sizes of K using the proposed hierarchical learner in Algorithm 6.2. It is found that the mAP score improve significantly to best result of 0.84587 at $K = 500$ or a gain of 15.428% as seen in Table 6.2. A single layer learner using K-means is also compared to show how it would perform for various sizes of K as seen in Figure 6.4.

Although, the proposed hierarchical learner only perform slightly better than single layer learner (K-means) at $300 \leq K < 1000$, a novel two layer learner using DPmeans and FoG is proposed.

This approach is able to model a complex texture patch using a set of simple

texture subpatch from as few as 19 clusters only.

Figure 6.5: Evaluation of proposed hierarchical learner using DPmeans and FoG for 400 training and 400 testing images.

Table 6.2: Best performance on the Multi-view Car dataset [84]

8 class of bin-pose	K-means	DPmeans	Hier. Learner (Proposed)
	$K = 1000$	$T = 19$	$K = 300$
Front	1	1	1
Front-Left	0.83463	0.817	0.71317
Left	0.55837	0.6355	0.82647
Rear-Left	0.4414	0.32969	0.4832
Rear	0.9769	0.96379	0.98396
Rear-Right	0.89651	0.41497	0.89604
Right	0.98217	0.50901	0.9085
Front-Right	0.97558	0.8628	0.95566
mean AP	0.83321	0.69159	0.84587

6.6 Conclusion

A new algorithm is proposed to train GMM for an image categorization problem. Although, it is only shown on a single GMM parameter case, it is possible to extend to full GMM parameters. Inspired by the Field of Expert [7, 95] method, the learning of GMM parameter is enhanced with MRF by considering a local neighborhood of maximal clique for learning each GMM sample. The approach is effectively taking the product of individual likelihood of each clique (patch)

within each maximal clique. The approach is called the Field-of-Gaussian (FoG) mixture model. The performance of FoG is tested on the Multi-view Car dataset [84] and the proposed method outperformed both K-means and GMM for all cluster sizes.

The proposed FoG is also extended to hierarchical feature learning. The hierarchical learner here uses the output of DP means to construct a FoG. This two layer feature learning approach is shown to be at least comparable or better than a single layer learner in the experiment. In future, it is interesting to investigate whether adding more layers to FoG can help achieve better performance.

7 Conclusion and Recommendations

7.1 Conclusion

Most of the works here are focused on using the MM algorithm for obtaining unique analytical solution to the variational inference framework for different BMM. The BMM here are the Gaussian mixture model (GMM), Dirichlet process mixture (DPM), sparse coding based Gaussian mixture model (sGMM) and the Field-of-Gaussian (FoG) mixture model.

Firstly a comparison of two different variational posterior learners, MM and EE is made on the Bayesian GMM case. EE is the de facto learner in most algorithms employing the variational inference framework and MM is not widely known. The proposed method using MM essentially perform MAP estimate of variational posteriors to obtain the expectations of variational posteriors. EE on the other hand rely on predefined analytical solution of conjugate priors expectations for computing the expectations of variational posteriors. These two learners are studied in depth and a few important findings was observed. MM is faster to compute with comparable accuracy. This is attributed to the fact that only the lower bound of variational posteriors expectations were computed which result in fewer computation required. Secondly, both EE and MM derive near identical analytical solutions despite their different theoretical background mainly EE rely on conjugate assumption for obtaining expectation while MM is free from such assumption. Thirdly, the analytical solution of MM makes it easy to observe that the Bayesian approach to GMM reduces to a maximum likelihood approach when noninformative priors are assumed.

Next, the unique characteristics of sacrificing model accuracy for speed in MM is exploited for addressing scalability issue in BNP algorithms, in particular DPM. Earlier, both MM and EE obtained very similar analytical solution for GMM. Thus, in theory MM should not be inferior to EE. This claim is now ex-

tended to DPM (BMM with stochastic process). Due to having a more complex model, there is greater difficulty computing expectations of variational posteriors. Similarly to earlier case, only the lower bound of variational posteriors expectations are computed as it significantly reduce the complexity of computation required. The analytical solution of MM for DPM is once again compared with EE [9]. In some case such as the mixture model parameters, both methods have very similar solution. However, for the stochastic process of the cluster assignment, MM has its own unique closed-form solution which is less complex to compute than EE but does not suffer from any drawback such as accuracy. Some recent fast algorithms of DPM are compared with MM and it is found that MM is much faster to converge than EE and also computationally comparable to other methods. Moreover, amongst all baselines the proposed MM approach obtained best overall clustering accuracy and model selection results.

After some success with the MM algorithm, the next focus is on specialized BMM. A new Bayesian model called the sGMM (Figure 5.1) is proposed. sGMM differs from GMM by the additional hidden variable introduced for modeling the sparse coefficient term for each cluster mean of GMM. The learnt parameters of sGMM are used to compute the recently proposed SC-FV [70] for image representation. Originally in [70], this sparse coefficient term was introduced for addressing overfitting issue which GMM based Fisher vector faced when modeling high dimensional feature. However, the sparse coefficient term can also significantly improve the image representation of Fisher vector. The main contribution here is the proposed analytical solution for sGMM using MM. As far as literature survey shows, the proposed method is novel. The proposed solution is successfully compared with a numerical convex optimization solver [62] originally used by SC-FV. It is comparable to the numerical solver and in some case even perform better results in terms of the model accuracy for image representation using the SC-FV.

In literature, most algorithms that employ mixture model do not model the local structure information found around an image patch. Instead, each image patch is treated as having no dependency to the rest of the image. A recent work explain this as the iid representation [16]. Alternatively, they proposed a non-iid representation to capture the dependency known as latent MoG, which is basically a standard Bayesian approach to GMM as covered in chapter 3. A different approach is taken to solve this problem. Inspired by an image prior technique called Field-of-Experts [95] originally for inpainting and image denoising, the same technique is extended to a newly proposed specialized BMM called the FoG. The key difference to latent MoG is that in FoG, it still rely on iid assumption (i.e. non-informative prior) but this iid representation is now modeled by a MRF. A GMM is used for learning the potential function of a MRF. Despite the additional complexity involved, it is straightforward to extend MM to solve FoG. Strictly speaking, FoG is not a BMM but more of a MLE since uniform distribution is assumed for the prior models (the relationship between MLE and BMM is covered in chapter 3). However, this is so as to reduce the complexity cost involved. The experimental results showed that FoG significantly outperform GMM due to the additional local structure information used in the modeling. FoG is further extended to a hierarchical feature learner based on a simple and complex cell model [94] with marginal improvement despite having a compact dictionary size.

7.2 Recommendations for Further Research Works

The focus of this thesis is on the approximate inference of Bayesian mixture model as well as its application to image modeling. Below are some possible directions for future works.

7.2.1 Non-Analytical Approximation

Analytical solution to BMM using variational inference allows good accuracy, efficiency and supports theoretical understanding as seen in chapter 3 and 4. However, there are at least two very recent works [54, 38] that proposed approximating the expectation of variational posteriors numerically. Their main reason is that when working with more complex BMM, analytical solution to the expectation of variational posteriors may not be available. To an extent, the authors is correct. With the advancement of computing power, it is not always necessary to rely on a choice of mathematical convenience for the prior models. For instance in [54], an approximation for the expectation of variational posterior known as the reparameterization trick is found by averaging the function within the expectation and applying the reparameterization $z = \mu + \sigma\epsilon$ as follows

$$E[f(z)] \approx \frac{1}{L} \sum_{l=1}^L f(\mu + \sigma\epsilon_l) \quad (7.1)$$

where $\epsilon_l \sim \mathcal{N}(0, 1)$. The above expression is inspired by the Monte Carlo integration

$$\begin{aligned} E[f(x)] &= \int_a^b f(x) dx \\ &\approx (b-a) \frac{1}{N-1} \sum_{i=1}^N f(x_i) \end{aligned} \quad (7.2)$$

The variational MM algorithm in this thesis rely on the MAP estimation of variational posteriors to approximate the expectations. It can also be as a good starting point to the above reparametrization trick. Instead of solving the MAP estimation analytically, one can alternatively propose efficient numerical solution using the MM algorithm instead.

7.2.2 Bayesian Neural Network

Figure 7.1: Illustration of autoencoder, an unsupervised neural network.

In the introduction of this thesis, some recent works were discussed which combine neural network (perception) and BMM (uncertainty), also known as BNN. Very recent works [32, 33, 106, 38, 54] have started to explore this field. A recent BNN of interest is the variational autoencoder in [54]. An illustration of an autoencoder is shown in Figure 7.1. It was earlier explained in the thesis that the evidence of a Bayesian model can be re-expressed as a sum of KL-Divergence from true posterior and the evidence lower bound

$$\ln p_{\theta}(x) = \mathcal{L}(\theta, \phi; x) + D_{KL}(q_{\phi}(z | x) \parallel pq_{\theta}(z | x)) \quad (7.3)$$

By assuming minimum KL-Divergence, the evidence lower bound becomes

$$\begin{aligned} \ln p_{\theta}(x) &\geq \mathcal{L}(\theta, \phi; x) \\ \mathcal{L}(\theta, \phi; x) &= E_{q_{\phi}(z|x)} [\ln p(x | z)] \end{aligned} \quad (7.4)$$

Now lets refer to the decoder part of autoencoder in Figure 7.1. The likeli-

hood, $\ln p(x | z)$ is modeled using a unit variance Gaussian

$$\begin{aligned}\ln p(x | z) &= \ln N(x; \mu, I) \\ \text{where } \mu &= W_4 h + b_4 \\ h &= \tanh(W_3 z + b_3)\end{aligned}\tag{7.5}$$

The expectation $E_{q_\phi(z|x)}[\cdot]$ is approximately an averaging function as seen in eqn (7.1). From the above expression, it is apparent that in order to obtain the decoder output, the output must be backpropagated from the multiplication with the weights in layer 4 to layer 3 then to the coding layer. This coding layer z , is responsible for cluster assignment. In order to obtain z , z is swapped with x for the encoder case. Thus, it can be seen that the variational inference framework can be interpreted as an autoencoder [54]. A future work could be a further study of this technique.

7.2.3 ConvNet based Fisher Vector

A recent work in 2016 achieved a significant breakthrough that demonstrates the integration of Fisher vector within deep ConvNet. The authors of netVLAD [2] proposed a new architecture layer within ConvNet that computes Fisher Vector using the output from the last convolutional layer. It is shown in Figure 7.2, which is briefly discussed in chapter 2 earlier. The main advantage of combining Fisher vector and ConvNet is that firstly, it is not necessary to rely on hand designed image feature such as SIFT. Also, the approach does not treat Fisher vector and ConvNet as separate algorithms unlike [36]. Lastly, the method in [2] can be fine-tuned to other datasets. There are some possible extensions to netVLAD. Firstly, a BMM (e.g. Bayesian GMM in chapter 3 and BNP in chapter 4) can be used to replace GMM in netVLAD. Similarly, the FoG model in chapter 6 and the sGMM in chapter 5 can also extend GMM in netVLAD.

Figure 7.2: Architecture of netVLAD

7.2.4 Stochastic Variational Inference

The main idea behind stochastic variational inference [47] is not to use the entire dataset to train a BMM, unlike variational inference. This is because if there are several millions of samples, standard variational inference may either take impossibly long to converge or run out of memory. Instead, it is desirable to train in smaller batches much like stochastic gradient descend. At the moment, the standard algorithm for stochastic variational inference require computing the local and global hidden variable for each iteration. From the perspective of a traditional variational inference, there is a missing link between stochastic variational inference and standard variational inference. It is desirable to have a more direct explanation or link between these two.

Author's Publications

Journal Papers

1. Lim, Kart-Leong, and Han Wang. "MAP approximation to the variational Bayes Gaussian mixture model and application." *Soft Computing* (2017): 1-13.
2. Lim, Kart-Leong, and Han Wang. "Sparse coding based Fisher vector using a Bayesian approach." *IEEE Signal Processing Letters* 24.1 (2017): 91-95.
3. Lim, Kart-Leong, and Han Wang. "Fast Approximation of Variational Bayes Dirichlet Process Mixture using the Maximization-Maximization Algorithm". *International Journal of Approximate Reasoning* 93 (2018): 153-177.

Conference Papers

1. Lim, Kart-Leong. "A center surround contrast based feature for object pose estimation." *Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on.* IEEE, 2014.
2. Lim, Kart-Leong, and Han Wang. "Learning a Field-of-Gaussian mixture model for image classification." *Control, Automation, Robotics and Vision (ICARCV), 2016 14th International Conference on.* IEEE, 2016.
3. Lim, Kart-Leong, Han Wang, and Xiaozheng Mou. "Learning Gaussian mixture model with a maximization-maximization algorithm for image classification." *Control and Automation (ICCA), 2016 12th IEEE International Conference on.* IEEE, 2016.

References

- [1] Fevzi Alimoglu, Dr Doc, Ethem Alpaydin, and Yagmur Denizhan. Combining multiple classifiers for pen-based handwritten digit recognition. 1996.
- [2] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307, 2016.
- [3] Hagai Attias. A variational bayesian framework for graphical models. *Advances in neural information processing systems*, 12(1-2):209–215, 2000.
- [4] Anoop Korattikara Balan, Vivek Rathod, Kevin P Murphy, and Max Welling. Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, pages 3438–3446, 2015.
- [5] Taoufik Bdiri, Nizar Bouguila, and Djemel Ziou. Variational bayesian inference for infinite generalized inverted dirichlet mixtures with feature selection and its application to clustering. *Applied Intelligence*, 44(3):507–525, 2016.
- [6] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.
- [7] Andrew Blake, Pushmeet Kohli, and Carsten Rother. *Markov random fields for vision and image processing*. Mit Press, 2011.
- [8] David M Blei and Michael I Jordan. Variational methods for the dirichlet process. In *Proceedings of the twenty-first international conference on Machine learning*, page 12. ACM, 2004.

- [9] David M Blei, Michael I Jordan, et al. Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–144, 2006.
- [10] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.
- [11] Nizar Bouguila, Djemel Ziou, and Jean Vaillancourt. Unsupervised learning of a finite mixture model based on the dirichlet distribution and its application. *IEEE Transactions on Image Processing*, 13(11):1533–1543, 2004.
- [12] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
- [13] Tamara Broderick, Brian Kulis, and Michael Jordan. Mad-bayes: Map-based asymptotic derivations from bayes. In *Proceedings of The 30th International Conference on Machine Learning*, pages 226–234, 2013.
- [14] Michael Scott Brown, Michael J Pelosi, and Henry Dirska. Dynamic-radius species-conserving genetic algorithm for the financial forecasting of dow jones index stocks. In *Machine Learning and Data Mining in Pattern Recognition*, pages 27–41. Springer, 2013.
- [15] Sotirios P Chatzis. A markov random field-regulated pitman–yor process prior for spatially constrained data clustering. *Pattern Recognition*, 46(6):1595–1603, 2013.
- [16] Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid. Approximate fisher kernels of non-iid image models for image categorization. *IEEE transactions on pattern analysis and machine intelligence*, 38(6):1084–1098, 2016.

- [17] Etienne Côme, Latifa Oukhellou, Thierry Denoeux, and Patrice Aknin. Learning from partially supervised data using mixture models and belief functions. *Pattern recognition*, 42(3):334–348, 2009.
- [18] Adrian Corduneanu and Christopher M Bishop. Variational bayesian model selection for mixture distributions. In *Artificial intelligence and Statistics*, volume 2001, pages 27–34. Morgan Kaufmann Waltham, MA, 2001.
- [19] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.
- [20] David B Dahl et al. Modal clustering in a class of product partition models. *Bayesian Analysis*, 4(2):243–264, 2009.
- [21] Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.
- [22] Hal Daumé III. Fast search for dirichlet process mixture models. In *AISTATS*, volume 3244, page 3245, 2007.
- [23] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.
- [24] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

- [25] Wentao Fan and Nizar Bouguila. Variational learning of a dirichlet process of generalized dirichlet distributions for simultaneous clustering and feature selection. *Pattern Recognition*, 46(10):2754–2769, 2013.
- [26] Wentao Fan, Nizar Bouguila, and Djemel Ziou. Variational learning for finite dirichlet mixture models and applications. *IEEE transactions on neural networks and learning systems*, 23(5):762–774, 2012.
- [27] Jason Farquhar, Sandor Szedmak, Hongying Meng, and John Shawe-Taylor. Improving" bag-of-keypoints" image categorisation: Generative models and pdf-kernels. 2005.
- [28] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.
- [29] Basura Fernando, Elisa Fromont, Damien Muselet, and Marc Sebban. Accurate visual word construction using a supervised approach. In *Image and Vision Computing New Zealand (IVCNZ), 2010 25th International Conference of*, pages 1–7. IEEE, 2010.
- [30] Basura Fernando, Elisa Fromont, Damien Muselet, and Marc Sebban. Supervised learning of gaussian mixture models for visual vocabulary generation. *Pattern Recognition*, 45(2):897–907, 2012.
- [31] Peter W Frey and David J Slate. Letter recognition using holland-style adaptive classifiers. *Machine learning*, 6(2):161–182, 1991.
- [32] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015.

- [33] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Insights and applications. In *Deep Learning Workshop, ICML*, 2015.
- [34] Samuel J Gershman and David M Blei. A tutorial on bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12, 2012.
- [35] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.
- [36] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *European Conference on Computer Vision*, pages 392–407. Springer, 2014.
- [37] Franz Graf, Hans-Peter Kriegel, Matthias Schubert, Sebastian Pölsterl, and Alexander Cavallaro. 2d image registration in ct images using radial image descriptors. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2011*, pages 607–614. Springer, 2011.
- [38] Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.
- [39] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2009.
- [40] Maya R Gupta and Yihua Chen. *Theory and use of the EM algorithm*. Now Publishers Inc, 2011.
- [41] Tom SF Haines and Tao Xiang. Background subtraction with dirichlet-process mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(4):670–683, 2014.

- [42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
- [43] Felix Heinzl and Gerhard Tutz. Clustering in linear mixed models with approximate dirichlet process mixtures using em algorithm. *Statistical Modelling*, 13(1):41–67, 2013.
- [44] Felix Heinzl and Gerhard Tutz. Additive mixed models with approximate dirichlet process mixtures: the em approach. *Statistics and Computing*, 26(1-2):73–92, 2016.
- [45] James Hensman, Magnus Rattray, and Neil D Lawrence. Fast nonparametric clustering of structured time-series. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):383–393, 2015.
- [46] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
- [47] Matthew D Hoffman, David M Blei, Chong Wang, and John William Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347, 2013.
- [48] Michael C Hughes and Erik Sudderth. Memoized online variational inference for dirichlet process mixture models. In *Advances in Neural Information Processing Systems*, pages 1133–1141, 2013.
- [49] Hemant Ishwaran and Lancelot F James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173, 2001.
- [50] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *Computer*

- Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311. IEEE, 2010.
- [51] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.
 - [52] Yushi Jing and Shumeet Baluja. Visualrank: Applying pagerank to large-scale image search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1877–1890, 2008.
 - [53] Michael I Jordan. Bayesian nonparametric learning: Expressive priors for intelligent systems. *Heuristics, probability and causality: A tribute to Judea Pearl*, 11:167–185, 2010.
 - [54] Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *Second International Conference on Learning Representations, ICLR*, 2014.
 - [55] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
 - [56] Brian Kulis and Michael I Jordan. Revisiting k-means: New algorithms via bayesian nonparametrics. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 513–520, 2012.
 - [57] Kenichi Kurihara and Max Welling. Bayesian k-means as a maximization-expectation algorithm. *Neural computation*, 21(4):1145–1172, 2009.

- [58] Kenichi Kurihara, Max Welling, and Yee Whye Teh. Collapsed variational dirichlet process mixture models. In *IJCAI*, volume 7, pages 2796–2801, 2007.
- [59] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, 2006.
- [60] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [61] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [62] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808, 2006.
- [63] Xiao-Chen Lian, Zhiwei Li, Changhu Wang, Bao-Liang Lu, and Lei Zhang. Probabilistic models for supervised dictionary learning. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2305–2312. IEEE, 2010.
- [64] Kart-Leong Lim and Han Wang. Learning a field of gaussian mixture model for image classification. In *Control, Automation, Robotics and Vision (ICARCV), 2016 14th International Conference on*, pages 1–5. IEEE, 2016.
- [65] Kart-Leong Lim and Han Wang. Map approximation to the variational bayes gaussian mixture model and application. *Soft Computing*, pages 1–13, 2017.

- [66] Kart-Leong Lim and Han Wang. Sparse coding based fisher vector using a bayesian approach. *IEEE SIGNAL PROCESSING LETTERS*, 24(1):91, 2017.
- [67] Kart-Leong Lim, Han Wang, and Xiaozheng Mou. Learning gaussian mixture model with a maximization-maximization algorithm for image classification. In *Control and Automation (ICCA), 2016 12th IEEE International Conference on*, pages 887–891. IEEE, 2016.
- [68] Haibin Ling and Stefano Soatto. Proximity distribution kernels for geometric context in category recognition. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.
- [69] Jun Liu and Haili Zhang. Image segmentation using a local gmm in a variational framework. *Journal of mathematical imaging and vision*, 46(2):161–176, 2013.
- [70] Lingqiao Liu, Chunhua Shen, Lei Wang, Anton van den Hengel, and Chao Wang. Encoding high dimensional local features by sparse coding based fisher vectors. In *Advances in Neural Information Processing Systems*, pages 1143–1151, 2014.
- [71] Zhanyu Ma and Arne Leijon. Bayesian estimation of beta mixture models with variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2160–2173, 2011.
- [72] Zhanyu Ma, Pravin Kumar Rana, Jalil Taghia, Markus Flierl, and Arne Leijon. Bayesian estimation of dirichlet mixture model with variational inference. *Pattern Recognition*, 47(9):3143–3157, 2014.
- [73] Zhanyu Ma, Andrew E Teschendorff, Arne Leijon, Yuanyuan Qiao, Honggang Zhang, and Jun Guo. Variational bayesian matrix factorization for

- bounded support data. *IEEE transactions on pattern analysis and machine intelligence*, 37(4):876–889, 2015.
- [74] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.
- [75] Nobuyuki Morioka and Shinichi Satoh. Building compact local pairwise codebook with joint feature space clustering. In *Computer Vision ECCV 2010*, pages 692–705. Springer, 2010.
- [76] Kevin P Murphy. Conjugate bayesian analysis of the gaussian distribution. *def*, 1(2 σ 2):16, 2007.
- [77] Radford M Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.
- [78] Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.
- [79] Thanh Minh Nguyen and QM Jonathan Wu. A non-parametric bayesian model for bounded data. *Pattern Recognition*, 48(6):2084–2095, 2015.
- [80] Tien Thanh Nguyen, Thi Thu Thuy Nguyen, Xuan Cuong Pham, and Alan Wee-Chung Liew. A novel combining classifier method based on variational inference. *Pattern Recognition*, 49:198–212, 2016.
- [81] Frank Nielsen and Vincent Garcia. Statistical exponential families: A digest with flash cards. *arXiv preprint arXiv:0911.4863*, 2009.
- [82] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

- [83] Dirk Ormoneit and Volker Tresp. Improved gaussian mixture density estimates using bayesian penalty terms and network averaging. In *NIPS*, volume 95, pages 542–548, 1995.
- [84] Mustafa Ozuysal, Vincent Lepetit, and Pascal Fua. Pose estimation for category specific multiview object localization. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 778–785. IEEE, 2009.
- [85] John Paisley, Chong Wang, David M Blei, and Michael I Jordan. Nested hierarchical dirichlet processes. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):256–270, 2015.
- [86] Mattis Paulin, Matthijs Douze, Zaid Harchaoui, Julien Mairal, Florent Perronnin, and Cordelia Schmid. Local convolutional features with unsupervised training for image retrieval. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.
- [87] Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, volume 1, 2000.
- [88] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*, pages 1–8. IEEE, 2007.
- [89] Florent Perronnin and Diane Larlus. Fisher vectors meet neural networks: A hybrid classification architecture. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3743–3752, 2015.
- [90] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 413–420. IEEE, 2009.

- [91] Carl Edward Rasmussen. The infinite gaussian mixture model. In *Advances in neural information processing systems*, pages 554–560, 2000.
- [92] Yordan P Raykov, Alexis Boukouvalas, Max A Little, et al. Simple approximate map inference for dirichlet processes mixtures. *Electronic Journal of Statistics*, 10(2):3548–3578, 2016.
- [93] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [94] Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–1025, 1999.
- [95] Stefan Roth and Michael J Black. Fields of experts: A framework for learning image priors. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 860–867. IEEE, 2005.
- [96] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International journal of computer vision*, 105(3):222–245, 2013.
- [97] Jayaram Sethuraman. A constructive definition of dirichlet priors. *Statistica sinica*, pages 639–650, 1994.
- [98] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.

- [99] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep fisher networks for large-scale image classification. In *Advances in neural information processing systems*, pages 163–171, 2013.
- [100] Vladyslav Sydorov, Mayu Sakurada, and Christoph H Lampert. Deep fisher kernels-end to end learning of the fisher kernel gmm parameters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1402–1409, 2014.
- [101] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Sharing clusters among related groups: Hierarchical dirichlet processes. In *NIPS*, pages 1385–1392, 2004.
- [102] D Michael Titterton. The em algorithm, variational approximations and expectation propagation for mixtures. *Mixtures: Estimation and Applications*, 896, 2011.
- [103] Dimitris G Tzikas, Aristidis C Likas, and Nickolaos P Galatsanos. The variational approximation for bayesian inference. *Signal Processing Magazine, IEEE*, 25(6):131–146, 2008.
- [104] Francesco Visin, Kyle Kastner, Kyunghyun Cho, Matteo Matteucci, Aaron Courville, and Yoshua Bengio. Renet: A recurrent neural network based alternative to convolutional networks. *arXiv preprint arXiv:1505.00393*, 2015.
- [105] Chong Wang and David M Blei. Truncation-free online variational inference for bayesian nonparametric models. In *Advances in neural information processing systems*, pages 413–421, 2012.
- [106] Hao Wang and Dit-Yan Yeung. Towards bayesian deep learning: A framework and some existing methods. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3395–3408, 2016.

- [107] Lianming Wang and David B Dunson. Fast bayesian inference in dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 20(1):196–216, 2011.
- [108] Xin Wei and Chunguang Li. The infinite student’s t-mixture for robust modeling. *Signal Processing*, 92(1):224–234, 2012.
- [109] Max Welling and Kenichi Kurihara. Bayesian k-means as a maximization-expectation algorithm. In *SDM*, pages 474–478. SIAM, 2006.
- [110] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801. IEEE, 2009.
- [111] Hao Zhang, Alexander C Berg, Michael Maire, and Jitendra Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2126–2136. IEEE, 2006.
- [112] Ruoqiao Zhang, Charles Bouman, Jean-Baptiste Thibault, Ken D Sauer, et al. Gaussian mixture markov random field for image denoising and reconstruction. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 1089–1092. IEEE, 2013.
- [113] Shiliang Zhang, Qingming Huang, Yijuan Lu, Gao Wen, and Qi Tian. Building pair-wise visual word tree for efficient image re-ranking. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 794–797. IEEE, 2010.
- [114] Fang Zhou, Q Claire, and Ross D King. Predicting the geographical origin of music. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 1115–1120. IEEE, 2014.

- [115] Jinlin Zhu, Zhiqiang Ge, and Zhihuan Song. Variational bayesian gaussian mixture regression for soft sensing key variables in non-gaussian industrial processes. *IEEE Transactions on Control Systems Technology*, 25(3):1092–1099, 2017.