

Contributions to music semantic analysis and its acceleration techniques

Boyang Gao

▶ **To cite this version:**

Boyang Gao. Contributions to music semantic analysis and its acceleration techniques. Other. Ecole Centrale de Lyon, 2014. English. NNT : 2014ECDL0044 . tel-01170652

HAL Id: tel-01170652

https://theses.hal.science/tel-01170652

Submitted on 2 Jul 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.





THESE

pour obtenir le grade de
DOCTEUR DE L'ECOLE CENTRALE DE LYON
Spécialité : Informatique

présentée et soutenue publiquement par

Boyang Gao

Contributions to Music Semantic Analysis and Its Acceleration Techniques

---

Ecole Doctorale InfoMaths

Directeur de thèse : Liming Chen
Co-directeur de thèse : Emmanuel Dellandréa

JURY

Pr. Frédéric BIMBOT	IRISA	Rapporteur
Pr. Hongying MENG	Brunel University, UK	Rapporteur
Pr. Jean-Paul HATON	Université Henri Poincare	Examinateur
Pr. Liming CHEN	Ecole Centrale de Lyon	Directeur de thèse
Dr. Emmanuel DELLANDRÉA	Ecole Centrale de Lyon	Co-directeur de thèse

Numéro d'ordre :

Contents

Abstract……..……..……..……..…….. i

Résumé…… ii

1. Introduction……..……..……..…….. 1
   1.1. Research Context……..……..…….. 1
   1.2. Problem and Objective……..……..……..... 2
   1.3. Our Approach and Contributions……..…….. 3
   1.4. Organization of the Thesis……..…….…… 6
2. Literature Review……..……..……..…….. 8
   2.1. Audio Features……..……..……..... 8
      2.1.1. Low level feature survey……..……..……... 9
      2.1.2. Low level feature modeling……..……..……... 17
         2.1.2.1. K-means dictionary learning……..……... 18
         2.1.2.2. GMM dictionary learning……..…… 20
         2.1.2.3. Gaussian super vector……..……. 21
      2.1.3. Mid-level features survey……..……..……... 23
   2.2. Audio-Based Musical Mood Detection Systems Survey……..……... 30
      2.2.1. Representation of emotions……..……..……..... 30
      2.2.2. Emotion classification……..……..……. 34
      2.2.3. Emotion regression……..……..…… 36
   2.3. Conclusion……..……..……. 37
3. Acceleration for Low Level Feature Modeling……..……..... 38
   3.1. Introduction……..……..…….. 39
   3.2. K-means and EM Algorithm in Matrix Format……..…… 41
      3.2.1. K-means in matrix format……..……..…….. 41
      3.2.2. EM for GMM in matrix format……..……..……... 42
      3.2.3. MAP in matrix format……..…… 44
   3.3. MapReduce Acceleration……..……..……. 45
      3.3.1. MapReduce structure……..……..…….. 45
      3.3.2. Hadoop architecture……..……..……... 45
         3.3.2.1. Hadoop distributed file system (HDFS) ……..…… 46
         3.3.2.2. Hadoop MapReduce……..……..……... 47
      3.3.3. MapReduce K-means in Matrix Format……..…… 48
         3.3.3.1. Map phase……..……..…….. 48
         3.3.3.2. Reduce phase……..……..……. 49
      3.3.4. MapReduce EM in matrix format……..…….. 50

3.3.4.1. Map phase…… 50
3.3.4.2. Reduce phase…… 50
3.3.5. MapReduce in Spark…… 51
3.4. Performance Tuning…… 52
3.4.1. Nvidia GPU…… 53
3.4.2. Hadoop and spark cluster…… 61
3.4.3. K-means and GMM refinement…… 63
3.5. Experiments on Music Genre and Mood Classification…… 64
3.5.1. Experiment setups…… 64
3.5.1.1. Computer Configuration…… 64
3.5.1.2. Datasets and Features…… 65
3.5.2. Results and Analysis…… 66
3.5.2.1. Single computer…… 66
3.5.2.2. Hadoop and Spark Clustering…… 70
3.6. Conclusion…… 70
4. Sparse Music Decomposition with MIDI Dictionary and Musical Knowledge…… 72
4.1. Introduction…… 72
4.2. Sparse Decomposition of Music Using a Musical Dictionary…… 73
4.2.1. Overview of the approach…… 73
4.2.2. MIDI musical dictionary…… 74
4.2.3. Sparse representation and positive constraint matching pursuit…… 75
4.2.4. A musical histogram as audio feature…… 78
4.2.5. Experiments and Results…… 78
4.2.5.1. Verification of the decomposition with MIDI dictionary…… 78
4.2.5.2. Mood classification with musical histogram feature…… 80
4.2.5.3. Results and analysis…… 81
4.3. Sparse Decomposition with Note Statistics…… 87
4.3.1. Frame level sparse decomposition…… 87
4.3.2. Global level optimal note path searching…… 91
4.3.3. Experiment and results…… 94
4.4. Conclusion…… 101
5. Conclusions and Future Work…… 103
5.1. Conclusion…… 103
5.2. Perspective and Future Work…… 107
Publications…… 108
Bibliography…… 109

List of Figures

Figure 2.1: main part of human cochlea..... 9
Figure 2.2: BoW based music classification framework..... 18
Figure 2.3: example of 2 dimensional k-means clustering result with 3 centers..... 19
Figure 2.4: Mikey mouse data example where GMM works better than k-means..... 20
Figure 2.5: GSV based classification framework..... 22
Figure 2.6: simple example of MAP estimation..... 23
Figure 2.7: NMF for mult-F0 estimation..... 28
Figure 2.8: Plutchik's emotional wheel..... 31
Figure 2.9: Two-dimensional emotion space and basic emotions..... 33
Figure 2.10: Three-dimensional emotion space and 6 basic emotions..... 33
Figure 3.1: HDFS Architecture 47
Figure 3.2: Hadoop MapReduce Architecture 48
Figure 3.3: OpenMP execution time vs. batch size 53
Figure 3.4: CUDA thread parallel architecture 54
Figure 3.5: CUDA memory hierarchies 55
Figure 3.6: Local memory banks with/without conflicts 56
Figure 3.7: Global memory access patterns 57
Figure 3.8: Time consumption comparison between column and row major order..... 58
Figure 3.9: Time consumption of 16x16 thread block dimension..... 59
Figure 3.10: Time consumption of 1.256 thread block dimension..... 59
Figure 3.11: Time consumption of CUBLAS sgemm with matrix transpose..... 60
Figure 3.12: Time consumption of CUBLAS sgemm without matrix transpose..... 60
Figure 3.13: K-means time per iteration vs. different data split size..... 62
Figure 3.14: K-mean time per iteration vs. number cores in cluster..... 62
Figure 3.15: GMM time per iteration vs. number cores in cluster..... 63
Figure 4.1: Musical histogram feature extraction flow chart..... 74
Figure 4.2: Note precision vs. recall..... 79
Figure 4.3: Note precision vs. recall with octave tolerance..... 80

Figure 4.4: Multiple candidate selection example..... 90
Figure 4.5: Optimal path decoding example..... 93
Figure 4.6: Note precision vs. recall of the two improvements..... 97

List of Tables

Table 2.1: Advanced emotion combination..... 31
Table 2.2: MIREX emotion clusters..... 32
Table 3.1: Fine tuned K-mean and GMM execute on different GPUs..... 61
Table 3.2: Cluster Computer Configurations..... 65
Table 3.3: Classical music dataset mood distribution..... 66
Table 3.4: Execution time and accuracy of different implementation for music genre classification..... 68
Table 3.5: Execution time and accuracy of different implementation for music mood classification..... 68
Table 3.6: Running time per iteration for mood GMM training..... 69
Table 3.7: CUBLAS mood classification confusion matrix (actual classes in rows, predicted classes in columns) 69
Table 3.8: ACML mood classification confusion matrix (actual classes in rows, predicted classes in columns)…..... 69
Table 3.9: OpenSMILE mood classification confusion matrix (actual classes in rows, predicted classes in columns)…..... 70
Table 4.1: Classical music dataset mood distribution..... 81
Table 4.2: APM dataset mood distribution..... 81
Table 4.3: Xiao [Xiao et al. 2008b] confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset..... 82
Table 4.4: OpenSMILE confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset..... 82
Table 4.5: GSV confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset..... 83
Table 4.6: General MIDI confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset..... 83
Table 4.7: Logic Pro 9 confusion matrix. (actual classes in rows, predicted classes in columns) using the classical dataset..... 83

Table 4.8: OpenSMILE fused with PCMP confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset..... 84
Table 4.9: GSV fused with PCMP confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset..... 84
Table 4.10: OpenSMILE confusion matrix (actual classes in rows, predicted classes in columns) using the APM dataset..... 85
Table 4.11: GSV confusion matrix (actual classes in rows, predicted classes in columns) using the APM dataset..... 85
Table 4.12: Logic Pro 9 confusion matrix (actual classes in rows, predicted classes in columns) using the APM dataset..... 86
Table 4.13: OpenSmile fused with PCMP confusion matrix..... 86
Table 4.14: GSV fuse PCMP confusion matrix (actual classes in rows, predicted classes in columns) using the APM dataset..... 86
Table 4.15: Average multiple pitch estimation performance on MIREX2007 dataset..... 95
Table 4.16: Average multiple pitch estimation performance on MUS dataset..... 96
Table 4.17: PCMP+multiple candidate confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset..... 98
Table 4.18: PCMP+multiple candidates+optimal path confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset..... 98
Table 4.19: GSV fused with PCMP+multiple candidates+optimal path confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset..... 99
Table 4.20: PCMP+multiple candidates confusion matrix (actual classes in rows, predicted classes in columns) using the APM dataset..... 99
Table 4.21: PCMP+multiple candidates+optimal path confusion matrix (actual classes in rows, predicted classes in columns) using the APM dataset..... 99
Table 4.22: GSV fuse PCMP+multiple candidates+optimal path confusion matrix (actual classes in rows, predicted classes in columns) using the APM dataset..... 100
Table 4.23: Results on all data sets..... 101

Abstract

Digitalized music production exploded in the past decade. Huge amount of data drives the development of effective and efficient methods for automatic music analysis and retrieval. This thesis focuses on performing semantic analysis of music, in particular mood and genre classification, with low level and mid level features since the mood and genre are among the most natural semantic concepts expressed by music perceivable by audiences. In order to delve semantics from low level features, feature modeling techniques like K-means and GMM based BoW and Gaussian super vector have to be applied. In this big data era, the time and accuracy efficiency becomes a main issue in the low level feature modeling. Our first contribution thus focuses on accelerating k-means, GMM and UBM-MAP frameworks, involving the acceleration on single machine and on cluster of workstations. To achieve the maximum speed on single machine, we show that dictionary learning procedures can elegantly be rewritten in matrix format that can be accelerated efficiently by high performance parallel computational infrastructures like multi-core CPU, GPU. In particular with GPU support and careful tuning, we have achieved two magnitudes speed up compared with single thread implementation. Regarding data set which cannot fit into the memory of individual computer, we show that the k-means and GMM training procedures can be divided into map-reduce pattern which can be executed on Hadoop and Spark cluster. Our matrix format version executes 5 to 10 times faster on Hadoop and Spark clusters than the state-of-the-art libraries.

Beside signal level features, mid-level features like harmony of music, the most natural semantic given by the composer, are also important since it contains higher level of abstraction of meaning beyond physical oscillation. Our second contribution thus focuses on recovering note information from music signal with musical knowledge. This contribution relies on two levels of musical knowledge: instrument note sound and note co-occurrence/transition statistics. In the instrument note sound level, a note dictionary is firstly built

from Logic Pro 9. With the musical dictionary in hand, we propose a positive constraint matching pursuit (PCMP) algorithm to perform the decomposition. In the inter-note level, we propose a two stage sparse decomposition approach integrated with note statistical information. In frame level decomposition stage, note co-occurrence probabilities are embedded to guide atom selection and to build sparse multiple candidate graph providing backup choices for later selections. In the global optimal path searching stage, note transition probabilities are incorporated. Experiments on multiple data sets show that our proposed approaches outperform the state-of-the-art in terms of accuracy and recall for note recovery and music mood/genre classification.

Résumé

La production et la diffusion de musique numérisée ont explosé ces dernières années. Une telle quantité de données à traiter nécessite des méthodes efficaces et rapides pour l'analyse et la recherche automatique de musique. Cette thèse s'attache donc à proposer des contributions pour l'analyse sémantique de la musique, et en particulier pour la reconnaissance du genre musical et de l'émotion induite (ressentie par l'auditoire), à l'aide de descripteurs de bas-niveau sémantique mais également de niveau intermédiaire. En effet, le genre musical et l'émotion comptent parmi les concepts sémantiques les plus naturels perçus par les auditoires. Afin d'accéder aux propriétés sémantiques à partir des descripteurs bas-niveau, des modélisations basées sur des algorithmes de types K-means et GMM utilisant des BoW et Gaussian super vectors ont été envisagées pour générer des dictionnaires. Compte-tenu de la très importante quantité de données à traiter, l'efficacité temporelle ainsi que la précision de la reconnaissance sont des points critiques pour la modélisation des descripteurs de bas-niveau. Ainsi, notre première contribution concerne l'accélération des méthodes K-means, GMM et UMB-MAP, non seulement sur des machines indépendantes, mais également sur des clusters de machines. Afin d'atteindre une vitesse d'exécution la plus importante possible sur une machine unique, nous avons montré que les procédures d'apprentissage des dictionnaires peuvent être réécrites sous forme matricielle pouvant être accélérée efficacement grâce à des infrastructures de calcul parallèle hautement performantes telle que les multi-core CPU ou GPU. En particulier, en s'appuyant sur GPU et un paramétrage adapté, nous avons obtenu une accélération de facteur deux par rapport à une implémentation single thread. Concernant le problème lié au fait que les données ne peuvent pas être stockées dans la mémoire d'une seul ordinateur, nous avons montré que les procédures d'apprentissage des K-means et GMM pouvaient être divisées par un schéma Map-Reduce pouvant être exécuté sur des clusters Hadoop et Spark. En utilisant notre format matriciel sur ce type de clusters, une accélération de 5 à 10 fois a pu être obtenue par rapport aux librairies d'accélération de l'état de l'art.

En complément des descripteurs audio bas-niveau, des descripteurs de niveau sémantique intermédiaire tels que l'harmonie de la musique sont également très importants puisqu'ils intègrent des informations d'un niveau d'abstraction supérieur à celles obtenues à partir de la simple forme d'onde. Ainsi, notre seconde contribution consiste en la modélisation de l'information liée aux notes détectées au sein du signal musical, en utilisant des connaissances sur les propriétés de la musique. Cette contribution s'appuie sur deux niveaux de connaissance musicale : le son des notes des instruments ainsi que les statistiques de co-occurrence et de transitions entre notes. Pour le premier niveau, un dictionnaire musical constitué de notes d'instruments a été élaboré à partir du synthétiseur Midi de Logic Pro 9. Basé sur ce dictionnaire, nous avons proposé un algorithme « Positive Constraint Matching Pursuit » (PCMP) pour réaliser la décomposition de la musique. Pour le second niveau, nous avons proposé une décomposition parcimonieuse intégrant les informations de statistiques d'occurrence des notes ainsi que les probabilités de co-occurrence pour guider la sélection des atomes du dictionnaire musical et pour construire un graphe à candidats multiples pour proposer des choix alternatifs lors des sélections successives. Pour la recherche du chemin global optimal de succession des notes, les probabilités de transitions entre notes ont également été incorporées. Les expérimentations menées sur plusieurs jeux de données ont montré que nos approches permettent d'avoir des résultats supérieurs à ceux de l'état de l'art pour l'identification des notes ainsi que pour la classification de la musique en genres musicaux et en émotions.

Chapter 1: Introduction

1.1 Research context

With the development of information technology, digitalized music production exploded in the past decade. For example, in the on-line music store of iTunes, there have been over 37 million songs[1] available by 2014 and the number is still growing fast. YouTube website receives 100 hours of video upload in every minute and a large portion is music. Such a huge amount of data drives the development of effective and efficient methods for automatic music analysis and retrieval.

Public availability of music reveals two pivot tasks. One is the semantic analysis increasingly demanded by users, because human beings tend to express idea and make judgment on semantic level. Among all semantic concepts associated with music, the mood and genre are the most natural semantic information expressed by music, which can be easily perceived by audiences even without musical trainings. Modern digital music technology not only assists in music composition but also provides digitalized music knowledge such as realistic music instrument sound library for music synthesis via MIDI scripts. Thanks to this advance, we can make use of music knowledge a prior to improve mood and genre classification accuracy.

The other is to improve the efficiency of entire systems in order to handle large scale data. In this era of big data, we do not worry about the insufficient samples any more. The main focus shifts to terminating computation in an affordable duration. Thanks to the parallel computing infrastructures such as multi-core CPU, GPU and cluster of workstations, we can process massive amount of data simultaneously on multiple computing units. In our specific case, with the high-end hardware in hand, it is now possible for us to scale up our model and accelerate the entire computation procedure, which leads to superior results.

[1] http://www.apple.com/itunes/features/

1.2 Problem and Objective

Music is a type of sound that has some stable frequencies in a time period. All musical sounds have their fundamental frequency and overtones. Fundamental frequency is the lowest frequency in harmonic series. Overtones are integer multiples of the fundamental frequency. Music can be produced by several methods. For example, the sound of a piano is produced by striking strings, and the sound of a violin is produced by bowing.

Music semantic information, in particular mood and genre, are expressed in different levels, from low level (signal level) to intermediate level (note level) and high level (direct semantics). In signal level, various features like MFCCs are extracted, in which all information is kept including noises. However, music semantics concerns abstract concepts that are expressed and perceived indirectly. To bridge the semantic gap between signal level features and human cognition, the signal level features need to be transformed to reflect the abstract concepts. Bag-of-word (BoW) [Yang *et al.* 2007] model has been demonstrated effective to transform original features to histogram of potential semantic centers or "words". In addition to BoW, UBM-MAP [Campbell *et al.* 2006] based super vector method not only considers feature distribution of individual sample but also concerns background distribution as well. In signal level, the main challenge for the two approaches is scalability when encountering large data sets. For example the size of clusters for k-means and GMM should increase as the number of training sample grows. Our first objective is to accelerate k-means, GMM and UBM-MAP procedures to handle large number of training data.

Although signal level features contain complete information that can describe high-level semantics of music, their direct use is rarely very efficient even with BoW and UBM-MAP transformation. Music is indeed composed by sequential combination of notes and is generated by instruments accordingly. People understand music by perceiving the note combination sequence too. Therefore sound of notes plays an essential role in the semantic carried by music for identifying high-level concepts such as mood. If music can be effectively

decomposed into note sound of instruments, statistics on the decomposition can provide valuable information for further semantic analysis. Thanks to Logic Pro 9, a high quality music producing software from Apple Inc., a "musical dictionary" made of musical words that are related to the notes produced by various instruments is built. Our second objective is to decompose music onto the dictionary as precisely as possible and take the advantage of statistics of the decomposition to improve mood and genre classification.

1.3 Our Approach and Contributions

As discussed above, music semantic analysis requires to handling information in different levels. In signal feature level, the main challenge is to accelerate the BoW process, namely to speed up k-means, GMM and UBM-MAP training with parallel infrastructure in order to incorporate massive samples in big data sets. In middle feature level, the main challenge is to recover note information as precise as possible with music knowledge in the form of instrument sound and note statistics. Our work mainly concentrates on the two aspects and is summarized as follows.

1. To incorporate signal level information for semantic analysis, we have adopted a bag-of-word model with k-means/GMM dictionary and a UBM-MAP based super vector approach. With scaled up data set, there are more and more demands for increasing number of clusters and mixtures for k-means and GMM [Over 2011]. The dictionary learning becomes the bottleneck of the whole computation pipeline.

   Our first contribution thus focuses on accelerating k-means, GMM and UBM-MAP frameworks and it is divided into two levels which concern the acceleration on single machine and on cluster of workstations. In fact huge parallelism exists for k-means and GMM dictionary learning, for example in distance and probability calculation the computation pattern is the same for all input feature vectors. However, to simply divide work load by the number feature vectors does not guarantee maximum throughput of input data because of huge overhead for threading scheduling and data sharing, which is shown in

our experiment with comparison with the state-of-the-art libraries [Gao *et al.* 2012]. To achieve the maximum speed on single machine, we show that dictionary learning procedures can elegantly be rewritten in matrix format, for example in GMM training only two matrix multiplications and one matrix transformation are needed to re-estimate the parameters. Once converted to matrix format, the calculation can be accelerated efficiently by high performance multithreading linear algebra libraries. Especially with GPU support and careful tuning, we can even push the top speed up to two magnitudes faster than single thread implementation [Gao *et al.* 2012].

Beyond single machine, we show that for big data set which cannot fit into the memory of individual computer, the k-means and GMM training procedures can be performed on independent sub-data blocks distributedly since the computation on each sub-data block is independent and the final result is obtained by aggregating sub-results. This map-reduce [Dean 2005] computational style inspires us to employ Hadoop [White 2009, Shvachko *et al.* 2010] and Spark [Zaharia *et al.* 2010, Zaharia *et al.* 2012] cluster frameworks to even extend parallelization onto multiple computers level. Our experiments show that not only on single machine but also on Hadoop and Spark cluster our proposed method outperforms the state-of-the-art libraries in terms of speed and performance of music mood and genre classification [Gao *et al.* 2012].

2. Although signal level features with BoW model can be effective for music semantic analysis, it is still necessary to explore mid-level features like notes of music since it contains higher level of abstraction of meaning beyond physical oscillation. Mixing different tracks of instrument playing by averaging is quite easy which means the other way round is difficult. Thanks to modern music software which provides realistic instrument sound we can make use of this musical knowledge to help recover notes.

   Our second contribution thus focuses on recovering note information from music signal with musical knowledge to aid semantic analysis. This contribution relies on two levels of musical knowledge: instrument note sound and note co-occurrence/transition statistics [Gao *et al.* 2012].

On the instrument note sound level, a note dictionary is firstly built from Logic Pro 9. We choose the first 80 realistic instruments as in general MIDI level 1 set and 31 percussion instrument sets including 1860 percussions. For each instrument, we choose 60 notes form note 31 to note 90. The 60 notes span 5 octaves from low to high, covering most instruments' playing note range. The duration of each note is set to 186ms, which are 4096 samples under a sample rate of 22050Hz. This duration is long enough to hold one attack-decay-sustain-release envelope (ADSR) and leads to 5.38Hz in terms of frequency resolution, which is sufficient to discriminate adjacent note spectrum. Our musical dictionary is finally built by computing each musical word in frequency domain as the single-sided power spectrum of each note wave generated by short time Fourier transform (STFT) with Hamming window. A musical word in the dictionary is thus a 2048 dimensional vector. The number of musical words is the number of instruments multiplying the number of notes (60) plus the number of percussion instruments that is 6660 musical words in total.

With a musical dictionary in hand, any sparse solution solver can be applied to produce sparse solution in respect to note. Unfortunately classical greedy algorithms, for example orthogonal matching pursuit (OMP) [Pati *et al.* 1993], cannot directly be applied to the decomposition. Because each musical word in dictionary is a single-sided power spectrum of certain note, a positive constraint is naturally imposed onto the sparse solution. Therefore we propose a positive constraint matching pursuit (PCMP) [Gao *et al.* 2012] algorithm to solve this problem. When scrutinizing the decomposition results of PCMP within one frame, we found irregular note combinations. This is due to PCMP's over-fitting target signals without considering any compatibility of concurrent notes. To solve this problem, we propose a two stage sparse decomposition approach integrated with note statistical information. In frame level decomposition stage, note co-occurrence probabilities are embedded to guide atom selection in modified matching pursuit algorithm with the dictionary. A sparse multiple candidate graph is then constructed to provide backup choices for later selections. In the global optimal path searching stage, note transition

probabilities are incorporated together with a goodness measure of frame decomposition. Its principle is to guide the local sparse music decomposition with co-occurred notes information and decode the global optimal decomposition path with consecutive note knowledge. Due to the Gabor limit [Yao 1993], time and frequency resolution cannot be well satisfied at the same time. Thus, we emphasize the frequency resolution aspect rather than the exact time location, since correct note recognition is more important for our following classification task.

Experiments on multiple data sets show that our proposed approaches outperform the state-of-the-art in terms of accuracy and recall for note recovery and music mood/genre classification.

1.4 Organization of the Thesis

- In chapter 2, we give a literature review of the works related to content based music signal analysis especially for emotion. We start with signal level features designed for content-based music classifications and corresponding low-level feature modeling methods. We then review middle-level features especially the notes or multi F0 estimation from music signal. Finally we present state-of-art music mood classification systems.
- In chapter 3, we focus on acceleration of low-level feature modeling ie. to speed up K-means, GMM and GSV. At first, we show that k-means clustering and EM algorithm for GMM can be translated to matrix multiplication format, which can be accelerated effectively under parallel computing infrastructures. We then show that when processing big data, we can even fit matrix format into MapReduce model so that the whole computation can be parallelized onto Hadoop and Spark clusters. Next, we introduce performance tuning for GPU, multi-core CPU and cluster implementations. The quality of trained GMM is also discussed in detail. Finally we show experiments on the execution speed and quality of the learnt dictionary, compared with the state-of-the-art libraries.
- In chapter 4, we focus on mid-level feature of musical note extraction and its application on mood classification. Firstly, we present our construction of a

musical dictionary, which consists of two sets of MIDI based instrument sounds. We then introduce our modified matching pursuit algorithm to perform music decomposition subject to a natural positive constraint. Next, we verify the quality of sparse music decomposition via mood classification experiments on two data sets. To overcome the discontinuity of previous sparse decomposition, we further present our two-stage approach involving frame-level statistical integration and global-level optimal paths searching. Finally we show experimental results of the improved decomposition approach on real-world music signals.

- In chapter 5, we draw the conclusions and propose some perspectives for future research directions.

Chapter 2: Literature Review

In this chapter, we give a literature review of the works related to content based music signal analysis especially for emotion. First, we survey signal level or low level of features designed for multi-purpose signal analysis and content based music classifications. We also review the mainstream low level feature modeling methods for classification tasks i.e. bag-of-words (BoW) and universal background model based Gaussian super vector (GSV-UBM). Then we review middle level features especially the notes or multi F0 estimation from music signal. Finally, as we concentrate in this thesis on mood as a high level concept to detection in music, we survey state-of-art music mood classification systems.

2.1 Audio Features

The mood induced by music, for example happiness, is however too abstract to have simple if-else style definition. Therefore, data driven machine learning techniques turn out feasible solutions. Compared with the necessity of innovation concerning classifiers, feature extraction has even more space to delve. To develop effective features, researchers firstly turn to ourselves for help.

Speaking of humans, the very first step to understand audio information is to extract features from audio signal through the front end of our elaborate auditory system. Figure 2.1 shows the main part of cochlea. As it can be observed in Figure 2.1, there is a stiff structural element named basilar membrane inside the cochlea. The basilar membrane is the base for the sensory cells of hearing or "Stereocilia" (approximately 30.000 cells), and hence plays a crucial role in the transfer of sound oscillation to the nerve impulse to the neural networks in brain. It also differentiates frequency distribution for incoming sound waves. Depending on the input frequencies, different regions from the basilar membrane will resonate, consequently only a small subset of sensory cells is activated. From the Figure 2.1 we also note that the distribution of frequencies is more logarithmic than linear, which influences many analysis methods and feature design for example wavelet analysis and MFCC. Further listening experiments show that

although humans are experts for distinguish the frequency, loudness and temporal information of the sound, we are not sensitive to the phase of the sound wave. These biological conclusions inspire researchers to design many effective features to characterize audio signal in different aspects.

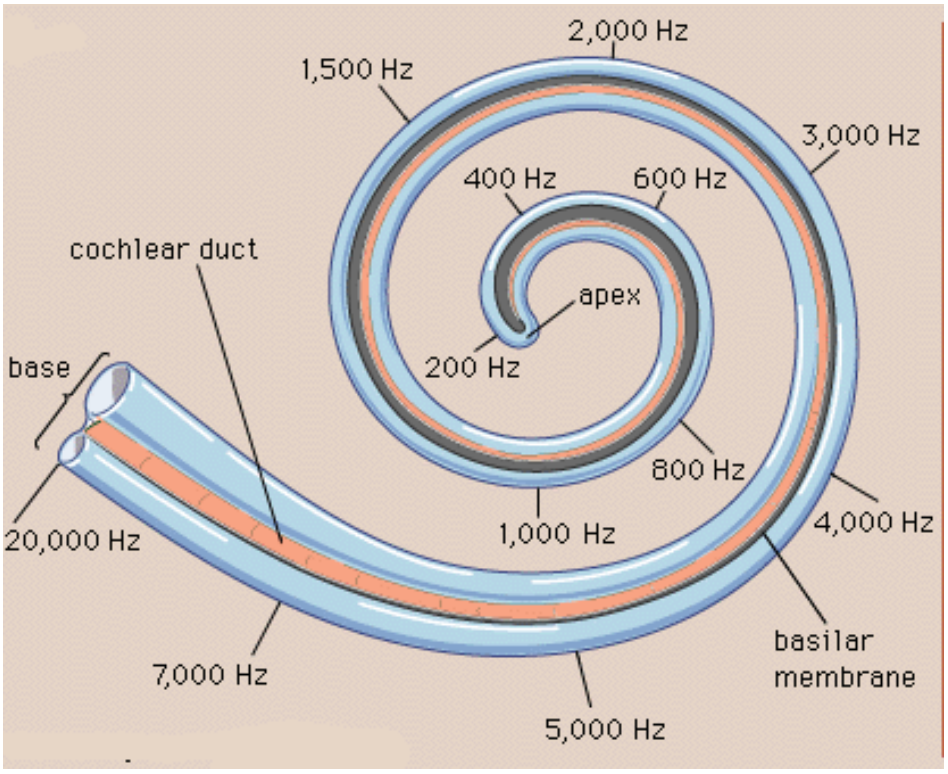


Figure 2.1: Main part of human cochlea [Pradier 2011].

Music related features are usually divided into 3 levels. Low level i.e. signal level features reflect intrinsic characteristics of the signal without conveying direct semantic information about the contents. Middle level features incorporate human knowledge of music so that somewhat semantic information is expressed through for example notes, chords, and rhythms. High level features from lyrics which is related to nature language contain direct meaning that author endow to the music. Level by level the features are developed to shorten so called semantic gap.

2.1.1 Low level feature survey

Although low level features are a little far from expressing semantics directly, they contain complete information of the signal and can be modeled to indirectly indicate semantics like mood. Low level features concerns 3 out of 4 main aspects

of the sound: frequency, energy and the temporal, since human is not sensitive to the phase distortion.

Frequency carries a large portion of information for the sound, just like the colors for images. Figure 2.1 verifies that sensing frequency is indeed the foundation for human to analyze audio signal. Therefore many effective features have been developed concerning frequency domain information.

All features in this group live in frequency or autocorrelation domain. The most popular methods are the Fourier transform and the autocorrelation. Other popular methods include the Cosine transform, Wavelet transform, and the constant Q transform. We firstly review frequency features related to the short-time Fourier transform (STFT) for computation of the spectrogram.

Spectral flux (SF). The SF is the 2-norm of the frame-to-frame spectral amplitude subtraction vector [Scheirer 1997]. It quantifies (abrupt) difference in the shape of the spectrum over consecutive frames. Signals that have slowly varying spectrum for example noise have low SF, while signals with abrupt spectral changes (e.g. note onsets) have high SF.

Lu *et al.* in [Lu *et al.* 2001] provide a slightly different definition where the SF is calculated in logarithm domain of the spectrum. Similarly to SF, the cepstrum flux is defined by Xu in [Xu 2005]. SF is widely used in audio retrieval, e.g. in speech/music discrimination [Khan *et al.* 2004, Jiang *et al.* 2005, Khan *et al.* 2006], music information retrieval [Tzanetakis 2002a, Li *et al.* 2004], and speech analysis [Tzanetakis 2005].

Spectral peaks is proposed by Wang in [Wang 2003, Wang 2006]. Spectral peaks are designed for a very compact and noise robust representation of an audio signal. Sparse set of time-frequency pairs the constellation map is firstly constructed from Fourier spectrogram with local peaks. Pairs of time-frequency points are then calculated from the constellation map. For each pair, the two frequency components, the time difference, and the time offset from the beginning of the audio signal are combined into a feature. Each piece of music is finally represented by a large number of such time-frequency pairs. Wang also proposed

an efficient search algorithm for large databases built from features described. The search system is detailed in [Wang 2003]. For spectral peak features, the music is represented in the form of spatio-temporal combinations of dominant frequencies. The advantage of the technique is that it only relies on the salient frequencies (peaks) and denies all other relatively weak spectral content. This preserves the main characteristics of the spectrum and makes the representation highly robust to noise since the spectrum of noise is usually flat thus the peak frequencies are less influenced by noise than the other signals. However, only considering the frequency with the largest energy can neglect useful information which hidden in for example second largest peaks.

Pitch is an important frequency related characteristic of sound, different from loudness, duration, and timbre. The hearing sensation of pitch is defined as "that attribute of auditory sensation in terms of which sounds may be ordered on a scale extending from low to high" [ANSI 1995]. The term pitch can refer to fundamental frequency or frequency and the perceived frequency of a signal depending on the application case.

When pitch refers to the perceived fundamental frequency of a sound, it stands for a subjective psychophysical attribute. According the auditory experiment human can differentiate about 1400 distinct tones that is the total number of perceptible pitch steps in the range of human hearing capability. Note that the total number of notes in the musical equal-tempered scale is 120 notes. Pitch is usually denoted by the fundamental frequency F0.

Fundamental frequency. The fundamental frequency is the lowest frequency of a harmonic series, which denote the base vibration. It coarsely approximates the psychoacoustic pitch. Former researchers have developed various methods to estimate fundamental frequency such as temporal autocorrelation, spectral, and cepstral methods and combinations of these techniques as well. An early survey can be found in [Hess 1983].

Pitch Histogram. The pitch histogram tends to depict pitch distributions of a signal in a compact way and has been introduced for musical genre classification

in [Tzanetakis 2002a, Tzanetakis 2002b]. In musical analysis the meaning of pitch equals to musical notes. The pitch histogram is a global representation that aggregates the pitch information or F0 from many short audio frames. Consequently, these pitch histograms provide the distribution of the musical notes in music segments.

Chromagram. The chromagram is a spectrogram that represents the spectral energy of each of the 12 pitch classes [Bartsch *et al.* 2005]. The logarithmized short-time Fourier transform is firstly performed. The frequencies are mapped periodically to the 12 pitch classes by an aggregation function since each octave is consisted of 12 notes. The result is a 12 element vector for each audio frame. A similar algorithm for the extraction of chroma vectors can be found in [Goto 2003].

The chromagram maps all frequencies into one octave. This results in a spectral compression that allows for a regulated description of harmonic information in the signal. Since the energy of the same note in different octaves is aggregated into the same slot, large harmonic series can be represented by only a few chroma values [Bartsch *et al.* 2005]. The advantage of chroma is to provide an octave-invariant (compressed) spectrogram that takes properties of musical perception into account. However, its drawback is co-occurring that chrome blurs the harmonic informat.

Pitch Profile. The pitch profile is a more accurate representation of the pitch distribution than the chroma features [Zhu *et al.* 2006]. It considers pitch mistuning introduced by mistuned instruments and is robust against noisy percussive sounds that do not have a pitch. Zhu *et al.* applyed the pitch profile in musical key detection and results show that the pitch profile outperforms traditional chroma features.

Harmonicity serves to distinguish periodic signals, for example harmonic sounds generated by instruments, from non-periodic signals like drum and noise-like sounds. Harmonics are frequencies located at integer multiples of the fundamental frequency. The harmonic spectrum shows peaks at the fundamental frequency and its integer multiples.

Harmonicity relates to the proportion of harmonic components in a signal, which is usually large in music signals. Harmonicity features can be used to distinguish musical instruments. For example harmonic instrument sounds like horns have stronger harmonic structure than percussive instrument sounds like drums. Furthermore, harmonicity can also be useful to separate environmental sound between harmonic and inharmonic sounds.

Inharmonicity measures. Most real world harmonic signals do not have a perfect harmonic structure. Inharmonicity features measure the difference between observed energy distribution along harmonics and their theoretical values which are exactly at integer multiples of the fundamental frequency.

A straight-forward cumulative measure for the deviation of the harmonics from their predicted values is introduced in [Agostini *et al.* 2001] and [Peeters 2004]. A more robust and more accurate feature is harmonicity prominence in which the energy and the bandwidth of each harmonic component are further considered in [Cai *et al.* 2006]. A variant feature is to calculate entropy of the distances of adjacent peaks in the spectrum. Perfect harmonic sounds have constant distances, while for non-harmonic sounds the distances may vary.

The concept of the "cepstrum" has been firstly introduced by Bogert *et al.* in [Bogert *et al.* 1963]. Cepstrum was originally used to detect echoes in seismic signals. Cepstral features were introduced into audio domain by [Noll 1964, Bridle *et al.* 1974, Davis *et al.* 1980] to perform speech analysis. Cepstral features presents smoothed frequency of the log magnitude spectrum. It also conveys timbral characteristics and reflects pitch information. Euclidean metric can be applied on cepstral features to measure their distances, since cepstral features are embedded in an orthogonal space. Today, cepstral features are widely used in many fields of audio retrieval some example can be found in [Lu *et al.* 2001, Xu *et al.* 2004].

Cepstral Features. Bogert *et al.* define the cepstrum as the Fourier Transform (FT) of the logarithm (log) of the magnitude (mag) of the spectrum of the original signal [Bogert *et al.* 1963]. The signal is firstly Fourier transformed. The log the

Fourier transform magnitude is then served as input of the second time Fourier transforms. The final cepstrum is the result of the second Fourier transforms. (e.g. signal→FT →log(mag)→FT →cepstrum) This sequence is the original form for the cepstral features. However, in practice the computation slightly differs from this definition. For example, the second Fourier transform is often replaced by a DCT due to its ability to decorrelate output data as shown in following classical MFCC features.

Mel-frequency cepstral coefficients (MFCCs). MFCC's most notable success is achieved in automatic speech recognition task and MFCC has evolved into one of the standard techniques in most domains of audio signal processing. Computation of MFCCs includes a conversion of the Fourier coefficients to Mel-scale [Stevens *et al.* 1937]. After conversion, the obtained vectors are logarithmized, and decorrelated by DCT in order to remove redundant information as mentioned in previous paragraph.

The components of MFCCs are usually the first 13 DCT coefficients that describe the coarse spectral shape. The first DCT coefficient represents the average power in the spectrum. The second coefficient approximates the broad shape of the spectrum and is related to the spectral centroid. The higher-order coefficients represent finer spectral details (e.g. pitch). MFCC can also be treated as an approximation of human hearing cochlea which shares the same characteristic of frequency response. In practice, the first 8-13 MFCC coefficients are sufficient to represent the shape of the spectrum. However, some applications may require more higher-order coefficients to capture pitch and tone information. For example in Chinese speech recognition up to 20 cepstral coefficients may be beneficial [Wang *et al.* 2000].

Beside classical MFCC, several variations of MFCCs have been proposed. They mainly differ in the applied psychoacoustic scale i.e. instead of using Mel-scale, variations consider the Bark[Zwicker 1961], ERB[Moore *et al.* 1990] and octave-scale [Madage *et al.* 2004]. A typical variation of MFCCs is Bark-frequency cepstral coefficients (BFCCs).

Up to now the short-time Fourier transform (STFT) serves as the fundamental building block for most frequency related features. However, STFT provides only a suboptimal tradeoff between time and frequency resolution since the fixed window locks both the frequency and time resolution. The advantage of adaptive time-frequency decompositions, like the Wavelet transform is that they provide a frequency resolution that varies with the temporal resolution.

Instead of short-time Fourier transform, Wavelet transform and related transformations for time-frequency decomposition are firstly applied to obtain coefficients based on Wavelet mother function. For example, Khan *et al.* have successfully applied the variance of Haar Wavelet coefficients over several frames to speech/music discrimination in [Khan *et al.* 2006]. We consider such features as low level features since they do not have a semantic interpretation.

Daubechies Wavelet coefficient histogram features (DWCH). DWCHs have been proposed by Li *et al.* for music genre classification in [Li *et al.* 2003]. The authors used Daubechies Wavelets to decompose the audio signal. Histograms from the Wavelet coefficients are then built for every subband. The subband histograms provide an approximation of the waveform variation in each subband. The final feature vector is composed of the first three statistical moments of each coefficient histogram together with the energy per subband. Li *et al.* also show that when combined with traditional features, DWCHs can further improve performance for music genre classification [Li *et al.* 2003]. DWCHs have been used in the fields of artist style identification, emotion detection, and similarity retrieval in [Li *et al.* 2004, Li *et al.* 2006].

Adaptive time frequency transform (ATFT). The ATFT proposed by Umapathy *et al.* in [Umapathy *et al.* 2005] is similar to the Wavelet transform. The signal is decomposed into a set of Gaussian basis functions of different scales, translations, and central frequencies. The scale parameter varies with the waveform envelope of the signal and represents for example rhythmic structures. It shows that the scale parameter contains discriminatory ability for musical genres classification.

Variable Resolutions Transform (VRT). The VRT is first derived from the classic definition of Continuous Wavelet Transform (CWT) in order to enable a variable time-frequency coverage which should fit to music signal analysis better. The consideration of specific properties of music signal finally leads to change the mother function as well and thus VRT is not a true CWT but a filter bank.

Auto-regression analysis is a standard technique in signal processing where the future signals are treated as linear combination of previous values. Linear predictor is built to estimate the value of each sample of a signal in the form of linear combination of it ancestors. Linear prediction analysis has a long tradition in audio retrieval and signal coding early examples can be found in [Rabiner *et al.* 1978, Tremain 1982].

Linear predictive coding (LPC). LPC is widely used in automatic speech recognition since it takes into account the source-filter model of speech production. The basic assumption is that vocal stimuli are modulated by filters like throat and mouth and filter can be parameterized, which means the filter parameter can be used as feature to represent speech signal [Rabiner *et al.* 1978]. Under the assumption the goal of LPC is designed to estimate basic parameters of a speech signal, such as formant frequencies and the vocal tract transfer function. LPC can be applied in other domains such as audio segmentation and general purpose audio retrieval where the LPC spectrum is used as an approximation of the spectral envelope.

As mentioned earlier, beside frequency and temporal features, loudness features that related to signal amplitude or energy is another important aspect of audio signal that human can easily perceive. Formally loudness is "that attribute of auditory sensation in terms of which sounds may be ordered on a scale extending from soft to loud" [ANSI 1995]. The auditory system incorporates a number of physiological mechanisms that influence the transformation of the incoming physical sound intensity into the sensational loudness [Zwicker *et al.* 1999].

Specific Loudness Sensation (Sone). Pampalk *et al.* propose a feature that approximates the specific loudness sensation per critical band of the human

auditory system [Pampalk *et al.* 2002]. A Bark-scaled spectrogram is firstly computed and then spectral masking and equal-loudness contours are applied. Finally, the spectrum is transformed to specific loudness sensation in sone.

Related to loudness the energy of a signal is the square of the amplitude represented by the waveform. The power of a sound is defined as the energy transmitted per unit time in second [Moore 2004]. Consequently, power is calculated as the mean-square of a signal. In many cases the root-mean-square is used in feature extraction.

Short-time energy (STE). STE describes the envelope of a signal and is extensively used in various fields of audio retrieval. STE is defined according to Zhang *et al.* as the mean energy per frame (which actually is a measure for power) [Zhang *et al.* 2001]. The same definition is used for the MPEG-7 audio power descriptor [ISO-IEC 2002].

2.1.2 Low level feature modeling

With such amount of features in hands, we can characterize music signal from so many different angles. For the tasks concerning music signal itself for example signal denoising, signal coding etc. low level features alone are sufficient. For semantic analysis, however, another layer of model has to be built to further extract semantics from low level feature since they still contain too much detail of signal's redundancy and randomness which shade semantic concepts like mood and genre.

K-means [Lloyd *et al.* 1982] and GMM [Reynolds *et al.* 1990] clustering based bag-of-words framework has demonstrated its effectiveness to model low level features and has been successfully applied to a number of multimedia classification tasks, *e.g.*, visual categorization in computer vision. The bag-of-words model is a simplifying representation originated in natural language processing and information retrieval (IR). In this domain, a text (such as a sentence or a document) is represented as the bag or multi-set of its words, disregarding grammar and even word order but keeping multiplicity. Bag-of-words method first constructs clusters of assembled low level features through k-means or GMM

clustering. It then generates histogram against k-means centers or GMM mixtures, which reflects feature distributions with respect to the "words" in bag. This procedure helps translate signal level information into meaningful audio and visual words which convey more definite information for example genre and mood for music signal. The overall BoW is shown in Figure 2.2. Low level features are firstly extracted from music signal. Low level features pass k-means or GMM dictionary. Low level features become to histogram feature vectors. Histogram feature are then passed into mood SVMs. SVMs give the final mood decision scores. From the whole process we can find that the quality of k-means and GMM is the key factor.

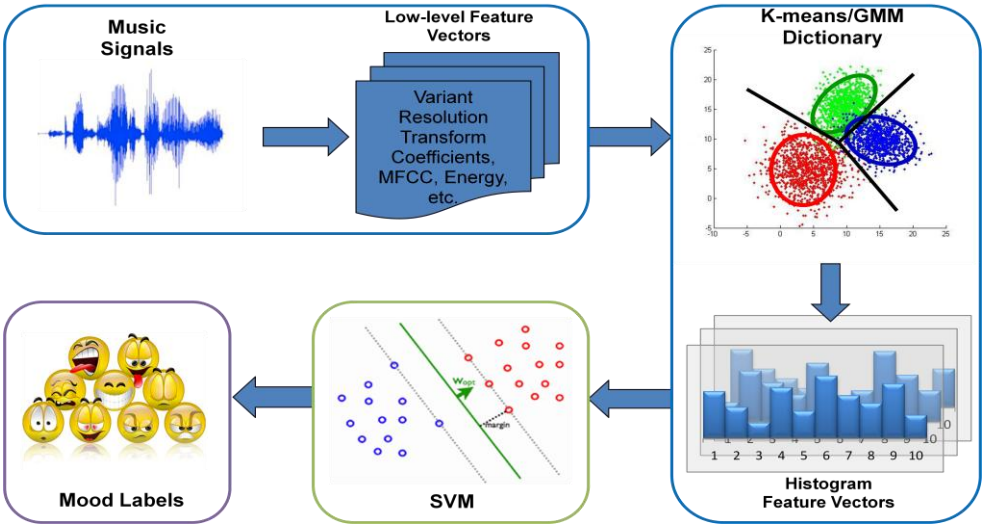


Figure 2.2: BoW based music classification framework.

2.1.2.1 K-means dictionary learning

k-means dictionary or cluster is obtained by classic Lloyd's algorithm. This algorithm aims at minimizing the objective function of

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

where $\|x_i^{(j)} - c_j\|$ is Euclid distance between a data point $x_i^{(j)}$ and the cluster centre c_j. The algorithm is composed of the following steps.

1. Initialize K points into the space.
2. Assign each object to the group that has the closest centroid.
3. Recalculate the positions of the K centroids.
4. Go to steps 2 until the centroids no longer change.

The output of k-means clustering is indeed a Voronio diagram that tiles n-dimensional space with respect to clustering centers. A two dimensional example is shown in Figure 2.3.

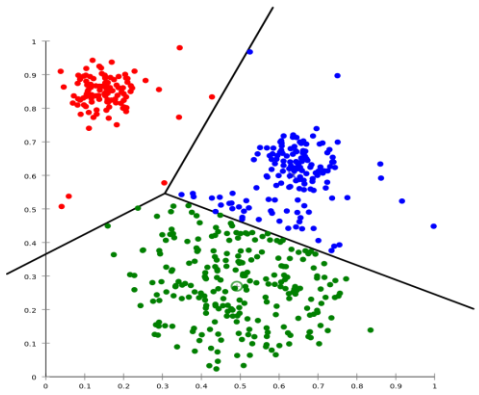


Figure 2.3: Example of 2 dimensional k-means clustering result with 3 centers

Although it can be proved that the procedure will always terminate, the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The algorithm is also significantly sensitive to the initialization of cluster centers. Many initialization methods including R-MEAN, R-SEL, SCS, KKZ and KR have been proposed to address the issue [He *et al.* 2004]. The generated k centroids are finally treated as a codebook that translates input feature vectors.

2.1.2.2 GMM dictionary learning

In addition to k-means, Gaussian mixture model (GMM) as another important stochastic dictionary has been successfully applied to encode low level feature vectors. A Gaussian mixture model is a parametric probability density function represented as a weighted sum of Gaussian component densities.

$$p(\boldsymbol{x}_i|\,\boldsymbol{\theta}) = \sum_{m=1}^{M} \alpha_m p(\boldsymbol{x}_i|m,\boldsymbol{\theta})$$

where

$$p(\boldsymbol{x}_i|m,\boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}_m|}} e^{-\frac{1}{2}(\boldsymbol{x}_i-\boldsymbol{\mu}_m)^T\boldsymbol{\Sigma}_m^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}_m)}$$

GMMs are widely used in audio signal processing domain, most notably in speaker recognition [Reynold 1992], due to their ability of representing a large class of sample distributions. The most powerful attributes of the GMM is its ability to approximate arbitrarily shaped densities functions.

GMM often performs better than k-means in two folds: as defined in equation GMM takes dimensional variances into account to shape the probability space near the centers while k-means only tiles the feature vector space according to pair-wise distance; GMM is soft whereas k-means is rigid in terms of clustering assignment of feature vectors, which means feature vectors near the border of two clusters are more reasonable handled by GMM than by k-means. Figure 2.4 shows a typical situation where GMM provides a better clustering result than k-means.

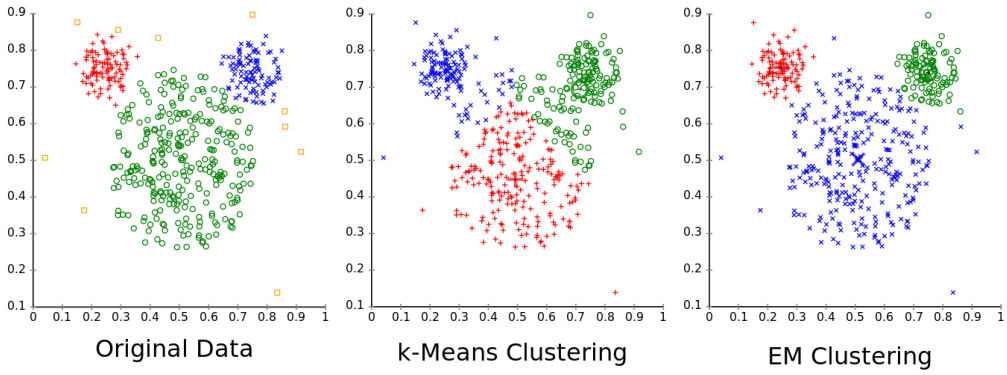


Figure 2.4: Mikey mouse data example where GMM works better than k-means.

Given training vectors and a GMM configuration, parameters of the GMM, $\boldsymbol{\theta}$ are to be estimated so as to best match the distribution of the training feature vectors. The most popular and well-established method is maximum likelihood (ML) estimation. ML parameter estimates can be obtained iteratively using expectation-maximization (EM) algorithm [Reynold *et al.* 2000]. The new parameters are re-estimated iteratively as follows

$$w_m^{new} = \frac{1}{N}\sum_{i=1}^{N} p(m|\boldsymbol{x}_i, \boldsymbol{\theta})$$

$$\boldsymbol{\mu}_m^{new} = \frac{\sum_{i=1}^{N} p(m|\boldsymbol{x}_i, \boldsymbol{\theta})\boldsymbol{x}_i}{\sum_{i=1}^{N} p(m|\boldsymbol{x}_i, \boldsymbol{\theta})}$$

$$\boldsymbol{\Sigma}_m^{new} = \frac{\sum_{i=1}^{N} p(m|\boldsymbol{x}_i, \boldsymbol{\theta})(\boldsymbol{x}_i - \boldsymbol{\mu}_m^{new})(\boldsymbol{x}_i - \boldsymbol{\mu}_m^{new})^T}{\sum_{i=1}^{N} p(m|\boldsymbol{x}_i, \boldsymbol{\theta})}$$

2.1.2.3 Gaussian super vector

In addition to BoW, Gaussian super vector (GSV) as another transformation of low level features has been successfully applied to many classification problems such as speaker verification [Compbell *et al.* 2006] and video annotation [Inoue *et al.* 2011] in particular music mood classification in Mirex evaluation[Wu 2013, Cao

et al. 2009]. In contrast to the histogram of bag-of-words, GMM super vector characterizes the mean shift between the signal given and the universal background GMM. In others words GMM super vector represents the change in contrast to the background model. The framework of GSV based classification is shown in Figure 2.5.

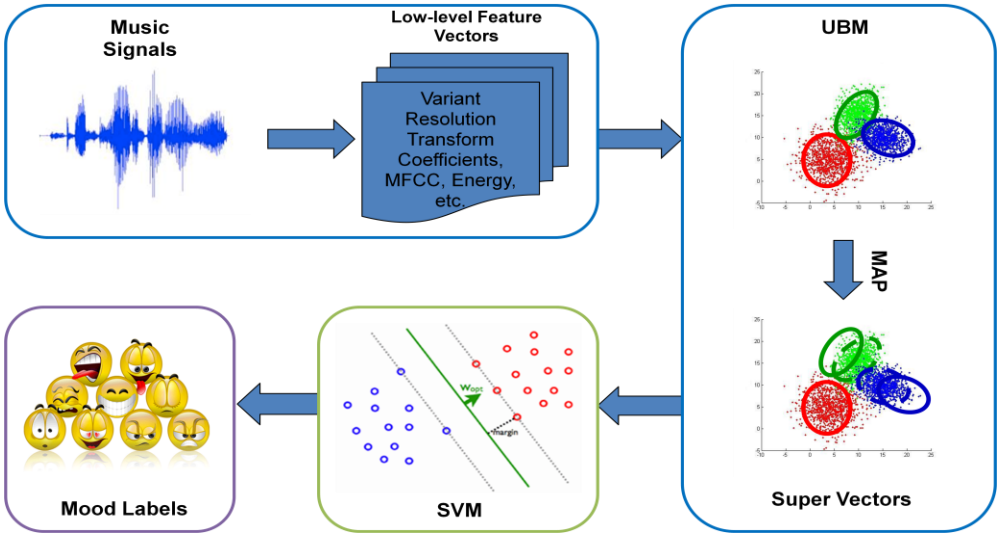


Figure 2.5: GSV based classification framework.

GSV is derived from concatenate mean vectors in the adapted GMM from a universal background model (UBM), using maximum a posteriori (MAP) estimation. The adapted GMM reflects the difference between data distribution and universal distribution. Like the EM algorithm, the MAP estimation is as follow.

$$w_m^{map} = \left[\frac{r_m o_m^{data}}{N^{data}} + (1 - r_m) w_m^{UBM}\right]\gamma$$

$$\boldsymbol{\mu}_m^{map} = r_m \boldsymbol{\mu}_m^{data} + (1 - r_m)\boldsymbol{\mu}_m^{UBM}$$

$$\boldsymbol{v}_m^{map} = r_m \boldsymbol{s}_m^{data} + (1 - r_m)(\boldsymbol{s}_m^{UBM} + \boldsymbol{\mu}_m^{UBM^2}) - \boldsymbol{\mu}_m^{map^2}$$

The first step of MAP is identical to the EM, where estimates of the sufficient statistics of the training data are computed. Unlike the second step of the EM

algorithm, the new statistics are combined with the old ones from UBM. The free parameter r_m is to balance which side should be emphasized. Figure 2.6 illustrate a simple example of the MAP estimation.

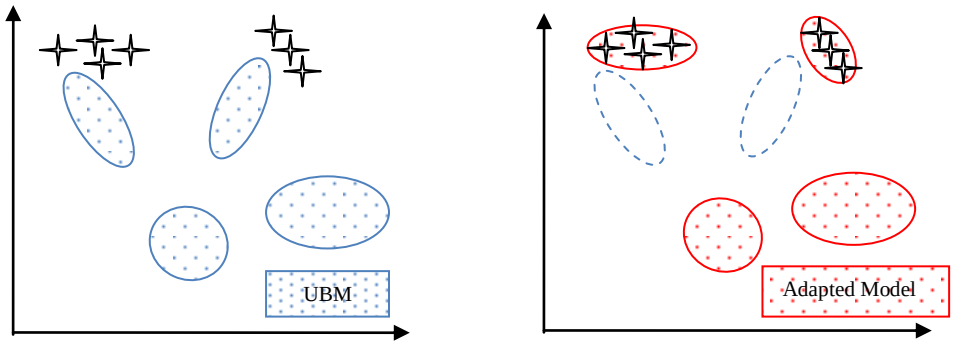


Figure 2.6: Simple example of MAP estimation. Left sub-figure denotes the universal background model with input data (stars). Right sub-figure denotes the adapted GMM with respect to new input data.

2.1.3 Mid-level features survey

Beside the signal level features, researchers also develop mid-level features which contain somewhat semantic information related to human knowledge. In this section we review mid-level features including tonality rhythm and especially note or multi-F0.

Tonality which is defined slightly different from musical system represents the sound property that can distinguish noise-like from tonal sounds [Zwicker *et al.* 1999]. Compared with tonal sounds tend to have sharp spectra, noise-like sounds have a continuous and flat spectrum. For example, instruments like violin produces regular harmonic tonal sound in contrast white noise has a flat spectrum standing for the minimum of tonality. Tonality is naturally related to the pitch strength that measures the strength of the perceived pitch.

Bandwidth. Bandwidth defined as the second-order statistic of the spectrum is usually calculated as the magnitude-weighted average of the differences between

the spectral components and the spectral centroid [Wold *et al.* 1996]. The bandwidth is thus reflects sound tonality in that tonal sounds usually have a low bandwidth since they contain single peaks in the spectrum while noise-like sounds have flat spectrum that leads to high bandwidth. However complex music sounds can still have high bandwidth meanwhile keep tonal characteristics. Therefore mere bandwidth is insufficient to distinguish tonality for the tasks that concern complex music. Besides, the bandwidth may be defined in the logarithm spectrum or the power spectrum to simulate human auditory perception [Sirinivasan *et al.* 2004]. As global bandwidth may blur the spectrum distribution, sub-bandwidth can be computed within subbands, which benefit some tasks. [Ramalingam *et al.* 2005]. In the MPEG-7 standard the measure for bandwidth is called spectral spread [ISO-IEC 2002]. Similarly to the bandwidth measures above, the MPEG-7 audio spectrum spread (ASS) is defined as the root-mean-square deviation with respect to the spectrum centroid.

Subband spectral flux (SSF). Cai *et al.* in [Cai *et al.* 2006] proposed the SSF to perform the environmental sound recognition. The feature measures the portion of prominent partials in different subbands. SSF is derived from the logarithmized short-time Fourier spectrum. For each subband the SSF is the summation of the differences between adjacent frequency slots in that subband. Low SSF represents flat subbands and high SSF indicates the subband contain variant frequency components, which reflect somewhat tonality of the signal.

Entropy. A natural measure of flatness of a spectrum is entropy. Shannonand Renyi entropy is usually computed in multiple subbands [Ramalingam *et al.* 2005]. The entropy represents the uniformity or, in the opposite angle, the chaos of the spectrum. Misra *et al.* have proposed a multi-resolution entropy feature is in [Misra *et al.* 2004, Misra *et al.* 2005]. The spectrum is split into overlapping Mel-scaled subbands and Shannon entropy is computed on these subbands. For a flat distribution in the spectrum the entropy is low meaning the spectrum is less chaotic whereas sharp peaks in a spectrum introduce high entropy which means more information exists. The entropy feature characterizes the peakiness of a

subband thus may be used for music/non-music or speech/non-speech related detection.

Rhythm is a mid-level feature that captures change patterns of timbre and energy over time span. Zwicker and Fastl have shown that the hearing sensation of rhythm depends on the temporal variation of loudness [Zwicker *et al.* 1999]. Rhythm is an important feature in music. In music it relates to the tempo of a piece of music measured in beats-per-minute (BMP). Unlike other frame based or statistic based feature, rhythm evolves over a relatively longer period. Therefore, the analysis windows of rhythm features are usually in the range of a few seconds (≈ 3-5s) [Tzanetakis 2002]. Analysis of low-frequency amplitude modulations is a common way to derive rhythmic patterns.

Pulse metric. A measure for the "rhythmicness" of sound is proposed by Scheirer and Slaney in [Scheirer *et al.* 1997]. To detect rhythmic modulations, the autocorrelation of several subbands is firstly performed and autocorrelation peaks are then identified. The autocorrelations in all subbands show peaks at similar positions means a high pulse metric, which indicates a strong rhythmic structure in the signal.

Band periodicity. The band periodicity also reflects the strength of rhythmic structures and is similar to pulse metric [Lu *et al.* 2001]. The analysis is conducted in every subbands. The maximum peak of the subband correlation function is estimated for each analysis frame. The band periodicity is then obtained by averaging the peaks in all frames. The band periodicity correlates with the rhythm content of a signal, since it captures the strength of repetitive structures over time verified by multiple analyzing windows.

Beat spectrum and beat spectrogram. The beat spectrum represents the self-similarity of a signal in different time delay which is similarly to autocorrelation. [Foote 2000, Foote *et al.* 2001]. Strong beats are implied by the peaks in the beat spectrum with a specific repetition rate. Since the peaks correspond to note onsets with high periodicity, strong beats essentially reflect high rhythm content appearing in the frame analyzed.

The beat spectrum is further concatenated for series of audio frames to form a 2 dimensional beat spectrogram. Each column of the beat spectrogram is the beat spectrum of a single frame. The beat spectrogram shows the rhythmic evolution of a signal over time. The 2D beat spectrogram visualizes how the tempo shifts over time therefore the beat spectrogram provide detailed sequential rhythm information that allows for further rhythmic structures analysis.

The beat spectrum is the foundation of onset detection and can be used to measure similarity of music in terms of rhythm. It may also be used to segment music into pieces with different rhythmical patterns, such as chorus and verse.

Cyclic beat spectrum. The CBS is a compact and robust representation of the fundamental tempo of a piece of music, which is similar to pitch classes in the chroma feature [Kurth *et al.* 2006]. Previous defined beat spectrum contains not only the fundamental tempo but also related tempos a harmonic and sub-harmonic scale to the fundamental tempo that is multiple fold tempo or half, one third etc. The cyclic beat spectrum aggregate tempos rooted in the same fundamental tempo into one tempo class. The CBS is computed from a beat spectrum. Low-pass filter is firstly used to remove timbre information irrelevant to the tempo analysis and the spectrogram is obtained from short-time Fourier transform. A novelty curve is then generated by summing the differences between adjacent spectral vectors. A bank of comb filters corresponding to particular tempos is then performed on the novelty curve. The analysis results in a beat spectrogram where peaks correspond to dominant tempos. The beat spectrum is then divided into logarithmically scaled tempo octaves. The CBS is finally obtained by aggregating the beat spectrum over all tempo classes.

Beat histogram. The beat histogram is designed as a compact global representation of the rhythm content of a piece of music [Tzanetakis *et al.* 2001]. Similarly to other rhythm features, periodicity analysis in multiple frequency bands is firstly performed. To obtain an octave-frequency decomposition the Wavelet transform is used to detect the most salient periodicities in each subband

and accumulate them into a histogram. The beat histogram thus depicts the repetition rates of main beat and sub beats together with their strength.

Each bin of the histogram corresponds to a beat period in beats-per-minute where peaks indicate the main and sub beats. The beat histogram compactly demonstrates the distribution of all occurring beat periods in a piece of music. The beat histogram can be applied to content based music classification, especially genre classification for music in different genres usually contain different beat patterns. Much useful information can be derived from the beat histogram for example a measure for the beat strength may be easily obtained from the beat histogram as in [Tzanetakis *et al.* 2002]. A derivation of the beat histogram is also proposed by Grimaldi *et al.* in [Grimaldi *et al.* 2003] which took the advantage of the discrete Wavelet packet transform (DWPT) [Mallat 1999].

Rhythm patterns. To measure music similarity and perform the retrieval Pampalk *et al.* introduced rhythm patterns in [Pampalk *et al.* 2002]. Given the spectrogram in the specific loudness sensation in sone, the amplitude modulations are extracted by Fourier transform of the critical bands over time. The extracted modulation frequencies are then weighted with respect to the fluctuation strength to simulate the human perception [Zwicker *et al.* 1999]. This results in a two-dimensional representation of acoustic versus modulation frequency.

Note (multi-F0), music is indeed sound poetry comprised of notes played by various instruments. People understand music by perceiving the note combination sequence too. Therefore sound of notes plays an important role in the semantics carried by music for identifying high-level concepts such as mood. So, if music can be effectively decomposed into note sound of playing instruments, statistics on the decomposition can provide valuable information for further music classification. However, mixing different instrument playing is trivial while decomposing is quite challenging due to the intrinsic complexity of polyphonic music.

Recovering notes from a music wave signal is usually referred to multiple F0 estimation. The approaches in literature can be roughly sorted into two categories:

parameterized like statistical model based methods and non-parameterized like non-negative matrix factorization (NMF) based methods. Parameterized approaches usually assume that multiple F0 can be described by particular models with a small number of free parameters that can be estimated from the signal. For example, in [Kmaeoka *et al.* 2007] Kameoka *et al.* propose a multi-pitch analyzer named the harmonic temporal structured clustering (HTC) method that jointly estimates pitch, intensity, onset and duration. HTC decomposes the power spectrum time series into distinct clusters such that each cluster has originated from a single source modeled by a Gaussian Mixture Model (GMM). The parameters of the source model are computed thanks to maximum a posteriori (MAP) estimation. In [Wu *et al.* 2001], Wu *et al.* extend Kameoka's work to propose a flexible harmonic temporal timbre model to decompose the spectral energy of the signal in the time-frequency domain into individual pitched notes. Each note is modeled with a 2-dimensional Gaussian kernel. Parameters of Gaussian mixtures are then estimated by expectation maximization (EM) algorithm with a global Kullback–Leibler (KL) divergence cost function.

Unlike parameterized approaches, non-parameterized methods like NMF focus on recovering pitch combinations from the signal data itself without presuming any underlying model forms. For example, NMF [Lee *et al.* 2001] based methods try to decomposes the multiple pitch spectrum matrix X into two matrices W and H [Rczynski *et al.* 2007]. W contains various harmonic patterns and H consists of activation behaviors so that $X = WH$, which is illustrated in Figure 2.7. In [Hoyer 2002], Hoyer extends the original NMF by adding a regulation term to make H sparse. Sparseness property is quite helpful especially for music note estimation, since a short period music can only contain a few notes played together, compared with all possible notes.

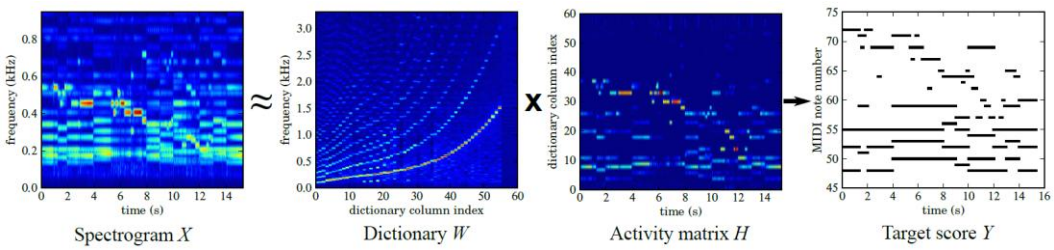


Figure 2.7: NMF for mult-F0 estimation.

NMF is such an extensible framework that it largely dominates non-parameter methods. For example, in [Zafeiriou 2006] Zafeiriou adds a linear discriminant analysis (LDA) stage to the activities extracted by NMF. In [Guan *et al.* 2011, Wang *et al.* 2004], fisher-like discriminant constraints are embedded inside the decomposition. In [Lwandowski 2012], Lewandowski proposes a supervised method with two discriminative criteria that maximize inter-class scatter and quantify the predictive potential of a given decomposition. In order to extract features that enforce the separability between pitch labels, pitch information present in time-aligned musical scores is fused in sparse NMF. In [Sakaue *et al.* 2012], Sakaue combines Bayesian inference with NMF to propose a Bayesian non-negative harmonic-temporal factorization (BNHTF). BNHTF models the harmonic and temporal structures separately with Gaussian mixture models. In [Gao *et al.* 2012], a music sparse decomposition approach is proposed using high quality MIDI dictionary. This work is a variant of sparse NMF and uses non-negative matching pursuit to solve sparse NMF. Unlike NMF that processes the entire signal, this work constructs the activity matrix H column by column. It is still worth mentioning the work in [Leveau *et al.* 2008] where Leveau *et al.* propose to learn instrument specified note atoms with a modified matching pursuit and a tracking of the played instrumental notes by searching an optimal path with respect to the reconstruction error.

Previous works in the literature have demonstrated the effectiveness of various approaches in multiple-F0 estimation, especially NMF based methods. However, under the NMF framework, the entire music spectrum series X are treated as a whole object to be reconstructed. Most of the algorithms focus on reducing the spectrum reconstruction error so as to overlook the compatibility in concurrent and consecutive notes. This batch processing style makes it hard to fuse note co-occurrence and transition information to guide note detection during the matrix factorization. Indeed, after the signal spectrum matrix is factorized, W and H are new represents of the music, which have lost signal context information for post-processing to correct possible error. Even in [Leveau *et al.* 2008], the Viterbi

algorithm is used to search the optimal path only with respect to a minimum reconstruction error and neglects the underlying note relations. Nevertheless correlation between concurrent and consecutive notes contains significant heuristics that can help to correct the decomposition error introduced by a signal level analysis.

2.2 Audio-Based Musical Mood Detection Systems Survey

2.2.1 Representation of emotions

Among all the concepts associated with music, the emotion or mood is probably the most natural semantic information expressed by music and can be easily perceived by audiences even without special music knowledge. Music is also referred to as a "language of emotion" [Pratt 1952] [Kim *et al.* 2010]. All these motivate music mood studies with high priority in both theoretical and practical perspectives.

Emotions root in highly subjective experiences therefore it is hard to find universal models to describe them. Generally speaking, there exist two directions in the psychological literatures, depending on whether emotions are considered discrete or continuous. In discrete case, categorical approaches are proposed involving finding and organizing some set of emotional descriptors (tags) based on their relevance to the corresponding music. [Hevner 1936] used 66 adjectives sorted into 8 groups. In spite of being disputed, many categorical studies inspired by Hevner demonstrate proposed tagging can be meaningful and consistent, regardless of the listener's musical background [Juslin *et al.* 2001] [Schubert *et al.* 2003]. In a sequence of music-listening studies, Zenter *et al.* [Zenter *et al.* 2008] reduced a set of 801 "general" emotional terms into a subset metric of 146 terms specific to music mood rating. Their studies, which involved rating music-specificity of words and testing words in lab and concert settings with casual and genre-aficionado listeners, revealed that the interpretation of these mood words varies between different genres of music. Another example is the BEEV (Basic English Emotional Vocabulary), which consists of 40 discrete words for automatic emotion recognition [Kim *et al.* 2008].

Another famous example is Plutchik's emotional wheel [Plutchik *et al.* 1980], shown in Figure 2.8. Plutchik created the wheel of emotions in 1980 which consisted of 8 basic emotions of joy vs. sadness, trust vs. disgust, fear vs. anger, surprise vs. anticipation and 8 advanced emotions each composed of 2 basic ones, which show in Table 2.1.

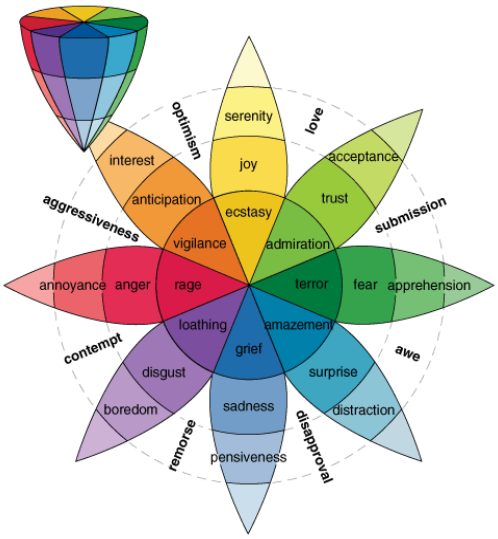


Figure 2.8: Plutchik's emotional wheel [Plutchik *et al.* 1980].

Table 2.1: Advanced emotion combination.

Human feelings (results of emotions)	Feelings	Opposite
Optimism	Anticipation + Joy	Disapproval
Love	Joy + Trust	Remorse
Submission	Trust + Fear	Contempt
Awe	Fear + Surprise	Aggression
Disapproval	Surprise + Sadness	Optimism
Remorse	Sadness + Disgust	Love
Contempt	Disgust + Anger	Submission
Aggressiveness	Anger + Anticipation	Awe

The MIREX evaluations for automatic music mood classification have categorized songs into one of five mood clusters, shown in Table 2.2. The five categories were derived by performing clustering on a co-occurrence matrix of mood labels for popular music from the All Music Guide [Hu 2008].

Table 2.2: MIREX emotion clusters.

Cluster	Mood Adjectives
Cluster 1	passionate, rousing, confident, boisterous, rowdy
Cluster 2	rollicking, cheerful, fun, sweet, amiable/good natured
Cluster 3	literate, poignant, wistful, bittersweet, autumnal, brooding
Cluster 4	humorous, silly, campy, quirky, whimsical, witty, wry
Cluster 5	aggressive, fiery, tense/anxious, intense, volatile, visceral

The continuous approach on the other hand consists in defining an N-dimensional emotional space. The most famous one is the two or three-dimensional model proposed by Russell [Russel 1980] and Schlosberg [Schlosberg 1954]. Each emotion are expressed as a point in emotional space consisting of valence (or evaluation), arousal (or activation) and potency (or power). Valence defines how positive or negative an emotion is; arousal measures the degree of excitement or involvement of the individual in the emotional state; potency accounts for the strength of the emotion. Discrete emotion categories can be transferred to the continuous by projecting in a continuous space. Figure 2.9 and Figure 2.10 shows the locations of basic emotions in the continuous space.

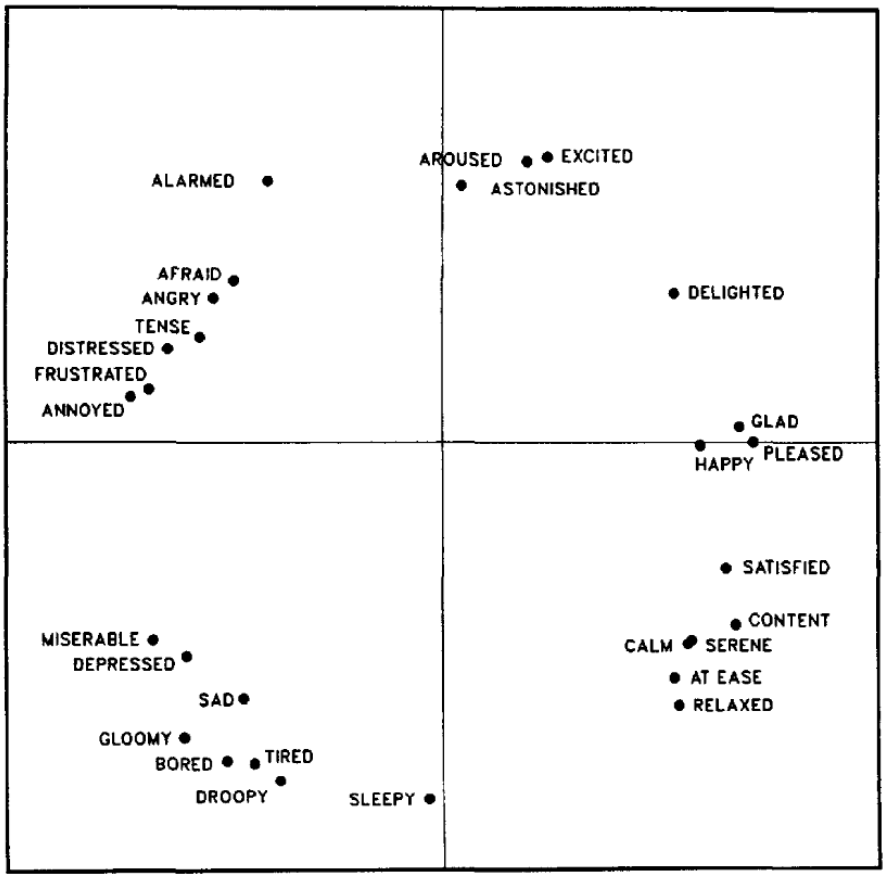


Figure 2.9: Two-dimensional emotion space and basic emotions [Russel 1980].

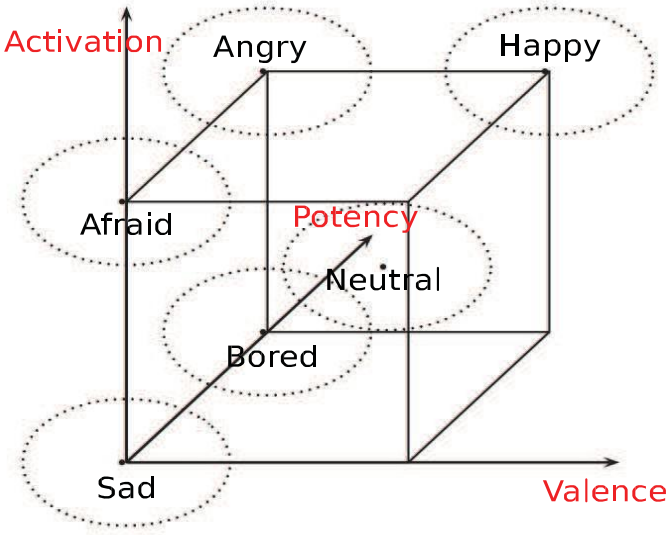


Figure 2.10: Three-dimensional emotion space and 6 basic emotions.

In following section we review the state-of-the-art of content-based emotion recognition system for both discrete and continuous emotion descriptions. The former results in a classification task and the latter in a regression task

2.2.2 Emotion classification

In one of the early works on this field, acoustic features related to timbre, rhythm, and pitch are used to train support vector machines (SVMs) to classify music into one of 13 mood categories [Li *et al.* 2003]. The system has achieved an accuracy of 45% on the data set consisting of a hand-labeled library of 499 music clips with 30-seconds each from a variety of genres including ambient, classical, fusion, and jazz.

In [Lu *et al.* 2006] mood detection and tracking was performed with a similar set of acoustic features including intensity, timbre, and rhythm. Instead of SVM Gaussian Mixture Models (GMMs) are employed as classifier for the four principal mood quadrants on the V-A classification. The system achieved an overall accuracy of 85% on a data set involving 800 classical music clips from a data set of 250 pieces with 20 seconds in duration. All music clips are labeled manually to one of the 4 quadrants.

In [Mandel *et al.* 2006] Mandel *et al.* developed an active learning system that can provide recommendations based upon any musical context defined by the user. To construct a personalized playlist, the users provide a set of seed songs to the system. The input songs represent the class of playlist desired. The system uses the initial data, combined with verification data from the user, to construct a binary SVM classifier using MFCC features. When tested on 72 distinct moods from AMG labels, the system achieved a maximum performance of 45.2%.

In [Skowronek *et al.* 2007] Skowronek *et al.* developed binary classifiers for each of 12 non-exclusive mood categories using a data set of 1059 song excerpts. Features that are used included temporal modulation, tempo and rhythm, chroma and key information, and occurrences of percussive sound events. Quadratic discriminant functions are trained for each mood, with accuracy ranging from 77% (carefree-playful) to 91% (calming-soothing) varying on the categories

In [Vaizman *et al.* 2011] the dynamic texture mixture (DTM) model is investigated for the representation of short-time audio features in an emotion classification problem. In their work they consider each audio segment to contain a static emotion in one of four categories. The dataset evaluated consists of 72 audio excerpts, each of about 30 seconds. With the DTM model, their best performing approach obtains 0.8692 in terms of area under ROC curve (AUC).

MIREX first introduced audio music mood classification as a task in 2007 [Xu *et al.* 2008]. In 2007, the highest percentage correct (61.5%) is achieved by Tzanetakis using only MFCC, and spectral shape, centroid, and rolloff features with an SVM classifier [Tzanetakis 2007]. The best system in 2008 submitted by Peeters achieved 2.2% of improvement (63.7%) by introducing a larger feature set including, MFCCs, Spectral Crest/Spectral Flatness together with a variety of chroma based features [Peeters 2008]. Before the final GMM based classification, Inertia Ratio Maximization with Feature Space Projection (IRMFSP) was first employed to perform the feature selection in which the most effective 40 features were preserved and Linear Discriminant Analysis (LDA) was also applied for further dimensionality reduction.

Starting from 2009 the Gaussian super vector based methods have dominated this evaluation. In [Cao *et al.* 2009] Cao and Li developed a system that achieved the best results in several categories, including mood classification (65.7%). Their system applied a Gaussian super vector of low-level acoustic features and followed by support vector machine as classifier. In 2013, Wu *et al.* submitted a GSV based system which topped mood classification task with 68.33% of overall accuracy. Their framework was following Cao's submission in [Cao *et al.* 2009]. In addition, two types of features are used in Wu's submissions, including visual features and acoustic features. The visual features capture characteristics of a spectrogram's texture from both local and global views. Acoustic features are used to represent global timbre characteristics. The two types of features are finally concatenated to a single long feature vector to feed into SVM classifier.

2.2.3 Emotion Regression

Parametric regression approaches have demonstrated the ability to outperform supervised classifications with similar features in music emotion prediction, presented in several recent works. In [Yang *et al.* 2008] Yang *et al.* proposed to perform regression for projecting high dimensional acoustic features to the two dimensional space to predict V-A values directly from audio, Support vector regression (SVR) [Smola *et al.* 2004] and a variety of ensemble boosting algorithms, including AdaBoost.RT [Shrestha *et al.* 2006], were employed to perform the regression. The ground-truth V-A label was collected for each of 195 music clips. 114 common features are extracted with tools of PsySound [Cabrera *et al.* 2007] and Marsyas [Tzanetakis *et al.* 1999]. PCA was applied prior to regression to reduce the data to a tractable number of dimensions. This system achieves an R^2(coefficient of determination) score of 0.58 for arousal and 0.28 for valence.

In [Han *et al.* 2009] Han *et al.* investigated a quantized representation of the V-A space and employed SVMs for classification. With inferior results of 33% accuracy in an 11-class problem, they turned to regression-based approaches. The problem was reformulated in the form of regression. They obtain a best performance of 95% classification accuracy with 11 quantized categories of GMM regression.

In [Eerola *et al.* 2009] multiple regression approaches, including Partial Least-Squares (PLS) regression were investigated. PLS is an approach that considers correlation between label dimensions. They achieved R^2 performance of 0.72, 0.85, and 0.79 for valence, activity, and tension, respectively.

In [Madsen *et al.* 2012] Madsen *et al.* propose a novel approach to develop a system that is trained on ranking data and afterwards can make V-A predictions in the testing phase. In their experiments subjects are simply asked to rate pairs of songs as to which song is higher in terms of valence and arousal. The drawback of the system is that full ranking procedure requires enumerating on all pair combinations and thus the dataset is limited in size. Their current set contains 20

songs, and therefore 190 unique pairings. With the complete training set (90% of all data) they obtain valence and arousal error of 0.13 and 0.14, respectively.

2.3 Conclusion

As surveyed in this chapter, content-based music classification, and in particular mood classification, relies on different level features extracted from music signals. Low level features preserve complete information from the original signal; however they are redundant to process which makes modeling low level features inevitable for example using BoW and GSV methods. When dealing with big data, we have to accelerate the bottle neck of the modeling which is the dictionary learning of k-mean and GMM. This leads to our first contribution for effective accelerate k-means, GMM and MAP which is developed in Chapter 3.

Regarding mid-level feature, previous work in literature hardly uses note information which is in fact the most natural semantic given by the composer. Therefore our second contribution concerns music signal decomposition into note histogram with the help of sparse representation. Two algorithms are proposed in chapter 4 to elaborate the whole process.

Chapter 3: Acceleration for Low Level Feature Modeling

3.1 Introduction

Bag-of-words (BoW) framework has demonstrated its effectiveness to model low level features and has been successfully applied to a number of multimedia classification tasks, *e.g.*, visual categorization in computer vision. In addition to BoW, MAP adaptation based GMM super vector (GSV) as another transformation of low level features has been successfully applied to many classification problems such as speaker verification [Campbell *et al.* 2006], video annotation [Inoue *et al.* 2011] and most notably in music mood classification [Wu *et al.* 2013, Cao *et al.* 2009]. In contrast to the histogram of bag-of-words, GMM super vector characterizes the mean shift between the input signal and the universal background GMM. K-means and GMM clustering, as dictionary learning procedures, lie at the heart of many audio and visual processing algorithms, in particular k-means, GMM based BoW framework and MAP based super vector approach.

In the era of big data, for example in iTunes music store, there have been over 37 million songs available by 2014 and the number is still growing fast. Google Image has being searched against 10 billion images since 2010 [Google blog 2010]. YouTube receives 100 hours of video upload in every minute. With the drastically increased data scale, the dictionary learning of k-means and GMM becomes a computational bottleneck and requires accelerations.

There are previous works in the literature employing GPU to accelerate EM for GMM training. [Kumar *et al.* 2009, Pangbom 2010, Azhari 2011, Gonina 2011, Machlica *et al.* 2011, Wu *et al.* 2012] have implemented EM algorithm on CUDA to train GMM. Wu in [Wu *et al.* 2012] also mentioned to use individual CUBLAS function to update means and variance matrix. Azhari in [Azhari 2011] implemented MFCC extraction in CUDA too. Gonina in [Gonina 2011] also provided Python interface to GPU based GMM training. Machlica in [Machlica *et al.* 2011] used cached textual memory and carefully

configured memory usage to elevate performance. In previous works user defined GPU kernel functions undertake main computation work. However, kernel functions require careful design and hardware related tuning thus is hard to translate into other languages for use in other computation platform.

In this chapter we present our approach to accelerating k-means, GMM and MAP, which can be effectively achieved on multiple parallel platforms of GPU, multi-core CPU and computer cluster such as Hadoop and Spark. The speed-up is mainly empowered by matrix-based operations. We firstly show that the three procedures can be concisely reformulated into matrix multiplications, which can be efficiently accelerated by parallel computation facilities on single machine. For example highly optimized matrix operation libraries of CUBLAS, ACML and ATLAS can be employed to speedup the calculations. Since the proposed computation structure is formulated into matrix operations, it also can be easily translated into languages with underlying BLAS support, *e.g.*, MATLAB or FORTRAN. When data is even too big to fit in single machine's memory, we show that the data can be divided into small blocks and processed block by block. When data is even bigger so as to make disk IO as bottleneck for single machine, the data has to be stored and processed distributedly in machine cluster such as Hadoop [White 2009] with Hadoop distributed file system (HDFS) [Konstantin *et al.* 2010]. Apache Hadoop is a framework for running applications on large cluster built of commodity hardware. The Hadoop framework transparently provides to applications both reliability and data motion. Hadoop implements a computational paradigm named MapReduce, where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in a cluster. In addition, it provides a distributed file system (HDFS) that stores data on the computer nodes, thereby enabling very high aggregate bandwidth across the cluster. Both MapReduce and the Hadoop distributed file system are designed so that node failures are automatically handled by the framework. Mahout is a machine learning library on Hadoop which provides k-means clustering algorithm.

Although Hadoop provides facilities to parallelize data accessing and computing, it is still designed for one pass processing in nature, which is

reading data from disks, computing and writing back results, which lacks data caching scheme. However, k-means, GMM and MAP are iterative algorithms which perform the computation on the same data for many iterations. Hadoop thus wastes disk reading overhead from the second iteration. To avoid the Hadoop's drawback, Spark framework [Zaharia 2010] has been developed by UC Berkeley AMPLab, in which Resilient Distributed Datasets (RDDs) [Zaharia 2012] is implemented to cache data into main memory when possible. RDD is a distributed memory abstraction that enables programmers to perform in-memory computations on large clusters in a fault-tolerant manner. RDDs are motivated by two types of applications that current computing frameworks handle inefficiently: iterative algorithms and interactive data mining tools. In both cases, keeping data in memory can improve performance by an order of magnitude. To achieve fault tolerance efficiently, RDDs also provide a restricted form of shared memory, based on coarse grained transformations rather than fine-grained updates to shared state. MLlib is a machine learning library on Spark that provides k-means clustering algorithm. In this chapter we show how these parallel frameworks accelerate k-means, GMM clustering as well MAP adaptation.

The contributions of this chapter are thus threefold:

- K-means, GMM, MAP reformulated into the matrix operation form
- GPU-based acceleration
- Cluster-based acceleration

The rest of the chapter is organized as follows: section 3.2 shows matrix multiplication format of k-means clustering and EM algorithm for GMM. Section 3.3 addresses Hadoop and Spark based acceleration in MapReduce model. Section 3.4 introduces performance tuning for GPU, multi-core CPU and cluster implementations. The quality of trained GMM is also discussed. Section 3.5 is dedicated to experiments where the execution speed and quality of learnt dictionary are then examined for three implementations on music genre and mood classification. Final conclusion is drawn in section 3.6.

3.2 K-means and EM Algorithm in Matrix Format

In this section k-means [Hartigan *et al.* 1979] and EM [Dempster *et al.* 1977] algorithm are shown in matrix multiplication format. The two algorithms are reformulated into matrix format in that the multiplication of matrix can be effectively accelerated via well tuned linear algebra libraries. The computational efficiency can be greatly enhanced on parallel infrastructure, compared with user defined programs. Several symbols are firstly defined: D is the number of dimensions for feature vectors, k-means cluster centers and Gaussian mixtures; N is the number of feature vectors; M is the number of k-means clusters or GMM mixtures; $\boldsymbol{X}$ is the feature matrix in which each column represents one feature vector; $\boldsymbol{C}$ is the matrix of k-means cluster centers in which each column represents one center; $\boldsymbol{V}$ represents the variance matrix of which, in contrast to a covariance matrix $\boldsymbol{\Sigma}$, $\boldsymbol{V}(i,j)$ is the variance σ^2 of dimension i for cluster or mixture j. In GMM, $\boldsymbol{w} = (\alpha_1, \alpha_2 \cdots \alpha_M)^T$ denotes the prior probabilities or weights of every mixture, $\boldsymbol{M} = [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2 \cdots \boldsymbol{\mu}_M]$ the matrix of means, $\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2 \cdots \boldsymbol{\Sigma}_M\}$ the set of covariance matrices. Other symbols are defined each time they are used.

3.2.1 K-means in matrix format

K-means clustering iteration contains two steps: cluster decision and cluster center updating. The first is to find the nearest center to which each feature vector is to be clustered. The second is to re-estimate cluster centers according to the nearest relationship.

To measure the closeness between data set $\boldsymbol{X}$ and centers $\boldsymbol{C}$, squared Euclid distance is commonly used, defined as

$$\begin{aligned} \boldsymbol{D}_s(\boldsymbol{x}_i, \boldsymbol{c}_j) &= (\boldsymbol{x}_i - \boldsymbol{c}_j)^T (\boldsymbol{x}_i - \boldsymbol{c}_j) \\ &= \boldsymbol{c}_j^T \boldsymbol{c}_j - 2\boldsymbol{x}_i^T \boldsymbol{c}_j + \boldsymbol{x}_i^T \boldsymbol{x}_i \end{aligned} \tag{3.1}$$

In matrix format, squared distance matrix can be written as

$$\begin{aligned} \boldsymbol{D}_s &= \mathbf{1} \cdot \boldsymbol{c}_s^T - 2\boldsymbol{X}^T\boldsymbol{C} + \boldsymbol{x}_s \cdot \mathbf{1}^T \\ &= [\mathbf{1}, \boldsymbol{X}^T, \boldsymbol{x}_s] \cdot \begin{bmatrix} \boldsymbol{c}_s^T \\ -2\boldsymbol{C} \\ \mathbf{1}^T \end{bmatrix} \end{aligned} \tag{3.2}$$

where $\boldsymbol{x}_s = (\boldsymbol{x}_1^T\boldsymbol{x}_1, \boldsymbol{x}_2^T\boldsymbol{x}_2 \cdots \boldsymbol{x}_N^T\boldsymbol{x}_N)^T$ is the vector of squared length of feature vectors and $\boldsymbol{c}_s = (\boldsymbol{c}_1^T\boldsymbol{c}_1, \boldsymbol{c}_2^T\boldsymbol{c}_2 \cdots \boldsymbol{c}_N^T\boldsymbol{c}_N)^T$ is the vector of squared length of centers.

For the center updating step the new center matrix can be written as $\boldsymbol{C}^{new} = \boldsymbol{X} \cdot \boldsymbol{O}$, where occupation matrix $\boldsymbol{O}$ is defined as

$$\boldsymbol{O}(i,j) = \begin{cases} \dfrac{1}{R_j}, & \text{if } \boldsymbol{x}_i'\text{s nearest center is } \boldsymbol{c}_j \\ 0, & \text{otherwise} \end{cases}$$

where R_j is the number of associated feature vectors to center j. Variance matrix can be computed similarly as $\boldsymbol{V}^{new} = \boldsymbol{X_s} \cdot \boldsymbol{O} - \boldsymbol{C}_s^{new}$, where $\boldsymbol{X}_s(i,j) = \boldsymbol{X}(i,j)^2$ and $\boldsymbol{C}_s^{new}(i,j) = \boldsymbol{C}^{new}(i,j)^2$.

3.2.2 EM for GMM in matrix format

In GMM, the probability density of a feature vector $\boldsymbol{x}_i$ is defined as

$$p(\boldsymbol{x}_i|\,\boldsymbol{\theta}) = \sum_{m=1}^{M} \alpha_m p(\boldsymbol{x}_i|m, \boldsymbol{\theta}) \tag{3.3}$$

where $\boldsymbol{\theta} = \{\boldsymbol{w}, \boldsymbol{M}, \boldsymbol{\Sigma}\}$ is the GMM parameter set and $p(\boldsymbol{x}_i|m, \boldsymbol{\theta})$ is the probability of a feature vector $\boldsymbol{x}_i$ under a single Gaussian mixture m, defined as

$$p(\boldsymbol{x}_i|m, \boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}_m|}} e^{-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_m)^T\boldsymbol{\Sigma}_m^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_m)} \tag{3.4}$$

Like k-means, EM iteration for GMM training includes two main steps: probability calculation and parameter updating. The first step decides relations

between feature vectors and Gaussian mixtures in terms of likelihood; the second step updates GMM parameters with the feature matrix according to how likely each mixture can generate the observed feature vectors.

In the first step $p(\boldsymbol{x}_i|m,\boldsymbol{\theta})$ are computed. To avoid floating point numeric overflow, the natural logarithm of $p(\boldsymbol{x}_i|m,\boldsymbol{\theta})$ is preserved during calculation

$$ln\big(p(\boldsymbol{x}_i|m,\boldsymbol{\theta})\big) = -\frac{1}{2}(g_m + \boldsymbol{x}_i^T\boldsymbol{\Sigma}_m^{-1}\boldsymbol{x}_i - 2\boldsymbol{x}_i^T\boldsymbol{\Sigma}_m^{-1}\boldsymbol{\mu}_m) \tag{3.5}$$

where $g_m = D\cdot ln(2\pi) + ln\big(\sigma_{m,1}^2\cdot\sigma_{m,2}^2\cdot\cdots\cdot\sigma_{m,D}^2\big) + \boldsymbol{\mu}_m^T\boldsymbol{\Sigma}_m^{-1}\boldsymbol{\mu}_m$. In practice covariance matrix $\boldsymbol{\Sigma}_m$ is often treated as diagonal, in order to simplify computation. Let $\boldsymbol{V} = [\boldsymbol{diag}(\boldsymbol{\Sigma}_1), \boldsymbol{diag}(\boldsymbol{\Sigma}_2)\cdots\boldsymbol{diag}(\boldsymbol{\Sigma}_M)]$ denote the GMM variance matrix; Let $\boldsymbol{P}$ denote the corresponding logarithmic probability matrix in which $\boldsymbol{P}(i,j) = ln\big(p(\boldsymbol{x}_i|j,\boldsymbol{\theta})\big)$, then $\boldsymbol{P}$ can be written as

$$\begin{aligned}\boldsymbol{P} &= -\frac{1}{2}(\boldsymbol{1}\cdot\boldsymbol{g}^T - 2\boldsymbol{X}^T\boldsymbol{M}_v + \boldsymbol{X}_s^T\boldsymbol{V}_r)\\ &= -\frac{1}{2}[\boldsymbol{1}, \boldsymbol{X}^T, \boldsymbol{X}_s^T]\cdot\begin{bmatrix}\boldsymbol{g}^T\\ -2\boldsymbol{M}_v\\ \boldsymbol{V}_r\end{bmatrix}\end{aligned} \tag{3.6}$$

where $\boldsymbol{g} = (g_1, g_2\cdots g_M)^T$, $\boldsymbol{V}_r$ is the reciprocal matrix of $\boldsymbol{V}$ in which $\boldsymbol{V}_r(i,j) = 1/\boldsymbol{V}(i,j)$ and $\boldsymbol{M}_v$ is $\boldsymbol{V}_r$-weighted mean matrix in which $\boldsymbol{M}_v(i,j) = \boldsymbol{M}(i,j)\cdot\boldsymbol{V}_r(i,j)$. Then $p(j|\boldsymbol{x}_i,\boldsymbol{\theta})$ is computed according to Bayes' rule

$$p(m|\boldsymbol{x}_i,\boldsymbol{\theta}) = \frac{w_m p(\boldsymbol{x}_i|m,\boldsymbol{\theta})}{\sum_{j=1}^{M} w_j p(\boldsymbol{x}_i|j,\boldsymbol{\theta})} \tag{3.7}$$

Posterior probability is firstly computed in logarithmic scale, that is $\ln\big(p(m|\boldsymbol{x}_i,\boldsymbol{\theta})\big) = \ln(\alpha_m) + \boldsymbol{P}(i,m) - \ln\,(p(\boldsymbol{x}_i|\,\boldsymbol{\theta}))$, and then converted into linear scale when needed. $\ln(p(\boldsymbol{x}_i|\,\boldsymbol{\theta}))$ are also aggregated directly in logarithm. When posterior probability is ready EM updating procedure can be performed to re-estimate GMM parameters as follows:

$$w_m^{new} = \frac{1}{N}\sum_{i=1}^{N} p(m|\boldsymbol{x}_i, \boldsymbol{\theta}) \tag{3.8}$$

$$\boldsymbol{\mu}_m^{new} = \frac{\sum_{i=1}^{N} p(m|\boldsymbol{x}_i, \boldsymbol{\theta})\boldsymbol{x}_i}{\sum_{i=1}^{N} p(m|\boldsymbol{x}_i, \boldsymbol{\theta})} \tag{3.9}$$

$$\boldsymbol{\Sigma}_m^{new} = \frac{\sum_{i=1}^{N} p(m|\boldsymbol{x}_i, \boldsymbol{\theta})(\boldsymbol{x}_i - \boldsymbol{\mu}_m^{new})(\boldsymbol{x}_i - \boldsymbol{\mu}_m^{new})^T}{\sum_{i=1}^{N} p(m|\boldsymbol{x}_i, \boldsymbol{\theta})} \tag{3.10}$$

Let $\boldsymbol{P}_p$ denote the posterior probability matrix, in which $\boldsymbol{P}_p(i,j) = p(j|\boldsymbol{x}_i, \boldsymbol{\theta})$ and let $\boldsymbol{O}$ denote the occupation matrix for all mixtures, in which

$$\boldsymbol{O}(i,j) = \frac{p(j|\boldsymbol{x}_i, \boldsymbol{\theta})}{\sum_{i=1}^{N} p(j|\boldsymbol{x}_i, \boldsymbol{\theta})} \tag{3.11}$$

The EM updating formulas can then be written as

$$\boldsymbol{w}^{\boldsymbol{T}} = \frac{1}{N}\mathbf{1}^T \cdot \boldsymbol{P}_p \tag{3.12}$$

$$\boldsymbol{M} = \boldsymbol{X} \cdot \boldsymbol{O} \tag{3.13}$$

$$\boldsymbol{V} = \boldsymbol{X}_s \cdot \boldsymbol{O} - \boldsymbol{M}_s \tag{3.14}$$

where $\boldsymbol{X}_s$ is the squared data matrix in which $\boldsymbol{X}_s(i,j) = \boldsymbol{X}(i,j)^2$ and $\boldsymbol{M}_s$ is the squared mean matrix of $\boldsymbol{M}$ just updated.

3.2.3 MAP in Matrix Format

To obtain the GMM super vector, UBM should be adapted via MAP as follows

$$w_m^{map} = \left[\frac{r_m o_m^{data}}{N^{data}} + (1 - r_m) w_m^{UBM}\right]\gamma \tag{3.15}$$

$$\boldsymbol{\mu}_m^{map} = r_m \boldsymbol{\mu}_m^{data} + (1 - r_m)\boldsymbol{\mu}_m^{UBM} \tag{3.16}$$

$$\boldsymbol{v}_m^{map} = r_m \boldsymbol{s}_m^{data} + (1 - r_m)(\boldsymbol{s}_m^{UBM} + \boldsymbol{\mu}_m^{UBM^2}) - \boldsymbol{\mu}_m^{map^2} \tag{3.17}$$

where

$$o_m^{data} = \sum_{i=1}^{N^{data}} p(m|\boldsymbol{x}_i, \boldsymbol{\theta}^{UBM}) \tag{3.18}$$

$$\boldsymbol{\mu}_m^{data} = \frac{1}{o_m^{data}} \sum_{i=1}^{N^{data}} p(m|\boldsymbol{x}_i, \boldsymbol{\theta}^{UBM})\, \boldsymbol{x}_i \tag{3.19}$$

$$\boldsymbol{s}_m^{data} = \frac{1}{o_m^{data}} \sum_{i=1}^{N^{data}} p(m|\boldsymbol{x}_i, \boldsymbol{\theta}^{UBM})\, \boldsymbol{x}_i{}^2 \tag{3.20}$$

$\boldsymbol{x}_i$ is the input feature vector to adapt on. $\boldsymbol{\theta}^{UBM} = \{w_m^{UBM}, \boldsymbol{\mu}_m^{UBM}, \boldsymbol{v}_m^{UBM}\}$ represents the universal background GMM, $r_m = o_m^{data}/(o_m^{data} + o^{\rho})$ denotes the relevant coefficient for mixture m, in which o^{ρ} is a free parameter of absolute relevant occupancy. $\boldsymbol{x}_i{}^2$ denotes $\mathrm{diag}(\boldsymbol{x}_i \boldsymbol{x}_i{}^T)$ the same as column vector of $\boldsymbol{X}_s$. According to the formula MAP for GMM formula can be computed in the same way as EM algorithm *i.e.*, in matrix multiplication format. The final GMM super vector for input data is then obtained by concatenating adapted mean vector weighted by corresponding standard derivation and mixture weight.

3.3 MapReduce Acceleration

It is quite straightforward for k-means and GMM in their matrix format to take the advantages of parallel computing power on an individual machine. However, when data scale gets such a size to which one computer cannot afford, we have to seek help from computer clusters. This section shows our matrix multiplication method can perfectly fit into computer cluster scenario, in particular MapReduce model so that the Hadoop and Spark framework can soundly be employed to further accelerate the computing.

3.3.1 MapReduce structure

MapReduce [Dean *et al.* 2008] is a programming model for processing large data sets with a parallel, distributed algorithm on a cluster. It starts to prevail

with the Apache Hadoop project. A MapReduce program is composed of a Map() and a Reduce() procedure. The Map() procedure performs dividing, filtering and sorting data blocks to conduct the distributed computation. The Reduce() procedure performs a collecting operation that aggregate sub results and transform them into the final one. The "MapReduce System" consists of distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing redundancy and fault tolerance.

3.3.2 Hadoop architecture

Hadoop is designed to efficiently process large volumes of information by connecting many affordable commodity computers together to work in parallel.

3.3.2.1 Hadoop distributed file system (HDFS)

A big bottleneck for parallel computing is data access bandwidth, especially across computing nodes. To achieve high data accessibility Hadoop Distributed File System (HDFS) has been developed. HDFS has a master/slave architecture as shown in Figure 3.1. An HDFS cluster consists of a single Namenode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of Datanodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of Datanodes. Data blocks are also replicated several times across Datanodes. The replication makes data block more locally available to computing nodes and more robust against possible disk failure. The Namenode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to Datanodes. The Datanodes are responsible for serving read and write requests from the file system’s clients. The Datanodes also perform block creation, deletion, and replication upon instruction from the Namenode. Hadoop tie these smaller and more reasonably priced machines together into a single cost-effective storage cluster.

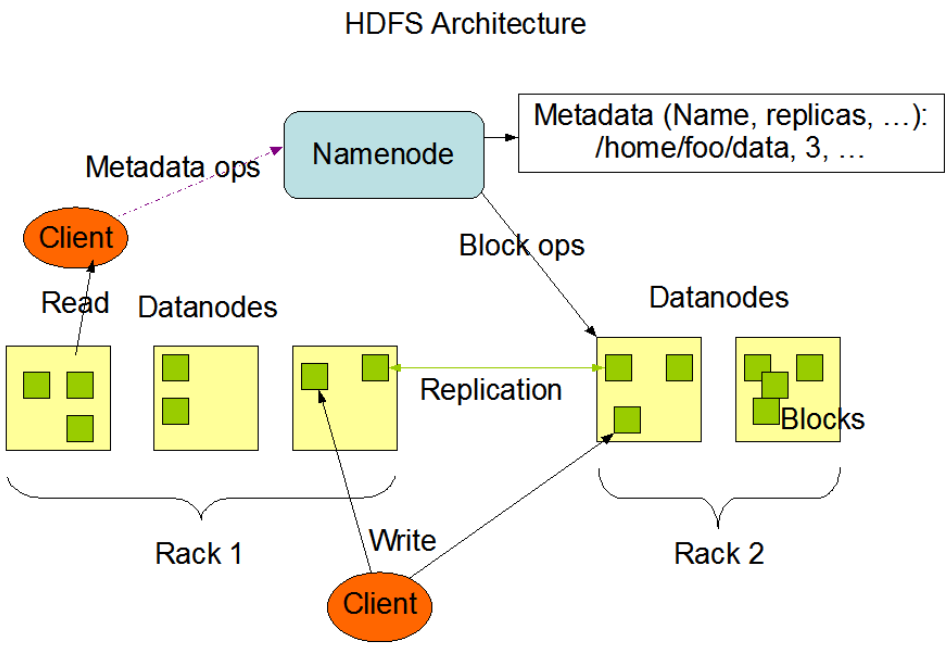


Figure 3.1: HDFS Architecture [Borthakur 2007]. Namenode manage data information of datanodes. Datanodes store duplicated blocks of data. Clients try to access blocks of data locally and remotely.

3.3.2.2 Hadoop MapReduce

Hadoop MapReduce architecture is shown in Figure 3.2. In Hadoop MapReduce, records are processed in isolation by tasks called Mappers. In each computing node there can exist multiple Mappers. Mappers are arranged as near data as possible so that inter-node communication is minimal. The output from the Mappers is then sorted according to the key and brought together into a second set of tasks called Reducers, where results from different mappers are merged together. Mapping and reducing tasks run on nodes where individual records of data are already present. Separate nodes in a Hadoop cluster still communicate with one another but only when necessary. However, in contrast to more conventional distributed systems where application developers explicitly marshal byte streams from node to node over sockets or through MPI buffers, communication in Hadoop is performed implicitly. Pieces of data are tagged with key names which indicate Hadoop to send related bits of information to a common destination node. Hadoop

internally manages all of the data transfer and cluster topology issues, from data blocks to machine and machine to rack.

To tolerate system fault, Hadoop isolate tasks in both Mappers and Reducers by wrapping up all necessary resources. By restricting the communication between nodes, Hadoop makes the distributed system much more reliable. Individual node failures can be worked around by rearrange tasks on other machines. Since user-level tasks do not communicate explicitly with one another, no messages need to be exchanged by user programs, nor do nodes need to roll back to pre-arranged checkpoints to partially restart the computation. The other workers continue to operate as though nothing went wrong, leaving the underlying Hadoop layer to partially restart the failed program.

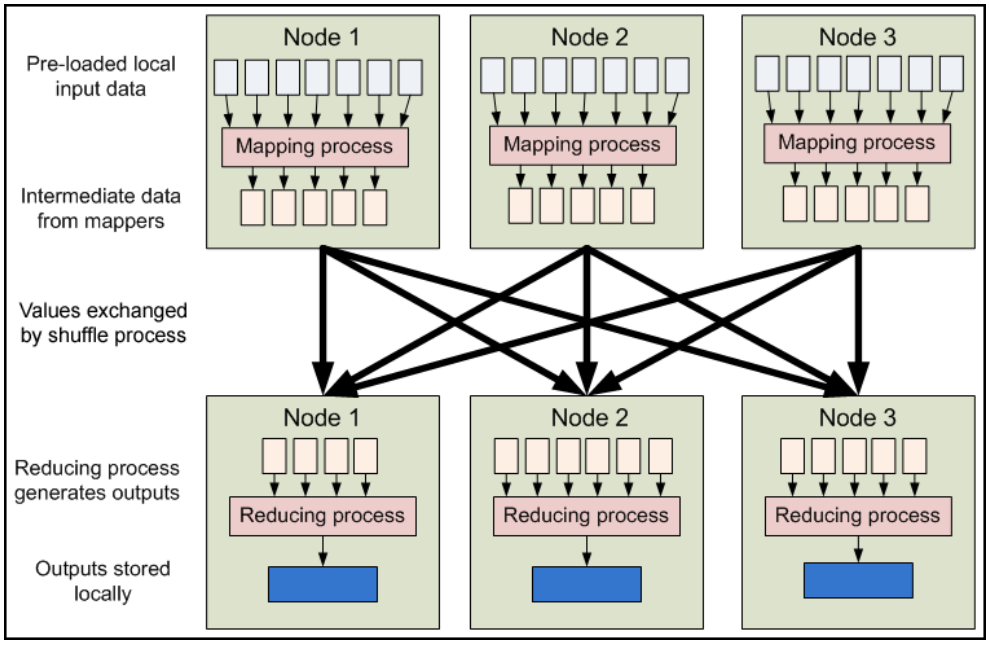


Figure 3.2: Hadoop MapReduce Architecture [Yahoo 2014]. Datablocks are read by the mapper and is processed. The intermediate results are then shuffled and provided to the reducer. The reducers generate the final combined result.

3.3.3 MapReduce k-means in matrix format

Since Hadoop provides infrastructure for MapReduce computing model, we show in this section how matrix format computation can fit into it. For large

data set, data feature matrix $\boldsymbol{X}$ is divided into blocks $\boldsymbol{X} = [\boldsymbol{X}^{blk_1}, \boldsymbol{X}^{blk_2} \cdots \boldsymbol{X}^{blk_B}]$ which in fact has been implicitly done by HDFS. What we need to do is just writing a data format to inform Hadoop how much data is needed for a Mapper.

3.3.3.1 Map phase

In each iteration, the Map phase consists of computing sub-results of weights, center and variance on $\boldsymbol{X}^{blk}$ as follow. $\boldsymbol{D}_s^{blk}$ is firstly computed as before with one matrix multiplication. For each feature block, calculate occupation matrix $\boldsymbol{O}^{blk}$ as

$$\boldsymbol{O}^{blk}(i,j) = \begin{cases} 1, & \text{if } \boldsymbol{x}_i' \text{s nearest center is } \boldsymbol{c}_j \\ 0, & \text{otherwise} \end{cases} \tag{3.21}$$

For each feature block, weight feature vector $\boldsymbol{w}^{blk}$ in which w_j indicates the number of related feature vector for center j, center matrix $\boldsymbol{C}^{blk}$ and variance matrix $\boldsymbol{V}^{blk}$ are computed by one matrix multiplication as

$$\begin{bmatrix} \boldsymbol{w}^{(blk)T} \\ \boldsymbol{C}^{blk} \\ \boldsymbol{V}^{blk} \end{bmatrix} = \begin{bmatrix} \boldsymbol{1}^T \\ \boldsymbol{X}^{blk} \\ \boldsymbol{X}_s^{blk} \end{bmatrix} \cdot \boldsymbol{O}^{blk} \tag{3.22}$$

3.3.3.2 Reduce phase

In each iteration, the Reduce phase is comprised of accumulating and averaging block-wise weight, center and variance output from B Mappers. The final re-estimated centers are computed as follow.

$$\boldsymbol{w}^{acc} = \sum_{b=1}^{B} \boldsymbol{w}_b^{blk} \tag{3.23}$$

$$\boldsymbol{C}^{acc} = \sum_{b=1}^{B} \boldsymbol{C}_b^{blk} \tag{3.24}$$

$$\boldsymbol{V}^{acc} = \sum_{b=1}^{B} \boldsymbol{V}_b^{blk} \tag{3.25}$$

In the end of iteration the updated cluster center j is calculated as

$$\boldsymbol{w}^{new} = \boldsymbol{w}^{acc} / \sum w_j^{acc} \tag{3.26}$$

$$\boldsymbol{c}_j^{new} = \boldsymbol{c}_j^{acc} / w_j \tag{3.27}$$

$$\boldsymbol{\sigma}_j^{2(new)} = \boldsymbol{\sigma}_j^{2(acc)} / w_j - \boldsymbol{c}_j^{2(new)} \tag{3.28}$$

where $\boldsymbol{c}_j^{2(new)} = diag(\boldsymbol{c}_j \boldsymbol{c}_j^T)$. Note that $\boldsymbol{V}^{blk}$ does not affect cluster center updating; therefore it is only computed in the last iteration. The matrix version k-means clustering main steps are summarized as follows.

- For each iteration
  - Map($\boldsymbol{X}^{blk}$) = {
    - Calculate $\boldsymbol{D}_s^{blk}$ with one matrix multiplication
    - Calculate $\boldsymbol{O}^{blk}$ from $\boldsymbol{D}_s^{blk}$
    - Calculate $\boldsymbol{C}^{blk}$ and $\boldsymbol{V}^{blk}$ with one matrix multiplication

    }
  - Reduce() = {
    - Aggregate $\boldsymbol{w}^{blk}$ $\boldsymbol{C}^{blk}$ and $\boldsymbol{V}^{blk}$ to update new $\boldsymbol{C}$

    }

3.3.4 MapReduce EM in matrix format

3.3.4.1Map phase

As performed in MapReduce k-means For each feature block, weight vector, mean matrix and variance matrix are computed in Mapper as follow. $\boldsymbol{P}^{blk}$ is firstly computed by one matrix multiplication as in section II. $\boldsymbol{P}_p^{blk}$ is then calculated accordingly. Sub result for $\boldsymbol{X}^{blk}$ is computed as

$$\begin{bmatrix} \boldsymbol{w}^{(blk)T} \\ \boldsymbol{M}^{blk} \\ \boldsymbol{V}^{blk} \end{bmatrix} = \begin{bmatrix} \boldsymbol{1}^T \\ \boldsymbol{X}^{blk} \\ \boldsymbol{X}_s^{blk} \end{bmatrix} \cdot \boldsymbol{P}_p^{blk} \tag{3.29}$$

3.3.4.2 Reduce phase

In each iteration, the overall accumulated weight vector, mean matrix and variance matrix for B feature blocks are computed in the phase of reduce as

$$\boldsymbol{w}^{acc} = \sum_{b=1}^{B} \boldsymbol{w}_b^{blk} \tag{3.30}$$

$$\boldsymbol{M}^{acc} = \sum_{b=1}^{B} \boldsymbol{M}_b^{blk} \tag{3.31}$$

$$\boldsymbol{V}^{acc} = \sum_{b=1}^{B} \boldsymbol{V}_b^{blk} \tag{3.32}$$

The updated GMM parameters are finally obtained as

$$\boldsymbol{w}^{new} = \boldsymbol{w}^{acc} / \sum w_j^{acc} \tag{3.33}$$

$$\boldsymbol{\mu}_j^{new} = \boldsymbol{\mu}_j^{acc} / w_j^{acc} \tag{3.34}$$

$$\boldsymbol{\sigma}_j^{2(new)} = \boldsymbol{\sigma}_j^{2(acc)} / w_j^{acc} - \boldsymbol{\mu}_j^{2(new)} \tag{3.35}$$

where $\boldsymbol{\mu}_j^{2(new)} = diag(\boldsymbol{\mu}_j \boldsymbol{\mu}_j^T)$. The matrix version main steps are summarized as follow.

- For each iteration
  - Map($\boldsymbol{X}^{blk}$) = {
    - Calculate $\boldsymbol{P}^{blk}$ with one matrix multiplication
    - Calculate $\boldsymbol{P}_p^{blk}$ from $\boldsymbol{P}^{blk}$
    - Calculate $\boldsymbol{w}^{(blk)T}$, $\boldsymbol{M}^{blk}$ and $\boldsymbol{V}^{blk}$ with one matrix multiplication
  - Reduce() = {
    - Aggregate $\boldsymbol{w}^{(blk)T}$, $\boldsymbol{M}^{blk}$ and $\boldsymbol{V}^{blk}$ to update new GMM parameters

    }

3.3.5 MapReduce in Spark

Although Hadoop provides efficient parallel mechanism, it introduces extra overhead of disk IO at each iteration. Hadoop supports neither data caching nor shared variable. Therefore for iterative algorithm like k-means and GMM,

Hadoop has to reload model and training data from disk for every Mapper at each iteration. When data is big whereas computation is relatively inexpensive, Hadoop distributed disk IO can even be a bottleneck for entire algorithms. To avoid this extra time consumption we switched to Spark the framework.

Similar to Hadoop, Spark also supports the MapReduce parallel model and HDFS accessing. The difference is that Spark introduces a data abstraction named resilient distributed datasets (RDD). An RDD is a read-only collection of objects partitioned across a set of machines that can be rebuilt if a partition is lost. RDDs can be explicitly cached in memory across machines and reused in multiple MapReduce parallel operations. RDDs achieve fault tolerance through a notion of lineage: if a partition of an RDD is lost, the RDD has enough information to rebuild just that partition. When an entire assigned RDD cannot hold into the memory of a node, RDD is then automatically and efficiently serialized onto its local disk and de-serialized back when necessary. Spark also supports shared variables which are copied to referenced Mappers and Reducers without loading from disk. For constant data, Spark even provides broadcast variables to ensure every node to receive only one copy and thereby avoids the underlying duplication for each Mapper or Reducer.

3.4 Performance Tuning

The formula in previous sections represents the theoretically ideal situation. However, to achieve the best performance, there are still several parameters that need to be tuned according to the underlying hardware configurations.

3.4.1 Multi-core CPU

For single machine based test run, when distance matrix $\boldsymbol{D}_s$, or probability matrix $\boldsymbol{P}$ is too large to fit into the memory, serialized MapReduce version of k-means or GMM must be adopted. In such a situation data blocks are processed one by one and finally aggregated. On machines equipped with AMD muti-core CPU, ACML is used for matrix multiplication to maximize the computing speed. On machine with Intel CPU, ATLAS is tuned and employed instead. To take full advantage of multi-core CPU, calculation of $\boldsymbol{O}^{blk}$ and $\boldsymbol{P}_p^{blk}$

is implemented with OpenMP[1] which takes charge of dispatching for-loop into multiple threads. However, spawning as many threads as possible is not the optimal way of acceleration. Because too many threads increase the overhead of thread scheduling and enhance cache faults or even page faults when a large number of threads access data. For example when we run the following code to calculate feature vectors' square length, the fastest speed is achieved where OpenMP schedule batch size is around 10240, which is shown in Figure 3.3.

```
#pragma omp parallel for default(shared) private(i) schedule(guided,10240)
  for (int i = 0; i < pData->nVecCurBlock; i++) {
    float * pVecSqr = pData->pVecSqrCurBlock + (long long)i*pData->nDim;
    float * pVec    = pData->pVecCurBlock + (long long)i*pData->nDim;
    for (int j = 0; j < pData->nDim; j++) {
      pVecSqr[j] = pVec[j]*pVec[j];
    }
  }
```

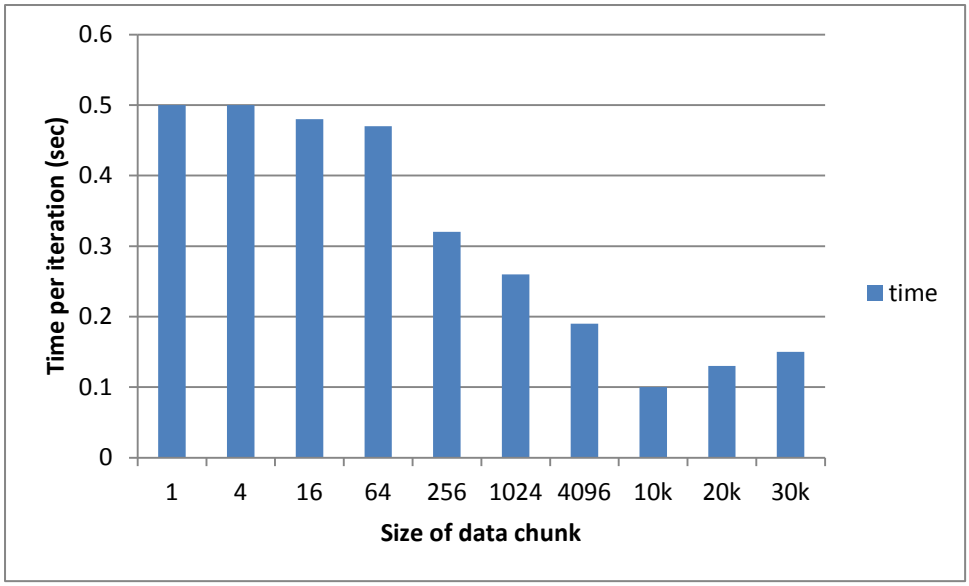


Figure 3.3: OpenMP execution time vs. batch size. 10k of data achieve the peak performance.

[1] http://openmp.org/wp/

3.4.2 Nvidia GPU

Similarly, on GPU version, the matrix multiplications are implemented with CUBLAS and calculation of $\boldsymbol{O}^{blk}$ and $\boldsymbol{P}_p^{blk}$ are implemented with custom CUDA kernel functions. To achieve the peak performance, matrix multiplication and kernel functions need to be further tuned.

CUDA computing architecture and memory access pattern have to be well understood in order to push GPU into top speed. CUDA adopts single instruction multiple threads (SIMT) architecture. Threads are firstly divided into grids and then grouped into block. The threads are index up to 3 dimensions. 32 threads within one block form one warp which is the unit of the GPU thread management. When GPUs are idle, the thread manager tries to arrange one or half warp of threads to execute at same time with the identical instructions. The CUDA parallel architecture is shown in Figure 3.4. Therefore dividing problem into threads block whose dimension is multiple of warp will execute faster, otherwise some threads will be wasted when computing the corner cases of problems.

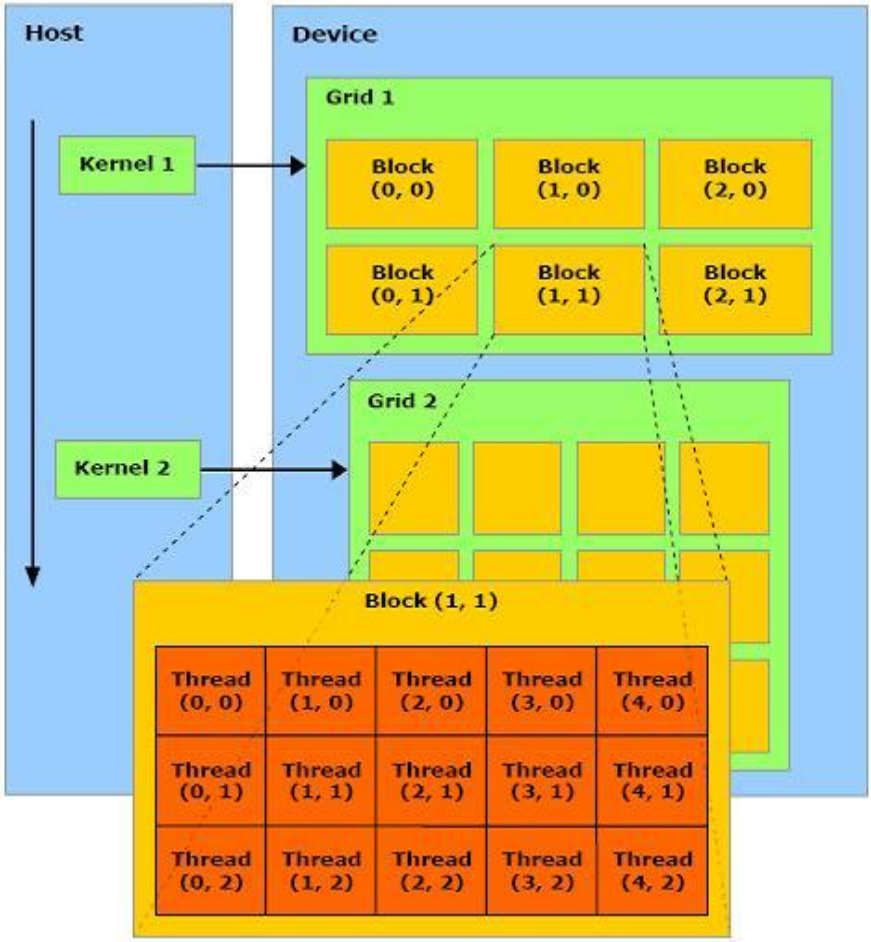


Figure 3.4: CUDA thread parallel architecture [Nvidia 2014]. Workloads are divided into 2D threads configuration.

Beside of thread architecture, memory access is another important aspect that affects parallel execution significantly. Memory in CUDA can roughly be divided into local (or shared) and global parts, as shown in Figure 3.5. Local memory is much faster, however, of too limited size. For example GTX285 has only 16KB. Local memory is shared and only accessible within the thread block. Local memory may also cause bank conflicts. As illustrated in Figure 3.6. To achieve high memory bandwidth for concurrent accesses, shared memory is divided into equally sized memory modules (banks) that can be accessed simultaneously. Therefore, any memory load or store of *n* addresses that spans *n* distinct memory banks can be serviced simultaneously, yielding an effective bandwidth that is *n* times as high as the bandwidth of a single bank. However, if multiple addresses of a memory request map to the same memory bank, the accesses are serialized. The hardware splits a memory request that has bank conflicts into as many separate conflict-free requests as

necessary, decreasing the effective bandwidth by a factor equal to the number of separate memory requests. The one exception here is when multiple threads in a warp address the same shared memory location, resulting in a broadcast. Despite these limitations, local memory should be used whenever shared data exists within thread block in order to boost memory throughput. Compared with local memory, global memory is larger, up to several gigabytes, yet slower. For our tasks, most memory access happens in global domain and for computation ability less than 2.0 the global memory does not have cache (GTX285 has computation ability of 1.3). Therefore memory access pattern affects computation throughput tremendously. In many scenarios memory access becomes the computational bottleneck.

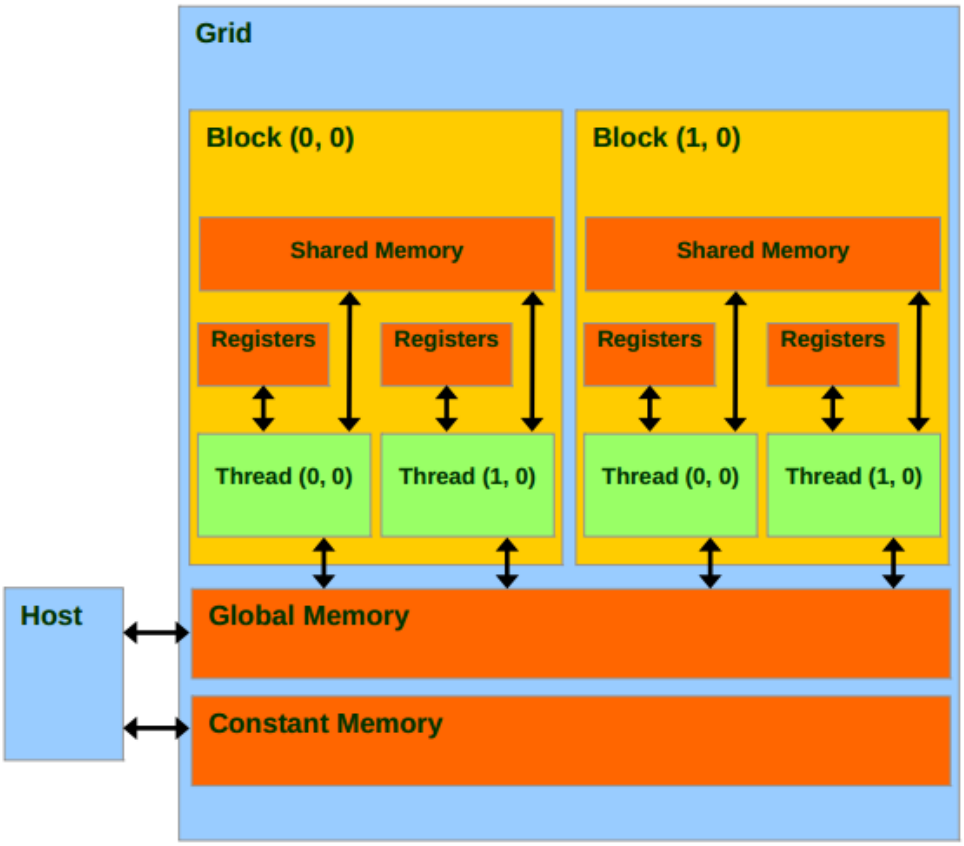


Figure 3.5: CUDA memory hierarchies [Nvidia 2014]. Global memory can be accessed by all threads. Shared memory is shared within thread block.

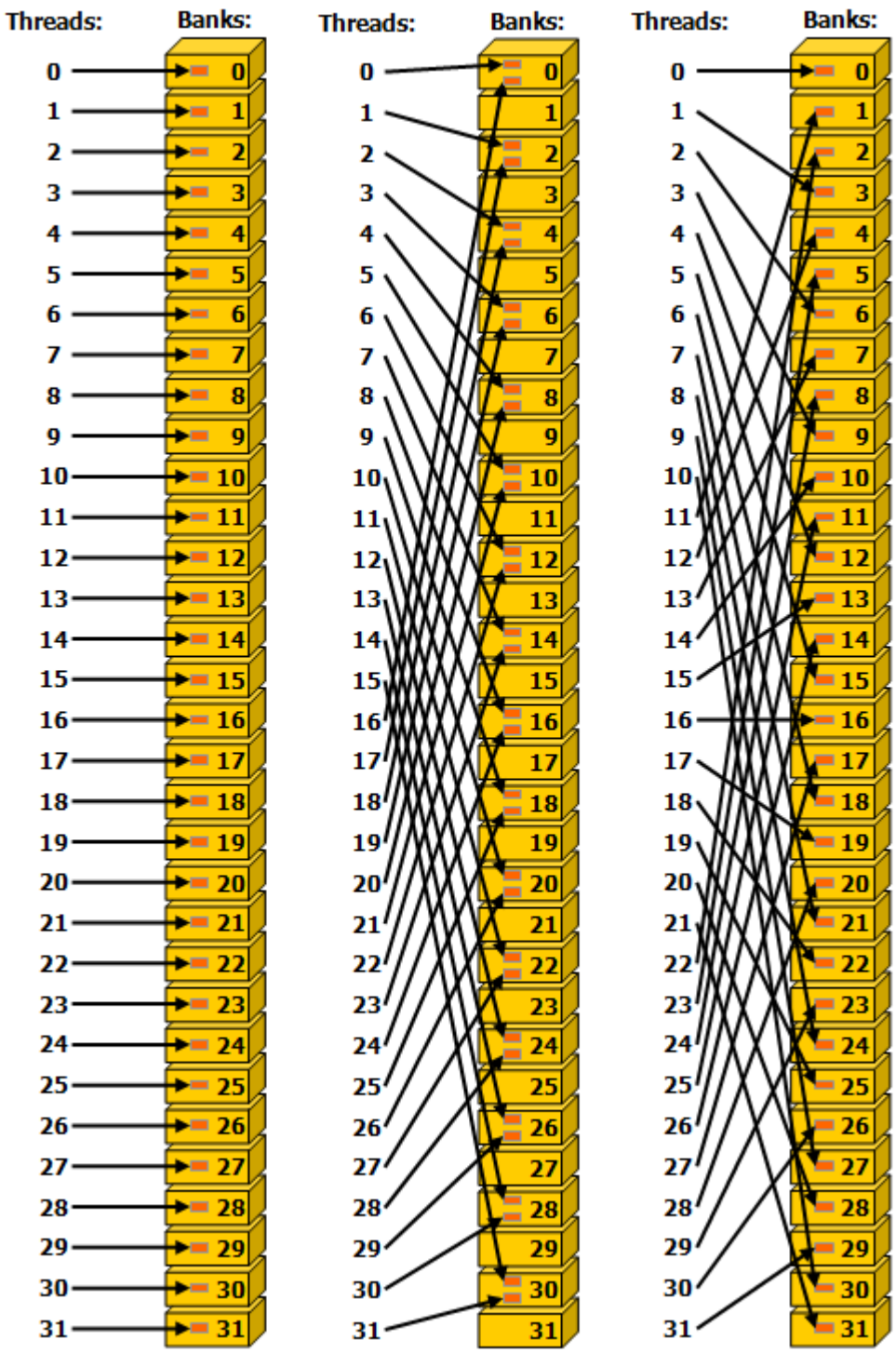
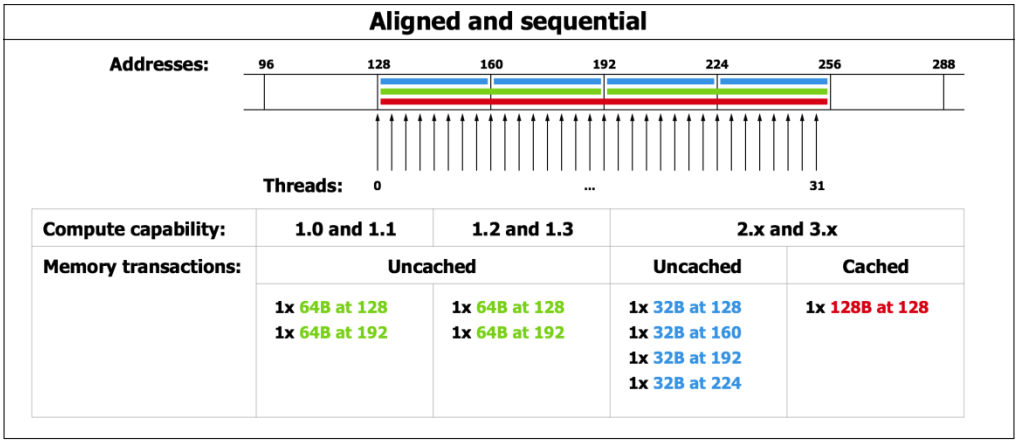
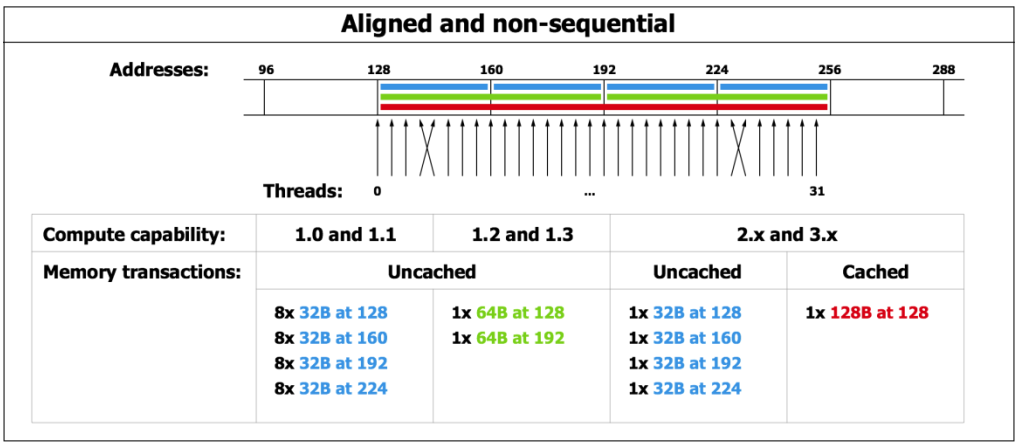


Figure 3.6: Local memory banks with/without conflicts [Nvidia 2014]. The three typical cases that will not generate bank conflicts.

In CUDA, global memory access is divided into memory transactions which can load up to 128 bytes at one time. Figure 3.7 illustrate three typical global memory access cases and corresponding memory transactions needed. From memory access scheme and SIMT architecture shown in Figure 3.4 and 3.7 we

can find that aligned continuous, namely coalesced, memory access is the most efficient way to access global memory, which prefer to neighbor threads in a block access neighbor data. Although threads block can be configured in 2D or 3D, the memory neighborhood is still confined in linear due to insufficient amount of cache.





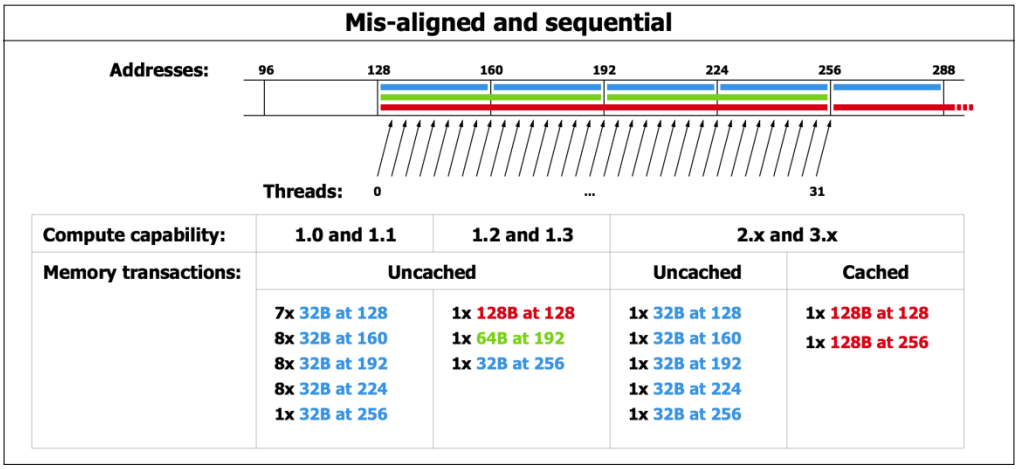


Figure 3.7: Global memory access patterns [Nvidia 2014].

For example, to compute occupation matrix $\boldsymbol{O}$ from matrix $\boldsymbol{D}_s$ in k-means algorithm, we have tested calculation performance on $\boldsymbol{D}_s$ in both row major order and column major order. In the case of row major order, adjacent threads read global memory address with big gaps whereas in column major order adjacent threads read adjacent memory. Figure 3.8 shows the time consumption of two memory accessing orders, the brown bar indicates column major order to read $\boldsymbol{D}_s$ while the bar in cyan denote row major order one. From Figure 3.8 we can find that same kernel function runs more than 7 times faster on $\boldsymbol{D}_s$.in column major order.

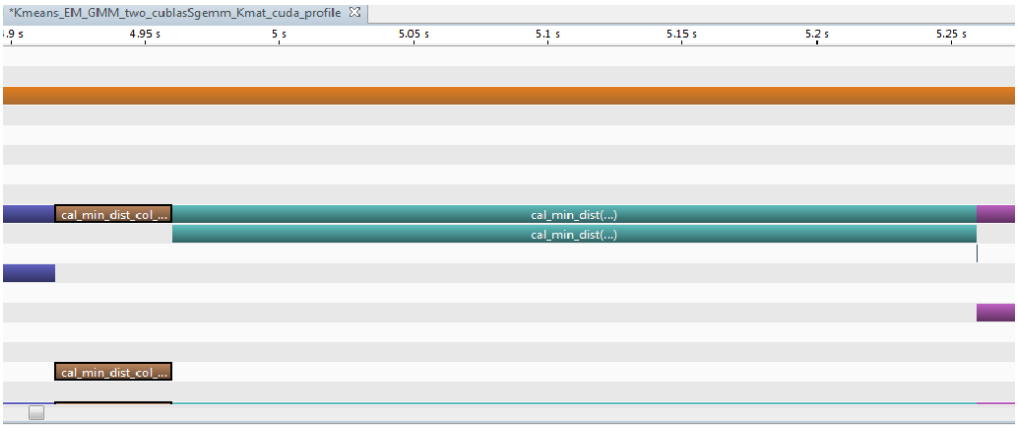


Figure 3.8: Time consumption comparison between column and row major order, the brown bar indicates column major order time consumption while the bar in cyan denotes row major order time consumption.

In former case the computation task is parallelized on each vector so that the speed gain is obtained from coalesced memory access. Another example is to compute posterior probability matrix $\boldsymbol{P}_p$ from $\boldsymbol{P}$in EM algorithm for GMM. The time consumption of 16x16 thread block dimension is shown in cyan bar in Figure 3.9 and the time consumption of 1x256 is shown in cyan bar in Figure 3.10. From the two figures we can find that thread block dimension of 1x256 is 6.9 time faster than of 16x16 (245.3ms down to 35.8ms), which lead overall 30% efficiency improvement. The acceleration is essentially due to coalesced memory access. Indeed, although taking exponential for each matrix element is a 2D parallelable operation, the matrix data is actually stored linearly in global memory. Therefore 16x16 configuration causes non-consecutive global memory access whereas 1x256 does.

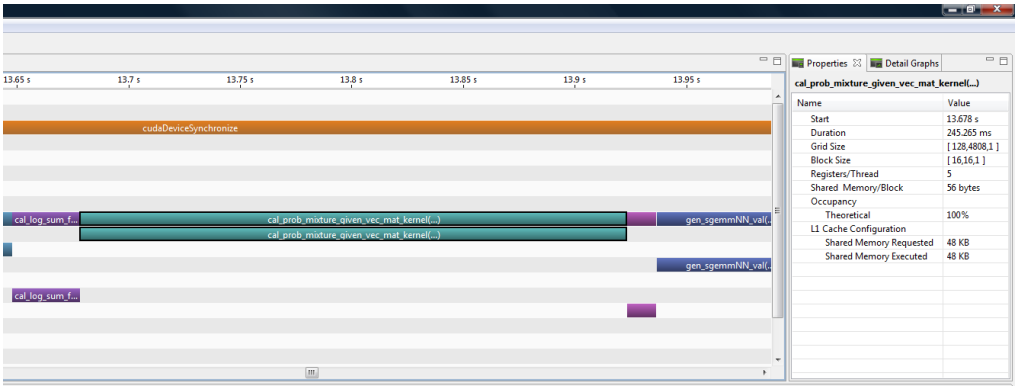


Figure 3.9: Time consumption of 16x16 thread block dimension (345.3ms in cyan bar).

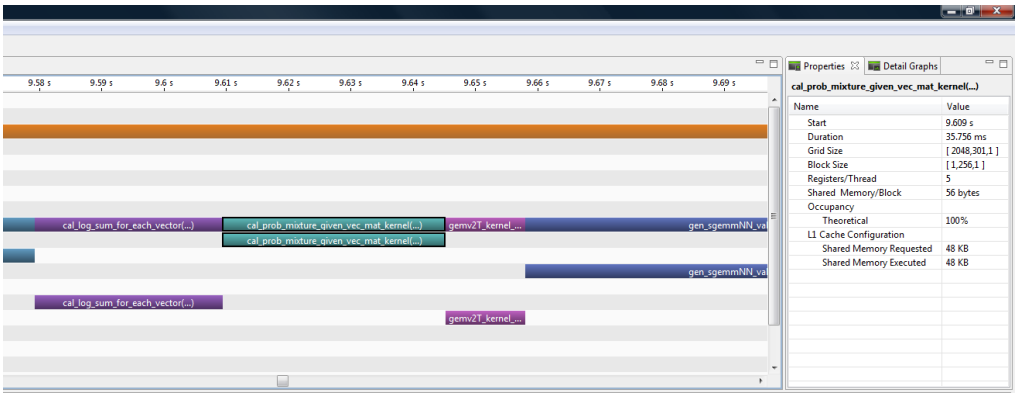


Figure 3.10: Time consumption of 1x256 thread block dimension (35.8ms in cyan bar).

In calling CUBLAS SDK functions, using non-transposed matrix function call whenever possible can improve the execution speed, because transpose operation is expensive and from NVIDIA's profiler we can find when cublasSgemm() is performed on matrix which needs to transpose the multiplication is actually performed by a CUDA kernel function, which is much slower. Figure 3.11 shows time consumption of matrix multiplication with transpose and Figure 3.12 shows time consumption without matrix transpose. From the two figures we observe 2.7 times speed-up (116.8ms down to 43.0ms) when calling cublasSgemm() with non-transposed arguments.

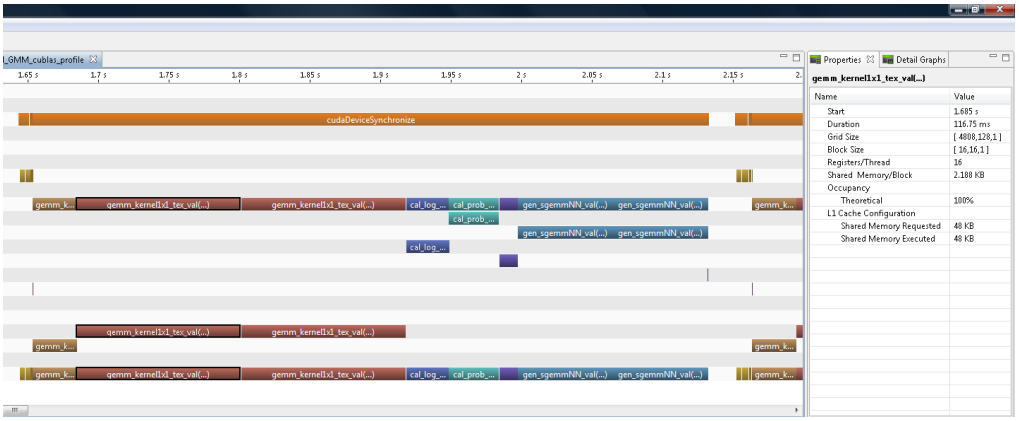


Figure 3.11: Time consumption of CUBLAS sgemm with matrix transpose (116.8ms in the first brown bar).

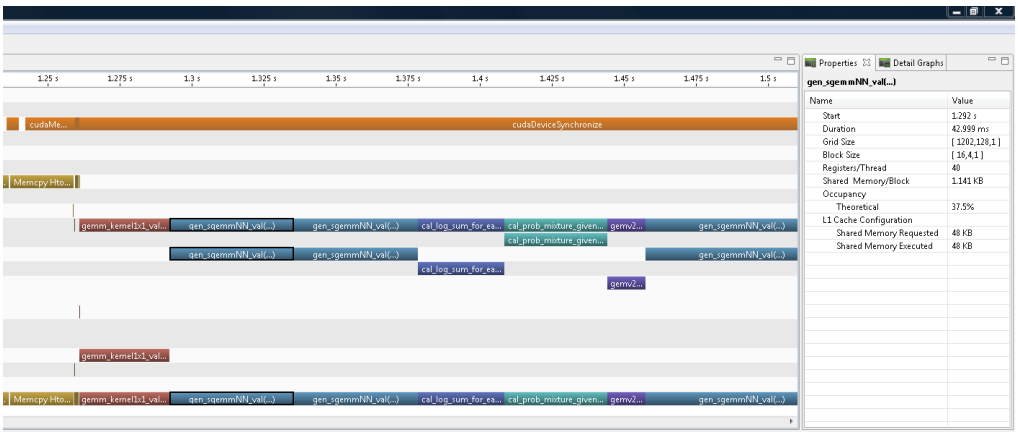


Figure 3.12: Time consumption of CUBLAS sgemm without matrix transpose (43.0ms in the first blue bar).

Another tuning point is to merge matrix operations whenever possible, like multiplication, can save overhead caused by function calling and repeated memory access. For example when block mean and variance matrix updating is combined into one matrix multiplication 10% execution time is saved.

The final aspect where acceleration can be gained is to upgrade to high performance graphic card with advanced GPU. Table 3.1 show our fine tuned implementation executes on 4 types of GPUs.

Table 3.1: Fine tuned K-mean and GMM execution time on different GPUs.

#Mixture: 2048 #Vector: 2.7M Feature: 39Dim MFCC	K-means (sec/iteration)	GMM (sec/iteration)
NVIDIA GeForce 9400 Mem:256 MB CUDA core:16	169.3	232.9
NVIDIA GeForce GT220 Mem:1G CUDA core:48	46.9	56.5
NVIDIA GeForce GTX 285 Mem:1G CUDA core:240	9.1	10.9
NVIDIA GeForce GTX 580 Mem:2G CUDA core:512	4.3	5.2

3.4.3 Hadoop and Spark cluster

On a cluster backed with HDFS, in order to achieve maximum computation throughput, the input data size is set to its largest value provided that the corresponding distance matrix or probability matrix can fit into memory. Since the data is stored in blocks in HDFS, configuring blocks size equal to input size can further avoid extra disk IO overhead. Figure 3.13 shows how the data split size influence the execution time. From Figure 3.13 we can find that our matrix based method prefers large block data which maximizes the parallel ability of each computing node. We also test the impact of number of cores to the execution time, which is illustrated in Figure 3.14 and 3.15. To leverage high performance linear algebra library such as ACML and ATLAS, computational

expensive procedures are all implemented with C language which are called via Java Native Interface (JNI) in both Hadoop and Spark. Because ACML and ATLAS are multi-threaded, limiting the number of parallel tasks on a single node is necessary to avoid over-competition for CPU time. In high-end servers with for example 32 cores, it is faster to run 8 32-threaded tasks than 8 4-threaded tasks.

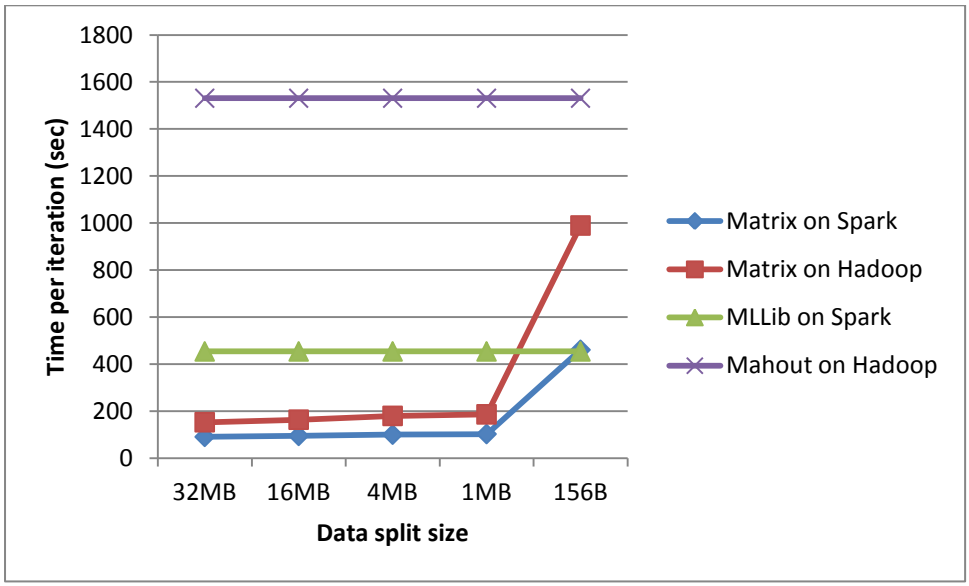


Figure 3.13: K-means time per iteration vs. different data split size.

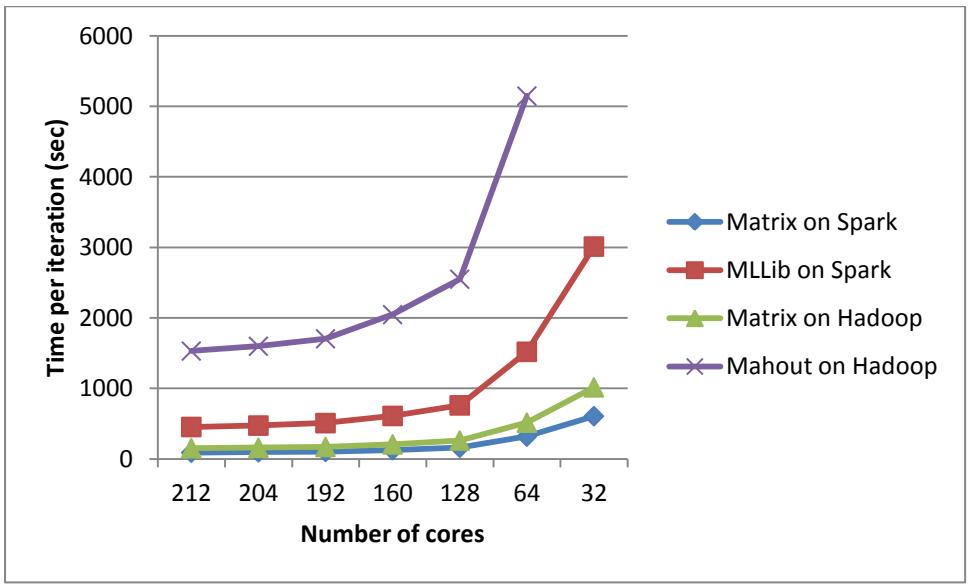


Figure 3.14: K-mean time per iteration vs. number cores in cluster.

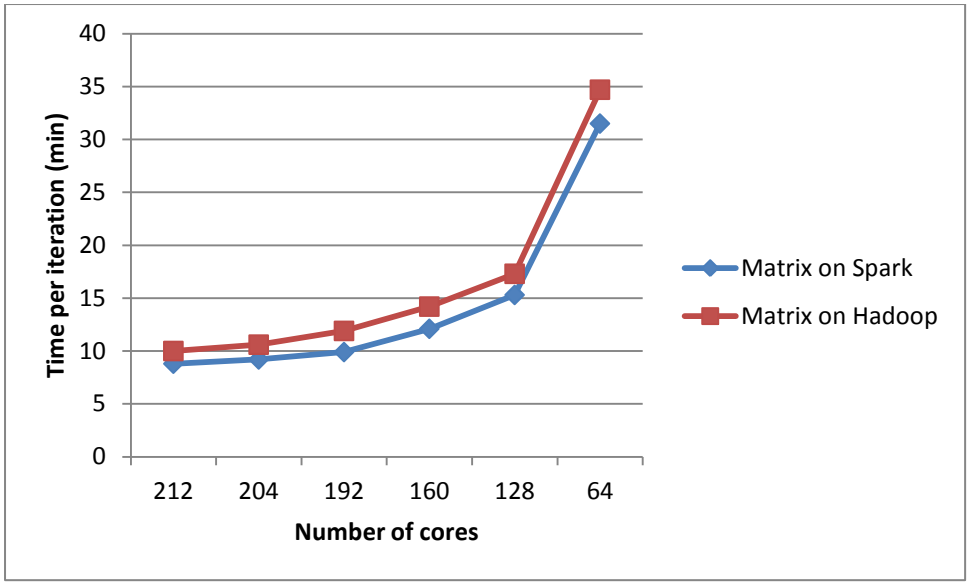


Figure 3.15: GMM time per iteration vs. number cores in cluster.

3.4.4 K-means and GMM refinement

K-means and GMM clustering results are highly sensitive to the initialization of centers and mixtures because both algorithms only guarantee to find a local optimal solution with respect to the initialization. To obtain reasonable initial centers, we first sub-sample the training data set into $\boldsymbol{X^{sub}}$ and then perform KKZ [Katsavounidis *et al.* 1994, He *et al.* 2004] algorithm on it. The advantage of sub-sampling KKZ is that outlier vectors located far from real data distribution have little possibility to be selected so that initial centers are near input data distribution yet sufficiently separated.

Even with decent initialization, k-means and GMM clustering can run into 3 awkward situations: 1) two or more centers compete to represent one cluster; 2) centers are trapped in very small clusters; 3) centers are stuck between two real data clusters which make the representing center or mixture looks too fat to split. In our experiment we developed a three stage method to detect and alleviate the 3 situations. In the first step to merge neighbors, Bhattacharyya distance between centers or mixtures are calculated. A "near" graph is then constructed. If two centers or mixtures are too close there will be an edge between them in the "near" graph. Independent connected components are then obtained from the graph to merge too near centers if any. The nearness threshold is set to 0.5. In the second step dead clusters are killed that is centers or mixtures with too small weight are eliminated. The smallness threshold is

calculated as one tenth of average weight. In the third step to split fat clusters, centers and mixtures with abnormally large variance in certain dimension are to split. The abnormally large threshold for one dimension is defined as 3 σ + the mean of standard derivations of all centers or mixtures. We have also tried to use kurtosis of Gaussian distribution to differentiate fat clusters. Unfortunately in high dimensional space kurtosis becomes insensitive even when the mixture covers two far away clusters. In our 39 dimensional data experiment we find kurtosis cannot effectively detect fat mixture compared with variance threshold. In the experiment of music genre and mood classification, 1.5% accuracy improvement is achieved with our three step refinement procedure.

Algorithm 3.1: Mixture refinement

Task: to refine the mixture obtained by EM algorithm

Input: GMM

Output: refined GMM

1. To merge near mixtures
   1.1. Calculate Bhattacharyya distance between centers or mixtures
   1.2. Construct a "near" graph: if two centers or mixtures are too close there will be an edge between them in the "near" graph.
   1.3. Obtain Independent connected components with the nearness threshold setting to 0.5.
2. To kill dead clusters with too small weight. The dead threshold is calculated as one tenth of average weight
3. To split fat mixtures with abnormally large variance in certain dimension. The abnormally large threshold for one dimension is set to 3 σ + the mean of standard derivations of all centers or mixtures.

Output result: refined GMM

3.5 Experiments on Music Genre and Mood Classification

To evaluate speed and quality of our implementations, music genre and mood classification have been conducted on GPU, multi-core CPU and Hadoop, Spark cluster.

3.5.1 Experiment setups

3.5.1.1 Computer configuration

A single thread implementation based on clapack[2] was used as a baseline using a single machine. It ran on a PC with Intel® Core™ i7-940 @2.93GHz. The implementation based on ACML ran on a server with 4 8-cored AMD Opteron™ Processor 6128 @2GHz and 100GB of memory. The implementation based on CUBLAS ran on 240-cored NVIDIA GTX 285. Yael[3] was also tested on the server as a benchmark for multi-threading implementation. The Hadoop and Spark clusters consist of 16 computers as shown in Table 3.2

Table 3.2: Cluster computer configurations.

Num.	CPU	Memory
2	4x16-cored AMD® Opteron™ CPU 6274 @2GHz	100GB
1	4x8-cored AMD® Opteron™ CPU 6128 @2GHz	100GB
3	1x4-cored Intel® Core™ i7 CPU 950 @3.07GHz (HT to 8-cored)	24GB
2	1x4-cored Intel® Core™ i7 CPU 940 @2.93GHz (HT to 8-cored)	24GB
8	1x4-cored Intel® Core™ i7 CPU 860 @2.80GHz (HT to 8-cored)	16GB

3.5.1.2 Datasets and features

[2] http://www.netlib.org/clapack/
[3] https://gforge.inria.fr/projects/yael

GTZAN [Tzanetakis *et al.* 2002] data set is used for genre classification. GTZAN contains 10 genres of music. Each genre contains 100 30-sec segments. MFCC with delta and delta delta features are extracted. MFCC analysis window shift is 10ms long, resulting in 100 39-dimensional feature vectors per second. To test k-means, GMM quality 10-fold cross validation is performed. In each fold K-means and GMM with 2048 clusters and mixtures are trained by 2.7 million 39-dimentional feature vectors.

Music mood classification is performed on two datasets. The smaller one is provided by Xiao *et al.* [Xiao *et al.* 2008] and it consists of 416 16s long pure classical music segments, manually divided into 4 moods as shown in Table 3.3. The same 39-dimentional MFCC features are extracted. K-means to GMM clustering are performed on both GPU and 32-cored server.

Table 3.3: Classical music dataset mood distribution.

	Anxious	Content	Depressed	Exuberant
# music	81	124	120	91

After dictionary learning, low level features of each music segments are converted to normalized dictionary words histogram. The normalized histograms as final feature vectors are used to train one-versus-others SVM classifier. To compare the mood classification performance, standard OpenSMILE [Eyben *et al.* 2010] low level features with statistic functions are extracted and used to train SVM as benchmark. The training and testing set are divided by 50%/50%. The SVM parameters are set to default. The running time and accuracy results of the two experiments are shown in Table 3.4 and Table 3.6-3.9.

The other music mood dataset evaluated is the "Now That's What I Call Music!" (NTWICM) dataset introduced in [Björn *et al.* 2010], containing 2648 songs annotated by four listeners on 5-point scales for perceived arousal and valence on song level. The NTWICM dataset is also served to benchmark efficiency of our proposed method with existing library of Mahout [Anil *et al.* 2011] and MLLib [Bahmani *et al.* 2012] on Hadoop and Spark cluster. For NTWICM datasets, the same 39-dimentional MFCC features are firstly

extracted. K-means, GMM clustering and MAP adaptation are then performed on the Hadoop and Spark clustering. Finally the GMM super vector serves as feature vector to train linear SVMs. The training and testing set are divided by 50%/50%. The SVM parameters are set to default. The running time on cluster and accuracy results are shown in Table 3.5

3.5.2 Results and analysis

3.5.2.1 Single computer

From Table 3.4 we can find that 1) high performance library based implementations proposed in this chapter achieve up to 5 times faster than multi-threading based implementation (Yael). 2) GPU based implementation executes 5 times faster than multi-core CPU based one on GMM training whereas multi-core version outperforms GPU on k-means clustering by 22% in terms of speed. GPU based k-means is slower than multi-core CPU in that CPU version executes $O(ND)$ operations for $\boldsymbol{C}^{blk}$ calculation while GPU version actually executes matrix multiplication containing $O(NDM)$ operations. We choose matrix multiplication for $\boldsymbol{C}^{blk}$ calculation in GPU because $O(ND)$ operations consume as twice time as $O(NDM)$ matrix multiplication on GPU architecture. This abnormal phenomenon is due to random memory access pattern of $O(ND)$ operations on GPU. Therefore in this scenario matrix multiplication is the optimal but still slower way for GPU.

In [Machlica *et al.* 2011] Machlica *et al.* reported a faster result of comparable data set, however, the duration profile in [Machlica *et al.* 2011] only recorded kernel function's running time without data IO and data preparation duration. If only GPU execution duration is considered, our method consumes 7 seconds compared with 9.1 in [Machlica *et al.* 2011]. From genre classification accuracy column in Table 3.4, we can find that the quality of dictionary trained by CPU is little better than GPU by less than 1% in terms of average classification accuracy. This is due to the less floating point error of CPU and double precision floating point numbers used during summation procedure on CPU.

From Table 3.6, we can find that with the same number of mixture, the same types of GPU and comparable data scale as in [Azhari *et al.* 2011], our

CUBLAS based implementation is 10 times faster than CUDA kernel implementation described in [Azhari *et al.* 2011]. From Table 3.7-3.9 we can find that 1024 dimensional bag-of-words histogram features outperforms the standard 6552 dimensional OpenSMILE emotional features by 3% in terms of average classification accuracy.

Table 3.4: Execution time and accuracy of different implementation for music genre classification.

# Centers/# Mixtures = 2048 # Vectors = 2.7M # Dimension = 39	K-means			GMM			MAP		
	time per iteration (sec)	speed-up (times)	Genre Acc (%)	time per iteration (sec)	speed-up (times)	Genre Acc (%)	time per song(sec)	speed-up (times)p	Genre Acc (%)
1 thread CPU	278.2	1.0		2,288.4	1.0		1.5	1	
32-cored CPU Yael(INRIA)	37.2	7.5	68.1	153.2	14.9	78.0			
32-cored CPU (ours)	7.3	38.0	69.6	70.1	32.6	78.7	0.07	21	80.5%
240-cored GPU	9.1	30.4	69.5	10.9	209.5	77.8	0.02	75	79.4%
UWB[Machlica *et al.* 2011]				9.1*					

*only kernel execution time counted for GMM with 2048 mixtures trained by 3.125M 40-dimensional vectors.

Table 3.5: Execution time and accuracy of different implementation for music mood classification.

# Centers/# Mixtures = 4096 # Vectors = 60M # Dimension = 39	K-means				GMM			MAP		
	time per iteration	Speed-up (times)	Valance Acc (%)	Arousal Acc (%)	time per iteration	Valance Acc (%)	Arousal Acc (%)	time per song	Valance Acc(%)	Arousal Acc (%)
Mahout on Hadoop	1530sec	1.0								
MLLib on Spark	453.7sec	3.4	54.1	48.9						
Matrix format on Hadoop	152.4sec	10.0	55.0	49.7	10.0min	55.9	50.3	0.71sec	56.7	51.8
Matrix format on Spark	90.2sec	17.0			8.8min			0.65sec		

Table 3.6: Running time per iteration for mood GMM training.

	32-cored CPU	240-cored GPU	Azhari [Azhari *et al.* 2011]
time (sec)	9.6	2.9	30.0

Table 3.7: CUBLAS mood classification confusion matrix (actual classes in rows, predicted classes in columns).

%	Anx	Con	Dep	Exu
Anx	81.0±8.9	1.2±2.0	1.0±1.7	16.8±7.4
Con	0.0±0.0	91.1±2.9	8.7±2.9	0.2±0.5
Dep	0.0±0.0	8.8±3.7	90.5±3.2	0.7±1.5
Exu	10.2±4.7	3.8±2.0	0.0±0.0	86.0±4.2
Average	87.2			

Table 3.8: ACML mood classification confusion matrix (actual classes in rows, predicted classes in columns).

%	Anx	Con	Dep	Exu
Anx	81.0±7.8	1.2±2.0	1.2±1.7	16.5±6.2
Con	0.0±0.0	91.9±3.0	8.1±3.0	0.0±0.0
Dep	0.0±0.0	8.3±3.5	91.0±3.2	0.7±1.5
Exu	11.3±5.3	3.6±1.8	0.0±0.0	85.1±5.0
Average	87.3			

Table 3.9: OpenSMILE mood classification confusion matrix (actual classes in rows, predicted classes in columns).

%	Anx	Con	Dep	Exu
Anx	85.5±7.1	2.5±1.6	0.5±1.0	11.5±6.8
Con	5.5±4.7	86.9±4.2	6.3±3.0	1.3±2.7
Dep	9.0±6.3	6.2±3.9	83.5±7.9	1.3±3.0
Exu	17.6±5.0	2.2±1.4	0.2±0.7	80.0±5.9
Average	84.0			

3.5.2.2 Hadoop and Spark clustering

Compared with the state-of-the-art implementation, from Table 3.5, we can find that the propose method for k-mean is 10 times faster than Mahout on Hadoop and 5 times faster than MLLib on Spark. Matrix format k-means on Hadoop even runs 3 times faster than MLLib does on Spark. Comparing Hadoop with Spark, we can also find that for iterative algorithms Spark with data caching mechanism runs faster than Hadoop. However, when computation becomes more expensive with the same amount of data, the gap between the two frameworks is shortening.

The reason why the proposed matrix format outperforms the two libraries is because of the JVM thread scheduling overhead. Compared with Mahout and MLLib in which one thread is arranged to one data vector to perform the calculation, the proposed method directly feeds large chunk of data to highly tuned math library and let the library decide the optimal thread configuration to run the calculation.

3.6 Conclusion

In this chapter, we have proposed to use matrix format on GPU multi-core CPU and cluster to accelerate bag-of-words and GMM super vector method, especially for dictionary learning of k-means and GMM clustering. To employ high performance matrix operation library of ACML, ATLAS and CUBLAS, we

propose to reformulate k-means and EM algorithm into matrix multiplications, which is also convenient to be implemented with other languages for example MATLAB and FORTRAN. Experiments on music genre and mood classification tasks show that the proposed implementations achieve 38 to 209 times acceleration, compared with single threaded CPU version on single machine. 240-cored GPU can run up to 5 times faster than 4 8-cored CPUs with just less 1% performance decline. On Hadoop and Spark cluster, the proposed method still achieves 10 and 5 speed-up, compared with Mahout and MLLib.

Our proposed approach thus allows going a step further for the modeling of low level features when dealing with large scale data. As low level features are important but generally insufficient to depict the whole picture of music, we will discuss in the next chapter mid-level features in particular note histogram feature with its application in content based music classification.

Chapter 4: Sparse Music Decomposition with MIDI Dictionary and Musical Knowledge

4.1. Introduction

As shown in previous chapters most of the automated music analysis approaches available in the literature rely on the representation of the music through a set of low-level audio features related to frequential, energy and temporal properties. Identifying high-level concepts, such as music mood, from this "black-box" representation is particularly challenging, even with BoW and GSV.

Therefore we propose in this chapter our new representation of the music that allows gaining an in-depth understanding of its structure and harmony that we expect to be important for further music analysis, and particularly mood classification. Its principle is to decompose sparsely the music into a basis of elementary audio elements, called musical words, which represent the notes played by various instruments generated through a MIDI synthesizer. To do so, our approach relies on the sparse theory [Elad 2010] and its essence is to represent the music, thanks to a "musical dictionary" made of over complete musical words that are related to the notes produced by various instruments. From this sparse representation, we also define a new feature which favors further mood classification. Experiments driven on two music datasets have shown the effectiveness of this approach to represent accurately music signals and to allow efficient music mood classification.

To achieve more accurate decomposition, statistical music knowledge is further integrated into the whole sparse decomposition process. We then present our two-stage sparse decomposition approach integrated with music knowledge. In frame level decomposition stage, note co-occurrence probabilities are embedded to guide atom selection in modified matching pursuit algorithm with a MIDI dictionary. A sparse multiple candidate graph is then constructed to provide backup choices for later selections. In the global optimal path searching stage, note transition probabilities are incorporated together with a goodness measure of

frame decomposition. Its principle is to guide the local sparse music decomposition with co-occurred notes information and decode the global optimal decomposition path with consecutive note knowledge. Due to the Gabor limit, time and frequency resolution cannot be well satisfied at the same time. Thus, we emphasize the frequency resolution aspect rather than the exact time location, since correct note recognition is more important for our following classification task. Experiments on real-world polyphonic music show that embedding music knowledge within the sparse decomposition achieves notable improvement in terms of note recognition precision and recall.

The rest of the chapter is organized as follow. Section 4.2 introduces sparse music decomposition. To build a musical dictionary, two sets of MIDI based instrument sound are generated, which is detailed in section 4.2.2. A modified matching pursuit algorithm is presented in section 4.2.3 to perform music decomposition subject to a natural positive constraint and note histogram feature is also defined. In section 4.2.4, the quality of sparse music decomposition is verified and mood classification experiments on two data sets are conducted. Section 4.3 introduces our two-stage approach in detail. Section 4.3.1 explains our frame level statistical integration while section 4.3.2 shows how global level optimal paths are constructed. Section 4.3.3 gives experimental results on real-world music signals. The conclusion is drawn in the final section.

4.2. Sparse Decomposition of Music Using a Musical Dictionary

4.2.1. Overview of the approach

The general principle of our music sparse representation and feature extraction is illustrated in Figure 4.1. It relies on a musical dictionary made of musical words. Each of these words corresponds to the spectrum of a note produced by an instrument through a MIDI synthesizer. To decompose a music signal onto this musical dictionary, the signal is firstly framed by Hamming window and the spectrum is computed in each of these frames. The sparse solution solver then decomposes each of the signal frames into the musical dictionary to obtain the sparse solution x describing the musical words that are used to represent the

frame content. The sparse solutions x are finally aggregated and normalized over all frames to generate an instrument note histogram, or musical histogram, which will form the feature representing the input music signal for further classification.

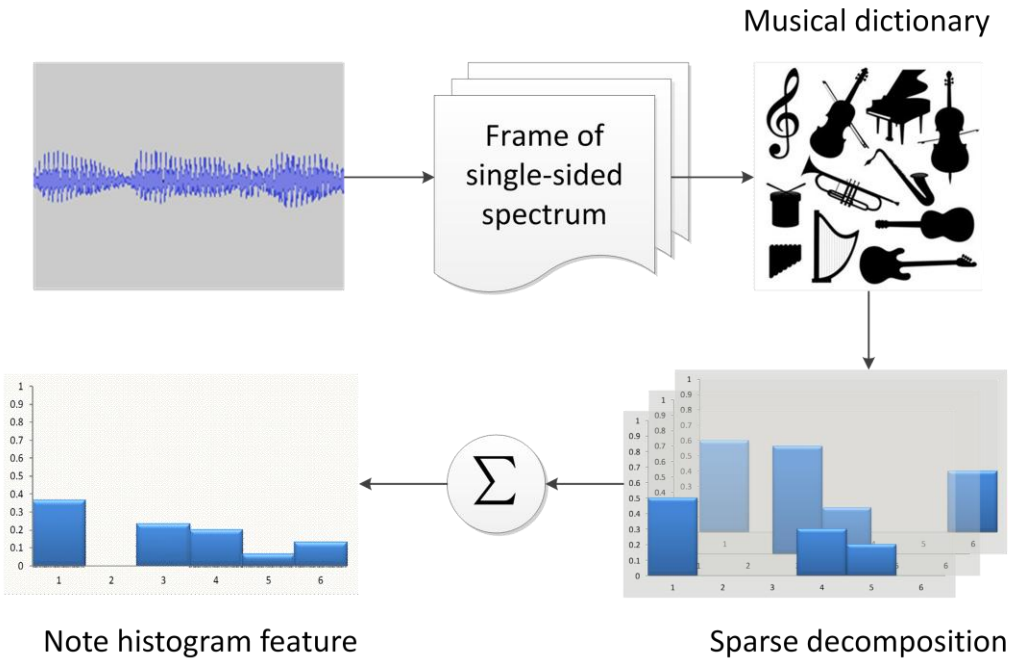


Figure 4.1: Musical histogram feature extraction flow chart.

4.2.2. MIDI musical dictionary

Elaborating an appropriate musical dictionary is a key issue since this set of atoms, or musical words, has to be rich enough to allow accurate music decomposition, in terms of notes and instruments. Unfortunately, it is infeasible to record such a huge data set. Although [Leveau 2008] has suggested learning atoms from instrument playing recordings, it is still impractical to apply to a large instrument set. Therefore, in order to get adequate instrument note sounds we propose to make use of a MIDI synthesizer.

Indeed, a MIDI synthesizer is able to generate various instrument sounds on almost all possible notes. In our system two sets of MIDI synthesized waves are tested. One is produced by "Windows Vista built-in Microsoft GS Wavetable

Synthesizer". The other is generated from Logic Pro 9, a high quality music producing software from Apple Inc.

For wavetable synthesizer we select all 128 "general MIDI level 1" sound instruments and 47 "general MIDI level 1" percussion instruments[1]. For Logic Pro 9, we choose the first 80 realistic instruments as in general MIDI level 1 set and 31 percussion instrument sets including 1860 percussions. For each instrument, we choose 60 notes form note 31 to note 90. The 60 notes span 5 octaves from low to high[2], covering most instruments' playing note range.

The duration of each note is set to 186ms, which are 4096 samples under a sample rate of 22050Hz. This duration is long enough to hold one attack-decay-sustain-release envelope (ADSR) and leads to 5.38Hz in terms of frequency resolution, which is sufficient to discriminate adjacent note spectrum. This window length is also convenient for FFT computation. Our MIDI musical dictionary is finally built by computing each musical word in frequency domain as the single-sided power spectrum of each note wave generated by short time Fourier transform (STFT) with Hamming window. The STFT is applied because with 5.38Hz resolution single-sided spectrum for each note is linearly addable and more robust against time-frequency transform like wavelet transform. According to our previous experiment atoms of wavelet transform coefficients cannot compare with FFT ones due to its sensitivity of time position and the quality of input signal.

A musical word is thus a 2048 dimensional vector. The number of musical words is decided by the number of instruments multiplying the number of notes (60) plus the number of percussion instruments. For the dictionary generated by "general MIDI level 1", 7727 musical words are created. For Logic Pro 9, 6660 musical words are created.

4.2.3. Sparse representation and positive constraint matching pursuit

Sparse representation [Elad 2010] is originated from finding the solution x^* of an underdetermined linear system $Dx = y$ so that x^* contains as few non-zero

[1] http://www.midi.org/techspecs/gm1sound.php
[2] http://tonalsoft.com/pub/news/pitch-bend.aspx

components as possible. This problem is formulated as $\min_x \|x\|_0$ subject to $Dx = y$. In many cases $Dx = y$ is hard to satisfy, thus in practice $\|x\|_0$ and $\|Dx - y\|_2$ are minimized simultaneously. When applied to signal processing, y represents the signal to analyze, D is an over-complete dictionary with atoms that can reconstruct y and x^* is the sparse interpretation of y under D. For similar example inverse Fourier matrix F^{-1} serves as dictionary to reconstruct signal y using its spare frequency domain coefficients x, although F^{-1} is not over-complete.

With a MIDI musical dictionary in hand, any sparse solution solver can be applied to produce sparse solution in respect to note. Unfortunately classical greedy algorithms, for example orthogonal matching pursuit (OMP), cannot directly be applied to the decomposition. Because each musical word in MIDI dictionary is a single-sided power spectrum of certain note, a positive constraint is naturally imposed onto the sparse solution. In another word, any negative component of sparse solution is prohibited, as any negative appearance of certain note is impossible.

We propose a positive constraint matching pursuit (PCMP) algorithm detailed in Algorithm 1 to solve this problem. The difference between OMP [Elad 2010] and the proposed algorithm is the positive constraint introduced in the step 3 of main iteration. For OMP, least mean square (LMS) suffices to solve the minimization in step 3, which also results in residual r^k orthogonal to support S^k. For PCMP, however, after positive constraint minimization, orthogonality between r^k and S^k is not guaranteed, thus the algorithm is turned to a special matching pursuit.

Algorithm 1: Positive constraint matching pursuit

Task: Approximate the solution of problem: $min_x \|x\|_0$, subject to $Ax = b, x \geq 0$

Input: Dictionary A, signal b, max iteration number N and error threshold ϵ_0.

Output: sparse solution: x^*

Initialization:

Initial solution: $x^0 = 0$.

Initial residual: $r^0 = b$.

Initial support: $S^0 = \emptyset$.

Main iteration: for $k = 1 \; to \; N$

1. Compute error: $\epsilon(j) = min_{z_j} \left\| a_j z_j - r^{k-1} \right\|_2^2$ for all j using the optimal choice $z_j^* = a_j^T r^{k-1} / \left\| a_j \right\|_2^2$.
2. Update support: Find a minimizer, j_0 of $\epsilon(j)$: $\forall j \notin S^{k-1}, \epsilon(j_0) < \epsilon(j)$, and update $S^k = S^{k-1} \cup \{j_0\}$.
3. Update provisional solution: Compute x^k, minimizer of $\|Ax - b\|_2^2$ subject to $support\{x\} = S^k, x \geq 0$.
4. Update residual: $r^k = b - Ax$.
5. If $\|r^k\|_2^2 < \epsilon_0$ break.

Output result: $x^* = x^k$.

4.2.4. A musical histogram as audio feature

As mentioned previously, in order to perform the music signal decomposition onto the musical dictionary, the music is firstly framed and the spectrum is computed in each of these frames. Then, a sparse decomposition of each frame is computed using PCMP onto the musical dictionary. The obtained sparse decompositions finally need to be combined in order to obtain a feature that can represent the whole music signal for further classification. To do so, we have defined a musical histogram that is built from the aggregation of the sparse representation of each frame using the following process. Let F be the total number of frames of the input music. For each frame i, compute decomposition sparse solution x_i. The dimension of x_i is equal to the number of musical words in the dictionary. The musical histogram feature h is computed as $h = \sum_i x_i / F$. For the consideration of efficiency and complete covering, in the following experiments frame shifting length is fixed to 186ms, the same as frame length.

4.2.5 Experiments and results

We present in the following subsections experiments we have driven in order to evaluate the ability of the proposed method to accurately decompose music, as well as the efficiency of our musical histogram feature for the problem of music mood classification.

4.2.5.1. Verification of the decomposition with MIDI dictionary

To evaluate the decomposition quality with proposed PCMP and MIDI musical dictionary, a multi-timbral music with time domain note reference has been used, which is from Mirex2007 multiF0 development data[3]. The music is a recording of the fifth variation from *L. van Beethoven Variations from String Quartet Op.18 N.5*, lasting for 54s. 5 instruments are included in the music: bassoon, clarinet, flute, horn and oboe. Each instrument was recorded separately while the performer is listening to the other parts through headphones. Later the 5 parts were mixed to a mono 44.1 kHz 16 bits wave file.

[3] http://www.music-ir.org/mirex/wiki/2007

In this evaluation, only Logic Pro 9 MIDI dictionary has been tested. Figure 4.2 shows precision and recall scatter diagram of the decomposition. Statistic of note recognition precision and recall has been made upon consecutive 186ms segments, the same as in computing the musical histogram feature. From Figure 4.2 we can find that when there is no threshold imposed on sparse solution, 69% of notes can be recalled, although the precision is 40%. If we only consider the recognition of notes, no matter the octave, the recall and precision increase to 85% and 47% as presented in Figure 4.3. These results show that the proposed PCMP with MIDI musical dictionary can detect the structure of multi-timbral music with a satisfactory efficiency.

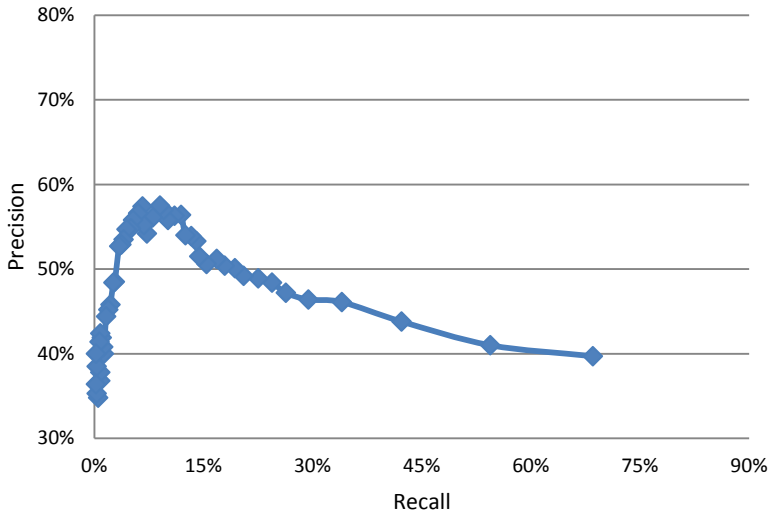


Figure 4.2: Note precision vs. recall.

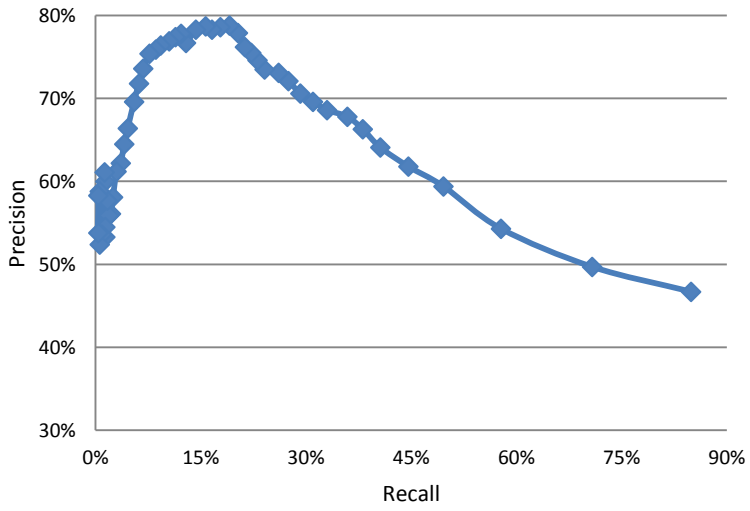


Figure 4.3: Note precision vs. recall with octave tolerance.

4.2.5.2. Mood classification with musical histogram feature

Two music datasets have been used in our experiments to evaluate the performance of our musical histogram feature for the problem of music mood classification. We have only considered datasets containing voiceless music titles, since our MIDI dictionary covers at present only instrument sounds. The first dataset provided by Xiao *et al.* [Xiao *et al.* 2008] consists of 416 16s long pure classical music segments, manually divided into 4 moods as shown in Table 4.1. According to [Xiao *et al.* 2008], 16s long segments are most sensitive to mood for this dataset. The second dataset contains 401 30s long segments of modern music from various genres obtained from APM Music web site[4] that provides the mood labels. The 401 excerpts are equally selected from 4 moods as shown in Table 4.2. The same type of dataset is also used for the mood classification task of MIREX challenge. Compared with the first classical dataset, APM music enjoys more

[4] http://www.apmmusic.com/myapm/main.php

variety not only in genres (classical, pop, jazz, rock etc.) but also in diversified musical instruments.

Table 4.1: Classical music dataset mood distribution**.**

	Anxious	Content	Depressed	Exuberant
# music	81	124	120	91

Table 4.2: APM dataset mood distribution.

	Fear	Happy	Relax	Sad
# music	101	100	100	100

In addition to our musical histogram feature, we have made use of the well-known large emotion feature set of openSMILE [Eyben *et al.* 2010] as a comparison reference. This large emotion feature set contains 6552 features including various time and frequency domain feature such as zero cross rate, log energy, filter bank energy, MFCC, pitch etc. and statistical functions performing on them. In our experiments, 5 times 2-fold cross-validation has been performed. For each time and each fold, the same training and testing division has been used by both openSMILE and musical histogram features. SVM classifiers with RBF kernel have been employed and a simple averaging probability output by SVM has been used for late fusion of the two classifiers results.

4.2.5.3. Results and analysis

The averaged confusion matrices for the first classical music dataset are presented in Tables 4.3-4.9. The results obtained by Xiao *et al.* [Xiao *et al.* 2008a, Xiao *et al.* 2008b] are shown in Table 4.3. From Table 4.4 and 4.5 we can find that Gaussian super vector outperforms OpenSmile feature by 4.3 percents. Comparing Tables 4.6 with 4.7, we can find that, no surprisingly, the dictionary built with high quality instrument waves (Logic Pro 9) produces a more discriminative musical histogram feature, probably due to its higher ability to accurately decompose the music signal. According to Table 4.8 and 4.9, when late

fusion is performed on Logic Pro 9 musical histogram feature with openSMILE features and GSV, additional 3.7% and 4.4% improvement are further obtained on average accuracy. Note that for each individual mood late fusion achieves best accuracy too. Speaking of individual feature, from Table 4.4, 4.5, 4.6, 4.7, we can find that the musical histogram feature generated from Logic Pro 9 MIDI dictionary outperforms the other 3 by 2% to 7% on average accuracy. For each individual mood, Logic Pro 9 MIDI dictionary also surpasses the others.

Table 4.3: Xiao [Xiao *et al.* 2008b] confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset.

%	Anx	Con	Dep	Exu
Anx	78.31	1.85	18.47	1.37
Con	2.10	79.14	0.00	18.77
Dep	10.25	1.33	88.08	0.33
Exu	2.20	11.43	0.11	86.26
Average	82.95			

Table 4.4: OpenSMILE confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset.

%	Anx	Con	Dep	Exu
Anx	85.5±7.1	2.5±1.6	0.5±1.0	11.5±6.8
Con	5.5±4.7	86.9±4.2	6.3±3.0	1.3±2.7
Dep	9.0±6.3	6.2±3.9	83.5±7.9	1.3±3.0
Exu	17.6±5.0	2.2±1.4	0.2±0.7	80.0±5.9
Average	84.0			

Table 4.5: GSV confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset.

%	Anx	Con	Dep	Exu
Anx	87.1	2.5	0.5	11.5
Con	3.5	92.9	2.3	1.3
Dep	4.1	5.6	88.4	1.9
Exu	8.6	6.8	0.1	84.7
Average	88.3			

Table 4.6: General MIDI confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset.

%	Anx	Con	Dep	Exu
Anx	81.5±5.8	1.0±1.2	8.3±6.2	9.3±3.5
Con	0.2±0.5	93.1±4.3	6.1±3.9	0.6±1.1
Dep	4.5±3.1	8.3±2.1	82.7±4.4	4.5±2.6
Exu	8.4±4.7	6.2±2.6	6.7±3.6	78.7±5.8
Average	84.0			

Table 4.7: Logic Pro 9 confusion matrix. (actual classes in rows, predicted classes in columns) using the classical dataset.

%	Anx	Con	Dep	Exu
Anx	87.3±3.6	2.5±2.5	2.0±1.5	8.2±4.6
Con	0.8±1.1	96.0±2.3	1.3±1.4	1.9±1.9
Dep	2.2±2.9	5.3±2.4	90.0±2.9	2.5±1.9
Exu	5.1±1.4	5.8±2.8	2.7±1.3	86.4±3.5
Average	90.0			

Table 4.8: OpenSMILE fused with PCMP confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset.

%	Anx	Con	Dep	Exu
Anx	91.5±1.7	1.0±1.2	1.0±1.2	6.5±2.3
Con	0.2±0.5	96.3±2.7	3.4±2.3	0.2±0.5
Dep	1.2±1.7	3.2±1.7	94.7±2.3	1.0±1.1
Exu	5.8±4.2	1.6±1.4	0.2±0.7	92.4±4.1
Average	93.7			

Table 4.9: GSV fused with PCMP confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset.

%	Anx	Con	Dep	Exu
Anx	92.4	0.6	1.0	6.0
Con	0.2	96.5	3.1	0.2
Dep	1.3	3.2	94.2	1.3
Exu	4.2	1.2	0.2	94.4
Average	94.4			

The averaged confusion matrices for the second APM music dataset are presented in Tables 4.10-4.14. Note that latest best result in MIREX2010 on a similar data source with more mood classes achieved 64% on average accuracy. Table 4.10 and Table 4.11 show that GSV still outperform OpenSMILE by 2%. From Table 4.10-4.12, we can find that compared with the classical music set the overall performance of musical histogram feature, openSMILE and GSV decline by 30%, 10% and 11.5%. OpenSMILE feature set outperforms Logic Pro 9 musical histogram feature by 15.5% and GSV surpass musical histogram feature by 17.4% However, according to Table 4.13 and 4.14, when these two levels of feature sets are combined in the late fusion, the average accuracy increases by 2.5% and 2.1%. Several other conclusions can also be drawn from these results. First, the musical histogram feature is less effective with this modern music dataset than

with the classical music dataset. The reason certainly lies in the MIDI dictionary that may have difficulties to accurately decompose this music due to its important complexity and the presence of instruments in music that are missing in the dictionary. This problem should be solved by using a MIDI synthesizer producing more realistic instrument sounds. Second important information provided by these results is that even though openSMILE feature set provides very complete low-level information characterizing temporal and frequential signal properties, its performance for classification can be improved by enriching this description with higher level information that we propose with our musical histogram feature which allows to explicit the music content.

Table 4.10: OpenSMILE confusion matrix (actual classes in rows, predicted classes in columns) using the APM dataset.

%	Fear	Happy	Relax	Sad
Fear	83.2±6.3	5.2±3.2	2.4±2.7	9.2±4.1
Happy	11.6±6.4	78.2±9.2	8.8±5.3	1.4±2.5
Relax	10.4±3.9	8.8±4.0	68.8±5.9	12.0±5.4
Sad	23.0±7.8	1.4±1.3	6.2±3.3	69.4±8.2
Average	74.9			

Table 4.11: GSV confusion matrix (actual classes in rows, predicted classes in columns) using the APM dataset.

%	Fear	Happy	Relax	Sad
Fear	86.2	3.3	2.4	8.1
Happy	9.5	81.4	7.8	1.3
Relax	10.2	8.6	69.8	11.4
Sad	23.0	1.4	5.7	69.9
Average	76.8			

Table 4.12: Logic Pro 9 confusion matrix (actual classes in rows, predicted classes in columns) using the APM dataset.

%	Fear	Happy	Relax	Sad
Fear	61.8±7.7	12.2±4.5	7.4±4.2	18.6±3.0
Happy	8.6±3.1	67.2±6.8	17.4±5.5	6.8±2.4
Relax	9.0±2.7	19.8±4.8	45.8±6.8	25.4±5.1
Sad	11.4±5.2	6.4±3.1	19.4±6.5	62.8±5.7
Average	59.4			

Table 4.13: OpenSmile fused with PCMP confusion matrix (actual classes in rows, predicted classes in columns).

%	Fear	Happy	Relax	Sad
Fear	80.6	4.4±2.0	1.4±1.3	13.6±4.9
Happy	5.4±3.5	83.8±7.6	7.8±6.0	3.0±2.9
Relax	7.8±4.2	10.2±3.9	68.6±6.9	13.4±4.7
Sad	17.8±5.8	0.4±0.8	5.4±2.4	76.4±6.1
Average	77.4			

Table 4.14: GSV fuse PCMP confusion matrix (actual classes in rows, predicted classes in columns) using the APM dataset.

%	Fear	Happy	Relax	Sad
Fear	82.3	4.0	1.4	12.3
Happy	5.3	84.0	7.7	3.0
Relax	7.1	10.0	70.9	12.0
Sad	16.3	0.4	4.9	78.4
Average	78.9			

4.3 Sparse Decomposition with Note Statistics

Our proposed method consists of two main steps. In the first step, the entire music signal is framed and a modified orthogonal matching pursuit algorithm is performed on each frame to generate decomposition candidates. In the second step, decomposition candidates are connected to form a directed graph and an optimal path is then constructed to produce the final decomposition result.

4.3.1 Frame level sparse decomposition

Former study shows that elaborating an appropriate musical dictionary is a key issue since this set of atoms has to be rich enough to characterize the varieties of real word music. Although [Leveau *et al.* 2008] has developed sophisticated method to learn atoms from instrument recordings, it is still impractical to apply to a large instrument set. Therefore, in order to get adequate instrument note sounds, we propose to make use of a MIDI synthesizer. Logic Pro 9 is employed in our approach to generate the MIDI note dictionary because of its huge instrumental library and the high sound quality. Unlike pre-installed MIDI synthesizer with sound card, Logic Pro 9 uses a large number of real instrument recordings to make synthesized wave signal as natural as possible.

To build the MIDI dictionary, we choose the first 80 realistic instruments as in general MIDI level 1 set and 31 percussion instrument sets including 1860 percussion sounds. For each instrument, we keep 60 notes from note 31 to note 90. The 60 notes span 5 octaves from low to high, covering most instrumental playing range. The duration of each note is set to 186ms, which are 4096 samples under a sample rate of 22050Hz. This duration is long enough to hold one attack-decay-sustain-release (ADSR) envelope and leads to 5.38Hz in terms of frequency resolution, which is sufficient to discriminate adjacent notes in piano roll. Our MIDI note wave is then converted into a single-sided power spectrum obtained by applying the short time Fourier transform (STFT) with a Hamming window. The final MIDI dictionary thus contains 6660 2048-dimensional vectors.

The adoption of the sparse representation is based on the hypothesis that during a 186ms time slot, there will not be many notes played together. Therefore,

concurrent notes are sparse within one frame. Sparse representation [Elad 2010] is originated from finding the solution x^* of an underdetermined linear system $Dx = y$ so that x^* contains as few non-zero components as possible. In most cases $Dx = y$ is hard to satisfy, thus in practice $\|x\|_0$ and $\|Dx - y\|_2$ are minimized simultaneously instead.

Armed with our MIDI dictionary, the classical matching pursuit algorithm like orthogonal matching pursuit (OMP) [Pati *et al.* 1993] must be modified because the single-sided power spectrum words in MIDI dictionary impose an inherent positive constraint on sparse solutions. In other words, any negative component of a sparse solution is prohibited, as negative appearance of certain notes is impossible. To solve this problem we adopt a positive constraint matching pursuit (PCMP) algorithm that is mentioned in [Gao *et al.* 2012, Bruckstein *et al.* 2008]. The difference between OMP and PCMP is in updating a provisional solution step: for OMP, least mean square (LMS) suffices to solve the minimization resulting in residual signals orthogonal to support set. For PCMP, however, after the positive constraint minimization, orthogonality is not always guaranteed, thus the algorithm is turned to a weak orthogonal matching pursuit.

When scrutinizing the decomposition results of PCMP within one frame, we found a number of irregular note combinations. This is due to PCMP's over-fitting target signals without considering any compatibility of concurrent notes. In fact, atom selection in each iteration of orthogonal matching pursuit algorithm is very important. OMP guarantees that expending support set with any linear independent atoms will decrease the reconstruction error and at the same time keep the residual signal orthogonal to the new expanded support set. Any atom selected in the support set will permanently reside. Therefore previously selected atoms have a great influence on following ones and alter the overall OMP performance. Although PCMP does not always hold orthogonal property, the principle remains the same.

Selecting a new atom in dictionary is thus the very place where concurrent note heuristic information should be embedded. To formulate concurrent note

information, Bayes model is employed in our approach to approximate the posterior probability of potential note given observed notes

$$P(X|\boldsymbol{O}) = \frac{\prod_{i=1}^{N} P(O_i|X)P(X)}{\sum_Y \prod_{i=1}^{N} P(O_i|Y)P(Y)} \tag{4.1}$$

where $\boldsymbol{O} = \{O_1, O_2 \cdots O_N\}$ denotes N observed notes obtained by first N PCMP iterations, X represents a potential co-occurred note with $\boldsymbol{O}$. The note prior probability $P(X)$ and the note co-occurrence posterior probability $P(X|Y)$ are estimated from our classical music MIDI database. To obtain $P(X|Y)$, a joint distribution $P(X,Y)$ is firstly estimated by accounting the frequency with overlap degree of the concurrent note X and Y. Then $P(X|Y)$ is obtained by normalizing $P(X,Y)$ over Y. Although equation (4.1) provides instructive information to help select appropriate note combinations, it is still risky to only consider the best note decomposition, since the second best one may be more appropriate in adjacent note context. To avoid the one best bias, we propose to preserve multiple candidates to give top-N best decompositions chances to recover in optimal path searching.

Orthogonal matching pursuit is a greedy algorithm. In each iteration only the best atom will be added into support set. This can be risky in some cases, since once a "bad" atom is selected, this error cannot be corrected in the future. In [Chen *et al.* 2001], it has been shown that it is possible to select "bad" atom initially so as to trap OMP from reconstructing target signals. Methods like OCMP in [Rath *et al.* 2008] are proposed to overcome the problem. However, in music decomposition the same note in different octave or from the same kind of instruments shares the similar harmonic pattern. Therefore it is hazardous to rule out a suboptimal decomposition too early before adjacent note compatibility is checked.

To overcome this drawback of OMP, we propose to keep N best candidates in each iteration instead of only one. To measure the goodness of frame decomposition we define $goodness = \alpha \log(P(\boldsymbol{X})) - \|\boldsymbol{r}\|_2^2$, where $\boldsymbol{X}$ is sparse note decomposition vector, $\boldsymbol{r}$ is decomposition residual signal, $P(\boldsymbol{X}) =$

$\sum_Y \prod_{i=1}^{N} P(X_i|Y)P(Y)$ denotes note concurrent probability, α is a free parameter that balances concurrent probability term and reconstruction error term.·As an example shown in Figure 4.4 we keep the top 3 decomposition candidates in every iteration. In the first iteration (C), (E), (G) are kept. In the second iteration, (C, D), (E, F) and (E, G) are obtained according to the reconstruction error and concurrent probability. Note that (G) selected in the first iteration is eliminated because its descendant combinations (G,*) are inferior to others'. After 3 iterations, combinations of (C, D, E), (C, D, G) and (E, F, B) survive, as shown in orange.

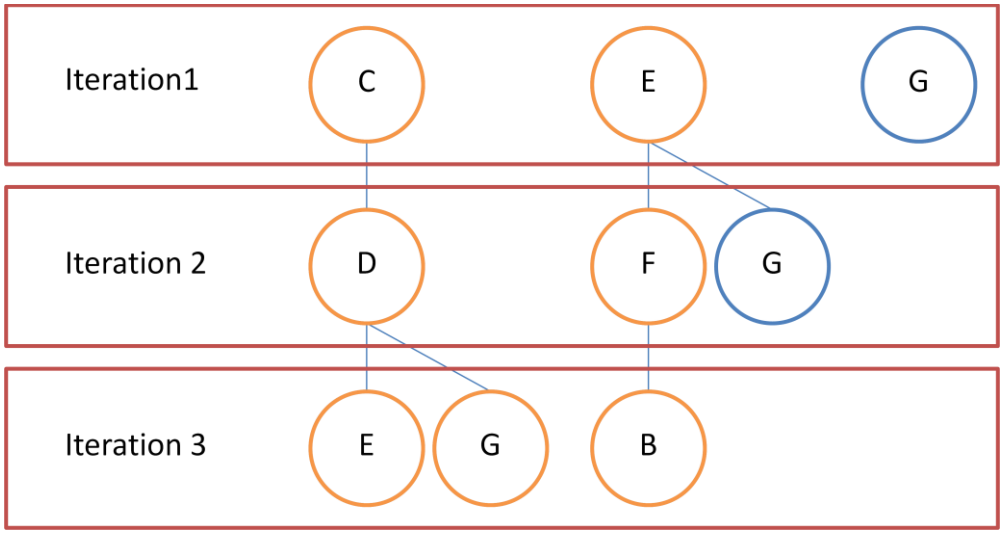


Figure 4.4: Multiple candidate selection example

When sparse decomposition terminates, the top N note candidates are derived for every signal frame. The best one can be treated as the decomposition result of the current frame. Besides, all candidates are preserved for constructing the optimal decomposition path when we further investigate inter-frame relations. The multiple candidate PCMP algorithm that we propose is summarized in Algorithm 2.

4.3.2 Global level optimal note path searching

All previous steps in sections 4.3.1 focus on improving sparse note decomposition within one signal frame. When further scrutinizing the PCMP decomposition between consecutive frames, we can still find a number of discontinuous note decompositions, in which the note sequence has sudden abnormal jumps in adjacent frames, including octave shift or sharp/flat drift. This is due to a lack of note transition regulation and because the sparse decomposition only minimizes reconstruction error in current frame without considering any neighbor frame contexts.

Besides the co-occurred ones, consecutive notes bear strong correlations which convey various melodies, temporal and dynamic information of music. It is reasonable to incorporate such sequential knowledge of notes as to suppress the discontinuous note error.

Algorithm 4.2:Positive Constraint Matching Pursuit Producing Multiple Candidates

Task: Approximate the solution of problem: $min_{\boldsymbol{x}}\|\boldsymbol{x}\|_0$, subject to $\boldsymbol{Ax} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}$.

Input: Dictionary $\boldsymbol{A}$, signal $\boldsymbol{b}$, max iteration number I, top N candidates to keep, balance parameter α , note posterior probability $P(X|Y)$ and error threshold ϵ_0.

Output: sparse solution: $\boldsymbol{x}^*$

Initialization:

Initial residual: $\boldsymbol{r}_c^0 = \boldsymbol{b}$.

Initial support: $S_c^0 = \emptyset$.

Initial candidate queue: $Q^0 = \{S_1^0, S_2^0 \cdots S_N^0\}$

Main iteration:

for $k = 1$ to I

for $c = 1$ to N

1. Compute $P(j|S_c^{k-1})$ according to equation (4.1)
2. Compute error: $\epsilon(j) = \min_{z_j}\left\|\boldsymbol{a}_j z_j - \boldsymbol{r}_c^{k-1}\right\|_2^2 - \alpha \cdot \log\left(P(j|S_c^{k-1})\right)$ all j using the optimal choice $z_j^* = \boldsymbol{a}_j^T \boldsymbol{r}_c^{k-1} / \left\|\boldsymbol{a}_j\right\|_2^2$.
3. Find top N minimizers of $\epsilon(j)$ to form $J = \{j_1, j_2 \cdots j_N\}$ such that $J \cap S_c^{k-1} = \emptyset$, and push $S^k = \left\{S_c^{k-1} \cup \{j_1\}, S_c^{k-1} \cup \{j_2\} \cdots S_c^{k-1} \cup \{j_N\}\right\}$ into Q^k

end

for each $S_i^k \in Q^k$

- Compute $\boldsymbol{x}$ that minimizes $\|\boldsymbol{Ax} - \boldsymbol{b}\|_2^2$ subject to $support\{\boldsymbol{x}\} = S_i^k, \boldsymbol{x} \geq \boldsymbol{0}$.
- Compute residual $\boldsymbol{r}_i^k = \boldsymbol{b} - \boldsymbol{Ax}$.

end

Ascendingly sort Q^k according to $\left\|\boldsymbol{r}_i^k\right\|_2^2 - \alpha \cdot \log\left(P\left(S_i^k\right)\right)$ and keep the first N items.

If any $\left\|\boldsymbol{r}_i^k\right\|_2^2 < \epsilon_0$ break.

end

Output result: $Q^* = Q^k$.

We thus apply transition probabilities to model relations between two decomposition candidates in adjacent frames. To formulate note transitions, Bayes model is adopted so that conditional probability can be approximated from individual note pairs. Since at most N candidates remain in one frame the posterior probability of candidate j in frame t given candidate i in frame $t-1$ is calculated as

$$P\left(\boldsymbol{X}_j^{(t)}\middle|\boldsymbol{X}_i^{(t-1)}\right) \stackrel{\text{def}}{=} \frac{\prod_{k=1}^{D} P\left(X_{k,j}^{(t)}\middle|\boldsymbol{X}_i^{(t-1)}\right)}{\sum_l \prod_{k=1}^{D} P\left(X_{k,l}^{(t)}\middle|\boldsymbol{X}_i^{(t-1)}\right)} \tag{4.2}$$

where $\boldsymbol{X}_t^{(j)} = \{X_{1,t}^{(j)}, X_{2,t}^{(j)} \cdots X_{D,t}^{(j)}\}$ denotes the decomposition candidate j in frame t containing D notes. $P\ (X_t|\boldsymbol{X}_{t-1})$ is calculated similarly as in equation (4.1).

Thanks to the multiple decomposition candidates generated by the modified PCMP previously, an inter-decomposition directed graph is further constructed to help determining the optimal decomposition path through all frames, as illustrated in Figure 4.5. In this directed graph, each decomposition candidate forms a node and outgoing edge denotes the transition probability computed by equation (4.2). The nodes are disconnected within the same frame indexed with t.

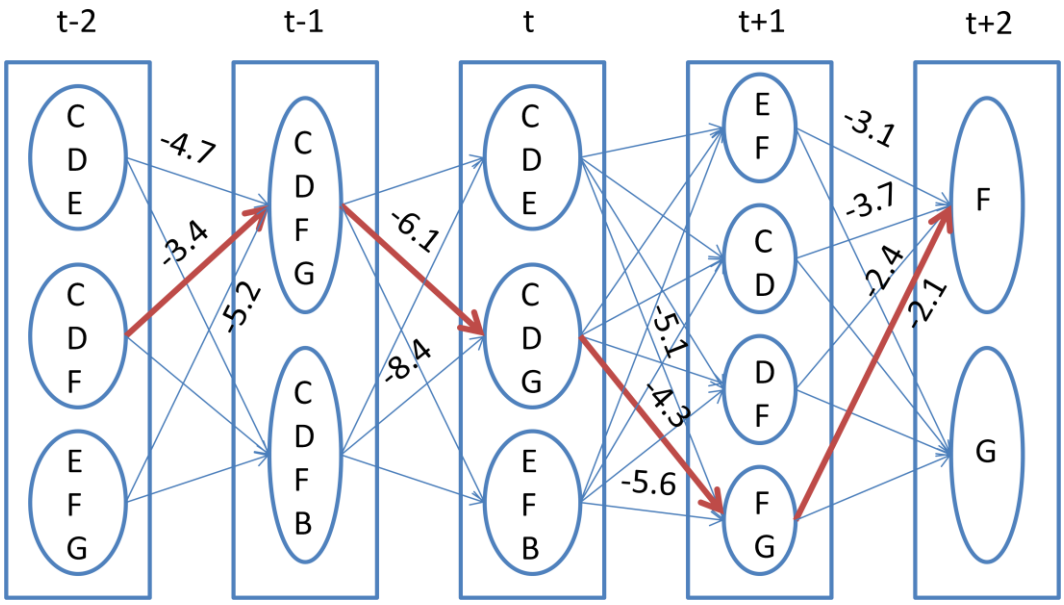


Figure 4.5: Optimal path decoding example

In order to connect transition probabilities with the sparse decomposition candidates, the decomposition goodness measure is converted into corresponding probabilities as $P(\boldsymbol{X})^{\alpha} \cdot \exp(-\|\boldsymbol{r}\|_2^2)$, since the frame signal $\boldsymbol{b}$ and atoms in $\boldsymbol{A}$ have been normalized to unit vectors. The conversion also reflects a reasonable assumption that the reconstructed signal is approximately Gaussian distributed around original one. Treating the decomposition candidates as hidden states of a first order Hidden Markov Model (HMM), Viterbi algorithm decodes the optimal decomposition candidate path $\boldsymbol{p}^*$:

$$\boldsymbol{p}^* = \underset{\boldsymbol{p}}{\operatorname{argmax}} \prod_{t=1}^{F} P(\boldsymbol{X}_{p_t}^{(t)})^{\alpha} \, P\left(\boldsymbol{X}_{p_t}^{(t)} \middle| \boldsymbol{X}_{p_{t-1}}^{(t-1)}\right)^{\beta} e^{-\left\|\boldsymbol{r}_{p_t}^{(t)}\right\|_2^2}$$

where t is the frame index, F is the total number of frames, β is a balance parameter to adjust emphasis, p_t denotes decomposition candidate index in frame t along path $\boldsymbol{p}$. Initially $P(\boldsymbol{X}_{p_1}^{(1)} | \boldsymbol{X}_{p_0}^{(1)})$ =1. A similar study on emotion recognition from audio signal is presented in [Meng et. al 2011] in which the hidden states of the HMMs are associated with the levels of affective dimensions to convert the classification problem into a best path finding problem in HMM.

4.3.3 Experiment and results

To evaluate the decomposition quality of the proposed PCMP with note statistics, a multi-timbral music with time domain note reference has been used, which is provided in Mirex2007 multiF0 development data [Mirex2007 2007]. The music is a recording of the fifth variation from *L. van Beethoven Variations from String Quartet Op.18 N.5,* lasting for 54s. 5 instruments are included into the music. Each instrument was recorded separately and then mixed to a mono 44.1 kHz 16 bits wave file. The whole music is tested by our system (PCMP with multi-candidate and Viterbi) against its ground truth MIDI file.

Another widely used dataset adopted in our experiments is MUS, provided in MAPS [Emiya *et al.* 2010]. MUS contains 270 pieces of classical and traditional music, recorded in different conditions which vary in piano instruments and surroundings. For each piano music piece, as in [Lewandowski *et al.* 2012], first 30 seconds are tested by our system against ground truth MIDI files.

Figure 4.6 shows precision and recall scatter diagram of the proposed decomposition that improve original PCMP, noted as PCMPMC and PCMPMCV. Table 4.15 displays the comparisons in terms of precision, recall and F-measure between our proposed method and the state of the art results. F-measure is defined as the harmonic mean of precision and recall. Statistic of note recognition precision and recall has been made upon consecutive 186ms frames. For ground truth MIDI, if 70% of some note lies in the frame the note is accounted and there is no frequency tolerance. Threshold for drawing the diagram is imposed on sparse solution vector in each frame to filter insignificant note detection according to its sparse solution value. Different thresholds result in scatter points in Figure 4.6. Two free parameters α and β are set to 0.8 and 1.3 to balance reconstruction error and note statistics.

Table 4.15: Average multiple pitch estimation performance on MIREX2007 dataset.

	Prec. (%)	Rec. (%)	F-meas. (%)
NMF[Raczynski *et al.* 2007]	41.1	46.6	45.3
HTC[Kameoka *et al.* 2007]	57.4	51.3	54.2
JHT[Wu *et al.* 2011]	**59.7**	61.4	**60.5**
PCMPMCV	51.8	**72.0**	60.3

From Figure 4.6 we can see that when co-occurrence note information is integrated into PCMP, the precision increases about 6% while recall increases by 2%~3%. When the note transition information is fused and the optimal path decoding is applied, the precision and recall are further improved by 5% and 2% approximately. From Table 4.15 and Figure 4.6 we can find that if no threshold is imposed on the sparse solution of PCMPMCV, 72% of the notes can be recalled while the precision is 51.8% resulting in an F-measure of 60.3 %. The recall of our best configuration outperforms state of the art result in [Wu *et al.* 2011] by

more than 10% while the precision is 8% lower, resulting in an F-measure 0.2% lower than that reported in [Wu *et al.* 2011].

Table 4.16: Average multiple pitch estimation performance on MUS dataset.

	Prec.(%)	Rec. (%)	F-meas.(%)
Spectral constraints [Vincent *et al.* 2010]	71.6	65.5	67.0
Isolated note spectra [Vincent *et al.* 2010]	68.6	66.7	66.0
DNMF-LV [Raczynski *et al.* 2012]	68.1	65.9	66.9
DNMF-AE [Raczynski *et al.* 2012]	66.8	68.7	67.8
SONIC [Marolt 2004]	**74.5**	57.6	63.6
PCMPMCV	60.7	**77.3**	**68.0**

Table 4.16 shows the precision, recall and F-measure results on MUS data set. All parameters and setups are the same as used in previous experiment except for the conditional probability estimation. In this experiment the rest data other than first 30 seconds are used to estimate conditional probabilities $P(X|Y)$. From Table 4.16 we can observe that the proposed approach achieves the highest recall and F-measure of 77.3% and 68%, although obtains the lowest precision of 60.7%.

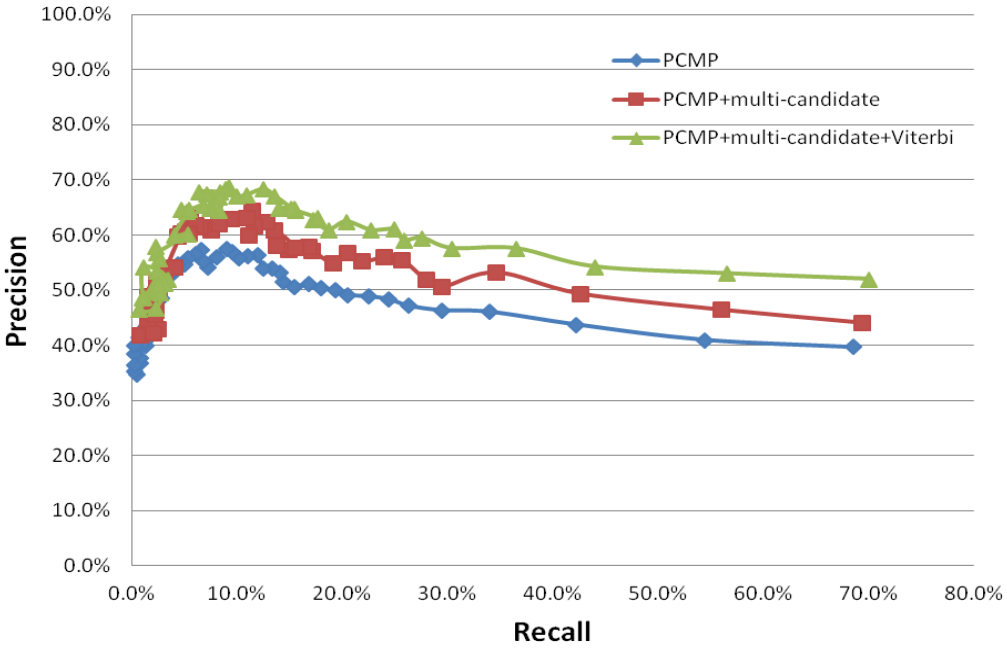


Figure 4.6: Note precision vs. recall of the two improvements.

From the two experiments, we can find that with statistical musical knowledge sparse decomposition is improved in terms of both precision and recall. The proposed approach tends to obtain superior recall and F-measures but lower precisions compared with variant NMF and other methods. Higher recall means the more information is preserved in the decomposition results. Since our final aim of the decomposition is to provide decent features for music classifications, the performance of our system is actually preferred. Our higher recalls and F-measures are attributed to the quality of MIDI dictionary as well as statistical music knowledge fused in sparse decomposition. Longer analysis window is another important factor.

When comparing decomposition with the ground truth, we found numbers of instrument errors even with correct note detections, which is likely caused by mismatches between the MIDI dictionary and the real-world data. In some cases concurrent and transition probability of notes can even make incorrect compensation to original PCMP, which is probably due to the limitation of the naive Bayes model. To overcome these drawbacks, dictionary adaptation

techniques and sophisticated graphical models will be proposed and investigated in our future work.

To test our PCMP based mid-level feature for content-based music classification, we have conducted experiments on previous mentioned data sets. The results are shown in following tables. From Table 4.17, compared with PCMP result, the multiple candidates improve the average accuracy by 0.7%. Table 4.18 shows that optimal path gain additional 0.5% of accuracy compared with multiple candidates situation. When low level information is added, as shown in Table 4.19, the overall performance is increased by 3.6%. Tables 4.20 to 4.22 demonstrate the similar results on APM data set. Compared with basic PCMP results, from Table 4.20 and 4.21, we can find that the two innovations of PCMP achieve 2% and 3% improvement. The Table 4.22 shows that the best performance reached is 80.2% when GSV is further fused.

Table 4.17: PCMP+multiple candidate confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset.

%	Anx	Con	Dep	Exu
Anx	88.7	2.3	1.5	7.5
Con	0.8	96.1	1.3	1.8
Dep	2.2	4.9	90.3	2.4
Exu	4.6	5.2	2.7	87.5
Average	90.7			

Table 4.18: PCMP+multiple candidates+optimal path confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset.

%	Anx	Con	Dep	Exu
Anx	89.3	2.3	1.3	7.1
Con	0.8	96.2	1.3	1.7
Dep	2.0	4.9	91.0	2.1
Exu	4.4	4.8	2.7	88.1
Average	91.2			

Table 4.19: GSV fused with PCMP+multiple candidates+optimal path confusion matrix (actual classes in rows, predicted classes in columns) using the classical dataset.

%	Anx	Con	Dep	Exu
Anx	92.9	0.3	1.0	5.8
Con	0.2	96.7	3.0	0.1
Dep	1.2	2.8	94.9	1.1
Exu	4.0	1.1	0.2	94.7
Average	94.8			

Table 4.20 PCMP+multiple candidates confusion matrix (actual classes in rows, predicted classes in columns) using the APM dataset.

%	Fear	Happy	Relax	Sad
Fear	63.1	11.9	7.8	17.2
Happy	8.5	68.0	17.2	6.3
Relax	7.6	18.1	51.0	23.3
Sad	10.0	5.6	18.5	65.9
Average	62.0			

Table 4.21: PCMP+multiple candidates+optimal path confusion matrix (actual classes in rows, predicted classes in columns) using the APM dataset.

%	Fear	Happy	Relax	Sad
Fear	64.3	11.5	7.6	16.6
Happy	8.3	69.2	16.5	6.0
Relax	7.2	18.1	51.8	22.9
Sad	10.0	5.1	18.1	66.8
Average	63.0			

Table 4.22: GSV fused with PCMP+multiple candidates+optimal path confusion matrix (actual classes in rows, predicted classes in columns) using the APM dataset.

%	Fear	Happy	Relax	Sad
Fear	83.4	3.6	1.3	11.7
Happy	4.7	85.7	7.0	2.6
Relax	6.1	9.6	72.6	11.7
Sad	16.3	0.4	4.4	78.9
Average	80.2			

We also tested our PCMP based feature on two public datasets: "Now That's What I Call Music" proposed in [Shuller *et al.* 2010] for mood classification and the famous GTZAN data set for genre classification. Integrated with Xiao's and APM data set, the overall results are shown in Table 4.23. From Table 4.23 we can observe that although PCMP based features cannot beat the state-of-the-art in [Shuller *et al.* 2010] and [Panagakis *et al.* 2009], they have demonstrated the ability to compensate the drawback of low level information by increasing classification accuracy by around 3%.

Table 4.23: Results on all data sets.

Accuracy (%)	Mood			Genre
	Xiao	APM	NTWICM (V/A)	GTZAN
Xiao [Xiao *et al.* 2008b]	82.95	N/A	N/A	N/A
Shuller [Shuller *et al.* 2010]	N/A	N/A	61.0/58.7	N/A
Topology preserving NTF +SRC[Panagakis *et al.* 2009]	N/A	N/A	N/A	93.7
BoW(GMM)	87.3	N/A	55.9/50.3	78.7
GSV	88.3	76.8	56.7/51.8	80.5
OpenSmile	84.0	74.9	N/A	N/A
PCMP	90.0	59.4	48.3/41.5	78.3
OpenSmile+PCMP	93.7	77.4	N/A	N/A
GSV+PCMP	94.4	78.9	N/A	N/A
PCMP+multiple candidates	90.7	62.0	N/A	N/A
PCMP+multiple candidates+optimal path	91.2	63.0	49.6/44.1	79.5
GSV+PCMP+multiple candidates +optimal path	94.8	80.15	56.9/52.3	83.2

4.4. Conclusion

We present in this chapter our novel music representation that aims at gaining an in-depth understanding of the music structure and harmony. It is obtained by decomposing sparsely the music onto a dictionary made of musical words that are a representation of the notes played instruments obtained from a MIDI synthesizer. Experiments have shown that our approach has the ability to decompose music into notes and the musical feature derived from this decomposition allows improving the music mood classification. Indeed, it has

outperformed openSMILE low-level features GSV for the classical music and improved the classification accuracy when combined with openSMILE and GSV for the APM data set which is more complex particularly due to the high variability of the instruments used.

To further improve decomposition accuracy, we have incorporated music knowledge which concern note statistical information into our sparse decomposition. In the frame level, music signals are decomposed onto a MIDI dictionary with a note co-occurrence heuristic. Transition probabilities are then computed between adjacent decomposition candidates through the whole frame sequence. The final optimal decomposition path is then constructed by the Viterbi algorithm. Experimental results show that both embedding concurrent note statistics in PCMP and applying a note sequence heuristic improve the note recognition precision and recall. Our proposed PCMP with optimal path search outperforms the state-of-the-art methods in terms of recall and f-measure. When fused with low level features our note histogram features increase the mood and genre classification accuracy.

Chapter 5: Conclusions and Future Work

5.1 Conclusion

In this thesis, we focus on using low level and mid level feature to perform semantic analysis of music in particular mood and genre classification. Regarding low level features, although they contain complete signal related information, their direct use for semantic analysis is hardly conceivable since too much signal details and redundancies prevent the semantics concepts from being efficiently detected. Thus, low level feature modeling techniques have to be applied to delve semantic information from the noisy features. K-means and GMM based BoW as well as Gaussian super vector methods have demonstrated their effectiveness in modeling low level features, their principle being to transform redundant signal features into "words" that express semantic meanings. However, the trend of "big data" challenges the efficiency of the modeling methods, particularly k-means, GMM and MAP procedures, which are the computational bottle necks. Therefore, we have proposed to use matrix format which can be effectively accelerated on GPU multi-core CPU and cluster to implement bag-of-words and GMM super vector method. To employ high performance matrix operation library of ACML, ATLAS and CUBLAS we reformulate k-means and EM algorithm into matrix multiplications, which is also very concise to implement in other languages like MATLAB and FORTRAN. Experiments on music genre and mood classification tasks show that the proposed implementations achieve 38 to 209 times acceleration, compared with single threaded CPU version. 240-cored GPU can run up to 5 times faster than 4 8-cored CPUs with just less 1% performance lost.

From experimental results we can find detailed improvement that high performance library based implementations proposed achieve up to 5 times faster than multi-threading based implementation (Yael). We also observe that GPU based implementation executes 5 times faster than multi-core CPU based one on GMM training whereas multi-core version outperforms GPU on k-means clustering by 22% in terms of speed. GPU based k-means is slower than multi-core CPU in that CPU version executes $O(ND)$ operations for $\boldsymbol{C}^{blk}$ calculation while

GPU version actually executes matrix multiplication containing $O(NDM)$ operations. We choose matrix multiplication for $\boldsymbol{C}^{blk}$ calculation in GPU because $O(ND)$ operations consume as twice time as $O(NDM)$ matrix multiplication on GPU architecture. This abnormal phenomenon is due to random memory access pattern of $O(ND)$ operations on GPU. Therefore in this scenario matrix multiplication is the optimal but still slower way for GPU.

Comparing with the state-of-the-art result in [Machlica *et al.* 2011] that only recorded kernel function's running time without data IO and data preparation duration, our method consumes only 7 seconds compared with 9.1 in their paper. From genre classification result, we can find that the quality of dictionary trained by CPU is little better than GPU by less than 1% in terms of average classification accuracy. This is due to the less floating point error of CPU and double precision floating point numbers used during summation procedure on CPU. Last but not the least, we can find that with the same number of mixture, the same types of GPU and comparable data scale our CUBLAS based implementation is 10 times faster than CUDA kernel implementation described. In mood classification phase, we can find that 1024 dimensional bag-of-words histogram features outperforms the standard 6552 dimensional OpenSMILE emotional features by 3% in terms of average classification accuracy.

On Hadoop and Spark cluster when optimal configuration is set the proposed method still achieves 10 and 5 speed-up, compared with the state-of-the-art libraries of Mahout and MLLib.

Regarding mid-level features, previous work in literatures seldom uses note information which is in fact among the most natural semantic given by the composer. The difficulties of decomposing music into note combination are attributed to the complexity of music signal, in which sound of different instruments entangle. With the help of a realistic instrument sound library, we have proposed our novel music representation feature that aims at gaining an in-depth understanding of the music structure and harmony. We decompose music sparsely onto a dictionary made of musical words. The musical words are a

representation of the notes played instruments obtained from a MIDI synthesizer. Experiments have shown that our approach has the ability to recover music notes meanwhile the musical feature derived from this decomposition allows improving the music mood classification. Indeed, it has outperformed openSMILE low-level features and GSV for the classical music, and improved the classification accuracy when combined with openSMILE and GSV for APM data set which is more complex particularly due to the high variability of the instruments used.

To further improve decomposition accuracy, we have proposed to incorporate music knowledge which concern note statistical information to improve sparse decomposition. In the frame level, music signals are decomposed onto a MIDI dictionary with a note co-occurrence heuristic. Transition probabilities are then computed between adjacent decomposition candidates through the whole frame sequence. The final optimal decomposition path is then constructed by the Viterbi algorithm.

From the experiments, we can find that with statistical musical knowledge sparse decomposition is improved in terms of both precision and recall. The proposed approach tends to obtain superior recall and F-measures but lower precisions compared with variant NMF and other methods. Higher recall means the more information is preserved in the decomposition results. For example we can find that when co-occurrence note information is integrated into PCMP, the precision increases about 6% while recall increases by 2%~3%. When the note transition information is fused and the optimal path decoding is applied, the precision and recall are further improved by 5% and 2% approximately. From MIREX2007 dataset we can find that if no threshold is imposed on the sparse solution of PCMPMCV, 72% of the notes can be recalled while the precision is 51.8% resulting in an F-measure of 60.3 %. The recall of our best configuration outperforms state of the art result by more than 10% while the precision is 8% lower. From another larger dataset MUS we can observe that the proposed approach still achieves the superior result compared with the state-of-the-art. We obtained the highest recall and F-measure of 77.3% and 68%, although obtains the lowest precision of 60.7%.

Since our final aim of the decomposition is to provide decent features for music classifications, the performance of our system is actually preferred. Our higher recalls and F-measures are attributed to the quality of MIDI dictionary as well as statistical music knowledge fused in sparse decomposition. Longer analysis window is another important factor.

From classical music dataset we can find that the dictionary built with high quality instrument waves (Logic Pro 9) outperforms Microsoft wavetable synthesizer. Our proposed histogram musical feature also beat the openSMILE features. Regarding to individual feature, we can find that the musical histogram feature generated from Logic Pro 9 MIDI dictionary outperforms the others by 2% to 7% on average accuracy. For each individual mood, Logic Pro 9 MIDI dictionary also surpasses the others. From APM music dataset we can find that compared with the classical music set the overall performance of musical histogram feature, openSMILE and GSV decline by 30%, 10% and 11.5%. OpenSMILE feature set outperforms Logic Pro 9 musical histogram feature by 15.5% and GSV surpass musical histogram feature by 17.4% However, when these two levels of feature sets are combined in the late fusion, the average accuracy increases by 2.5% and 2.1%. The musical histogram feature is less effective with this modern music dataset than with the classical music dataset. The reason certainly lies in the MIDI dictionary that may have difficulties to accurately decompose this music due to its important complexity and the presence of instruments in music that are missing in the dictionary. This problem can be solved by using a MIDI synthesizer producing more realistic instrument sounds. Even though openSMILE feature set provides very complete low-level information characterizing temporal and frequential signal properties, its performance for classification can be improved by enriching this description with higher level information that we propose with our musical histogram feature which allows to explicit the music content.

To sum up, embedding concurrent note statistics in PCMP and applying a note sequence heuristic allows improving the note recognition precision and recall. Our proposed PCMP with optimal path search outperforms the state-of-the-art

methods. Our proposed mid level histogram achieve the best results on classical music. Although inferior to the low level feature performance our proposed feature provide compensated information for mood classification on modern music dataset.

5.2 Perspective and Future Work

We plan to perform the following improvements to our current system. In the low level feature modeling acceleration part, we plan to install GPU on the Hadoop and Spark cluster so that the overall computation can be parallelized even between CPUs and GPUs. The challenge will be here to coordinate data consumption between CPUs and GPUs since GPU executes faster.

In the note decomposition part, we plan to explore dictionary self-adaptation techniques, since even if Logic Pro can generate realistic instrument sound, the dictionary is not universally compatible. There can be many factors like recording distortion that affect the accuracy of note decomposition. If the MIDI note dictionary can be effectively adapted to the input signal, not only the note recovery accuracy but the final classification performance can be further improved.

Up to now, our system neglects high level information which concerns human natural language. Indeed, lyrics can represent interesting semantic information particularly related to emotions. Therefore another important direction of improvement is to employ natural language processing techniques to represent lyrics information that could be combined with our low and middle level audio features for music mood and genre classification.

Publications

- **Boyang Gao**, Emmanuel Dellandréa, Liming Chen, "Accelerate K-means,GMM and MAP Computation with GPU, Multi-core CPU and Computer Cluster" **submitted** to *IEEE Transaction on Multimedia*
- Xiaofang Wang, **Boyang Gao**, Simon Masnou, and Liming Chen, "Active Colloids Tracking: Recover Trajectories Globally via Min-cost/max flow" **submitted** to *IEEE Transaction on Image Processing*
- **Boyang Gao**, Emmanuel Dellandréa, Liming Chen, "Sparse Music Decomposition onto a Midi Dictionary Driven by Statistical Music Knowledge", in *Proc.* 14th International Society for Music Information Retrieval Conference (ISMIR), 2013
- Nicolas Ballas, … **Boyang Gao**, Emmanuel Dellandréa, Liming Chen …, "IRIM at TRECVID 2012: Semantic Indexing and Instance Search," in *Proc. TRECVID 2012 workshop*, 2012
- **Boyang Gao**, Emmanuel Dellandréa, Liming Chen, "Accelerate Bag-of-Words Method With GPU And Multi-Core CPU And Its Application to Music Classification" in *Proc. to the 11th International Conference on Signal Processing (ICSP)*, 2012
- **Boyang Gao**, Emmanuel Dellandréa, Liming Chen, "Music Sparse Decomposition onto a MIDI Dictionary of Musical Words and Its Application to Music Mood Classification," in *Proc. 10th Workshop on Content-Based Multimedia Indexing (CBMI)*, 2012
- Chao Zhu, **Boyang Gao**, Charles-Edmond Bichot, Emmanuel Dellandréa, Liming Chen, Ningning Liu, Yu Zhang, "ECL-LIRIS at TRECVID 2011: Semantic Indexing", 2011 TRECVID Workshop, 2011, Washington
- Qian Yao, Wu Zhi-Zheng, **Gao Bo-Yang** and Soong Frank K., "Improved Prosody Generation by Maximizing Joint Probability of State and Longer Units", *IEEE Transactions on Audio, Speech and Language Processing*, Vol 19, Issue 6, pp. 1702-1710, Aug. 2011

Bibliography

[Josef 2009] Sivic, Josef (April 2009). "Efficient visual search of videos cast as text retrieval". *IEEE Transactions On Pattern Analysis And Machine Intelligence*, Vol. 31, No. 4. IEEE. pp. 591–605.

[Liu *et al.* 2013] Ningning Liu, Emmanuel Dellandréa, Chao Zhu, Yu Zhang, Charles-Edmond Bichot, Stéphane Bres, Bruno Tellez, Liming Chen, "Multimodal Recognition of Visual Concepts using Histograms of textual Concepts and Selective Weighted Late Fusion Scheme", *Computer Vision and Image Understanding (CVIU)* 117(5):493-512 (2013)

[Kong *et al.* 2010] J. Kong, "GPU accelerated face detection", *Proceeding of Intelligent Control and Information Processing (ICICIP)*, pp.584-588, 2010.

[van de Sande *et al.* 2011] K. van de Sande and T. Gevers, "Empowering visual categorization with the GPU," *IEEE Transactions on Multimedia*, Volume 13 (1), page 60-70, 2011.

[Kumar *et al.* 2009] N. Kumar, S. Satoor and I. Buck, "Fast parallel Expectation Maximization for Gaussian Mixture Models on GPUs using CUDA," *Proceeding of 11th IEEE International Conference on High Performance Computing and Communications*, 2009.

[Pangborn *et al.* 2010] A. D. Pangborn, "Scalable data clustering using GPUs," Masters thesis, Rochester Institute of Technology, 2010.

[Azhari *et al.* 2011] M. Azhari and C. Ergün, "Fast Universal Background Model (UBM) Training on GPUs using Compute Unified Device Architecture (CUDA)," *International Journal of Electrical & Computer Sciences IJECS-IJENS*, vol. 11, no. 4, pp. 49-55, 2011.

[Gonina 2011] E. Gonina, "Fast Speaker Diarization Using a Specialization Framework for Gaussian Mixture Model Training," master thesis, University of California Berkeley, 2011.

[Vanek *et al.* 2011] L. Machlica, J. Vanek, and Z. Zajic, "Fast Estimation of Gaussian Mixture Model Parameters on GPU Using CUDA", *Proceeding of*

12th International Conference on Parallel and Distributed Computing Applications and Technologies, no. 1, pp. 167-172, 2011.

[Wu *et al.* 2012] K. Wu, Y. Song and L. Dai, "CUDA-Based Fast GMM Model Training Method and Its Application," *Journal of Data Acquisition & Processing*, vol. 27, no. 1, pp. 85-90, 2012.

[Tzanetakis *et al.* 2002] G. Tzanetakis and P. Cook, "Music genre classification ofaudio signals", *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[Xiao *et al.* 2008] Z. Xiao, E. Dellandréa, W. Dou and L. Chen, "What is the best segment duration for music mood analysis ?," in *Proc. International Workshop on Content-Based Multimedia Indexing (CBMI)*, London, 2008.

[Eyben *et al.* 2010] F. Eyben, M. Wöllmer and B. Schulle, "OpenSMILE the munich versatile and fast open-source audio feature extractor," in *Proc. ACM Multimedia (MM)*, Florence, 2010.

[GoogleBlog 2010] http://googleblog.blogspot.fr/2010/07/ooh-ahh-google-images-presents-nicer.html

[Campbell *et al.* 2006] Campbell W M, Sturim D E, Reynolds D A. Support vector machines using GMM supervectors for speaker verification[J]. *Signal Processing Letters*, IEEE, 2006, 13(5): 308-311.

[Inoue *et al.* 2011] Inoue N, Shinoda K. A fast MAP adaptation technique for GMM-supervector-based video semantic indexing systems[C] in *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 2011: 1357-1360.

[White 2010] White, Tom. *Hadoop: The Definitive Guide*. O'Reilly Media, 2009.

[Konstantin *et al.* 2010] Shvachko, Konstantin, *et al.* "The hadoop distributed file system." In *Proceedings of Symposium on Mass Storage Systems and Technologies (MSST)*, 2010 IEEE 26th. IEEE, 2010.

[Zaharia *et al.* 2010] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010, June). Spark: cluster computing with working sets. In

Proceedings of the 2nd USENIX conference on Hot topics in cloud computing (pp. 10-10).

[Zaharia *et al.* 2012] Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Stoica, I. (2012, April). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation* (pp. 2-2). USENIX Association.

[Dean *et al.* 2008] Dean, J., Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1), 107-113.

[Katsavounidis *et al.* 1994] I. Katsavounidis, C. Kuo, and Z. Zhang. A new initialization technique for generalized Lloyd iteration. *IEEE Signal Processing Letters*,1(10):144–146, 1994.

[He *et al.* 2004] He, J., Lan, M., Tan, C. L., Sung, S. Y., & Low, H. B. (2004, July). Initialization of cluster refinement algorithms: A review and comparative study 2004. *Proceedings.of 2004 IEEE International Joint Conference on . Neural Networks*, (Vol. 1). IEEE.

[Björn *et al.* 2010] Björn, S., Johannes, D., Gerhard, R. (2010). Determination of nonprototypical valence and arousal in popular music: features and performances. *EURASIP Journal on Audio, Speech, and Music Processing*, 2010.

[Anil *et al.* 2011] Anil, R., Dunning, T., Friedman, E. (2011). *Mahout in action* . Manning.

[Bahmani et al 2012] Bahmani, B., Moseley, B., Vattani, A., Kumar, R., Vassilvitskii, S. (2012). Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7), 622-633.Chicago

[Yahoo 2014] http://developer.yahoo.com/hadoop/tutorial/module1.html

[Borthakur 2007] Borthakur, D. (2007). *The hadoop distributed file system: Architecture and design*. Hadoop Project Website.

[Hartigan 1979] Hartigan, John A., and Manchek A. Wong. "Algorithm AS 136: A k-means clustering algorithm." *Applied statistics (1979)*: 100-108.

[Dempster *et al.* 1977] Dempster, Arthur P., 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) (1977): 1-38.

[Kim *et al.* 2010] Y. E. Kim, E. M. Schmidt, R. Migneco, B. G. Morton, P. Richardson, J. Scott, J. A. Speck and D. Turnbull, "Music emotion recognition: a state of the art review," in *Proc. 11th International Society for Music Information Retrieval Conference*, Utrecht, 2010.

[Pratt 1952] C. C. Pratt, *Music as the language of emotion*, Oxford: The Library of Congress, 1952.

[Lim *et al.* 2011] S.-C. Lim, S.-J. Jang, S.-P. Lee and M. Y. Kim, "Music genre/mood classification using a feature-based modulation spectrum," in *Proc. International Conference on Mobile IT Convergence (ICMIC)*, Gumi, 2011.

[Wang *et al.* 2010] J.-C. Wang, H.-Y. Lo, S.-K. Jeng and H.-M. Wang, "MIREX 2010: audio classification using semantic transformation and classifier ensemble," in *Music Information Retrieval Evaluation eXchange (MIREX)*, 2010.

[Cao *et al.* 2009] C. Cao and M. Li, "THINKIT's submissions for MIREX2009 audio music classification and similarity tasks," in *Music Information Retrieval Evaluation eXchange (MIREX)*, 2009.

[Peeters *et al.* 2009] G. Peeters, "MIREX-09 "music mood, mixed-genre, latin-genre and classical composer classification" tasks: ircamclassification08 submission," in *Music Information Retrieval Evaluation eXchange (MIREX)*, 2009.

[Russell 1980] J. A. Russell, "A circumplex model of affect," *Journal of Personality and Social Psychology*, vol. 39(6), pp. 1161-1178, 1980.

[Shuller *et al.* 2010] B. Schuller, J. Dorfner and G. Rigoll, "Determination of nonprototypical valence and arousal in popular music: features and performances," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2010, 2010.

[Yang *et al.* 2008] Y.-H. Yang, Y.-C. Lin, Y.-F. Su and H. Chen, "A regression approach to music emotion recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 448 457, 2008.

[Elad 2010] M. Elad, *Sparse and Redundant Representations*, New York: Springer, 2010.

[Leveau et al 2008] P. Leveau, E. Vincent, G. Richard and L. Daudet, "Instrument-specific harmonic atoms for mid-level music representation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 116-128, 2008.

[Xiao *et al.* 2008] Z. Xiao, E. Dellandréa, W. Dou and L. Chen, "What is the best segment duration for music mood analysis ?," in *Proc. International Workshop on Content-Based Multimedia Indexing (CBMI)*, London, 2008.

[Xiao 2008] Z. Xiao, "Classification of emotion in audio signals," *Doctor's thesis of Ecole Centrale de Lyon*, 2008.

[Eyben *et al.* 2010] F. Eyben, M. Wöllmer and B. Schulle, "OpenSMILE the munich versatile and fast open-source audio feature extractor," in *Proc. ACM Multimedia (MM)*, Florence, 2010.

[Kameoka *et al.* 2007] H. Kameoka, T. Nishimoto, and S. Sagayama, "A multi-pitch analyzer based on harmonic temporal structured clustering," *IEEE Transactions on Audio, Speech, and Language Processing* , vol. 15, no. 3, pp. 982–994, 2007.

[Wu *et al.* 2011] J. Wu, E. Vincent, S. A. Raczynski, T. Nishimoto, N. Ono, and S. Sagayama, "Multipitch estimation by joint modeling of harmonic and transient sounds," in *Proceedings of 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 25–28, 2011.

[Lee *et al.* 2001] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in neural information processing systems*, vol. 13, pp. 556-562, 2001.

[Raczynski *et al.* 2007] S. A. Raczynski, N. Ono, and S. Sagayama, "Multipitch analysis with harmonic nonnegative matrix approximation," in *Proceedings of 8th International Conference on Music Information Retrieval (ISMIR)*, 2007.

[Hoyer *et al.* 2002] P. O. Hoyer, "Non-negative sparse coding," in *Proceedings of 12th IEEE Workshop on Neural Networks for Signal Processing*, pp. 557–565, 2002.

[Zafeiriou *et al.* 2006] S. Zafeiriou, A. Tefas, I. Buciu, and I. Pitas, "Exploiting discriminant information in nonnegative matrix factorization with application to frontal face verification," *IEEE Transactions on Neural Networks*, vol. 17, no. 3, pp. 683–695, 2006.

[Nvidia 2014] http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

[Guan *et al.* 2011] N. Guan, D. Tao, Z. Luo and B. Yuan, "Manifold regularized discriminative nonnegative matrix factorization with fast gradient descent," *IEEE Transactions on Image Processing*, vol. 20, no. 7, pp. 2030–2048, 2011.

[Wang *et al.* 2004] Y. Wang, Y. Jia, C. Hu and M. Turk, "Fisher non-negative matrix factorization for learning local features," in *Proceedings of Asian Conference on Computer Vision (ACCV)*, 2004.

[Boulanger-Lewandowski *et al.* 2012] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Discriminative non-negative matrix factorization for multiple pitch estimation," in *Proceedings of 13th International Conference on Music Information Retrieval (ISMIR)*, 2012

[Sakaue *et al.* 2012] D. Sakaue, T. Otsuka, K. Itoyama, and H.G. Okuno, "Bayesian non-negative harmonic-temporal factorization and its application to multipitch analysis," in *Proceedings of 13th International Conference on Music Information Retrieval (ISMIR)*, 2012.

[Gao *et al.* 2012] B. Gao, E. Dellandréa, and L. Chen, "Music sparse decomposition onto a midi dictionary of musical words and its application to music mood classification," in *Proceedings of 10th IEEE International Workshop on Content-Based Multimedia Indexing (CBMI)*, pp. 1 – 6, 2012.

[Leveau *et al.* 2008] P. Leveau, E. Vincent, G. Richard and L. Daudet, "Instrument-specific harmonic atoms for mid-level music representation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 116-128, 2008.

[Bruckstein *et al.* 2008] A.M. Bruckstein, M. Elad, and M. Zibulevsky, "Sparse non-negative solution of a linear system of equations is unique," in *Proceedings of 3rd IEEE International Symposium on Communications, Control and Signal Processing (ISCCSP)*, pp. 762–767, 2008.

[Chen *et al.* 2001] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

[Mirex 2007] http://www.music-ir.org/mirex/wiki/2007:Multiple_Fundamental_Frequency_Estimation_%26_Tracking

[Pati *et al.* 1993] Y. C. Pati, R. Rezaiifar, P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition." In *Proceedings of IEEE Signals, Systems and Computers*, vol. 1, pp. 40-44, 1993

[Emiya *et al.* 2010] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643-1654, 2010.

[Rath et al 2008] G. Rath, and C. Christine, "A complementary matching pursuit algorithm for sparse approximation," in *Proceedings of European Signal Process. Conf.*, Lausanne, Switzerland. 2008.

[Vincent *et al.* 2010] E. Vincent, N. Bertin, and R. Badeau, "Adaptive harmonic spectral decomposition for multiple pitch estimation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3:pp. 528–537, 2010.

[Marolt 2004] M. Marolt, “A connectionist approach to automatic transcription of polyphonic piano music,” *IEEE Transactions on Multimedia*, vol. 6, no. 3, pp. 439–449, 2004.

[Abarbanel *et al.* 1996] H. Abarbanel. *Analysis of Observed Chaotic Data*. Springer, New York, New York, 1996.

[Scheirer *et al.* 1997] E. Scheirer and M. Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1331-1334, Munich, Germany, Apr. 1997.

[Lu *et al.* 2001] G. Lu. Indexing and retrieval of audio: A survey. *Multimedia Tools and Applications*, 15(3):269-290, Dec. 2001.

[Xu *et al.* 2005] C. Xu, N.C. Maddage, and X. Shao. Automatic music classification and summarization. *IEEE Transactions on Speech and Audio Processing*, 13(3):441-450, May 2005.

[Khan *et al.* 2006] M.K.S. Khan and W.G. Al-Khatib. *Machine-learning based classification of speech and music. Multimedia Systems*, 12(1):55-67, Aug. 2006.

[Khan *et al.* 2004] M. Kashif Saeed Khan, Wasfi G. Al-Khatib, and Muhammad Moinuddin. Automatic classification of speech and music using neural networks. In MMDB 04: *Proceedings of the 2nd ACM international workshop on Multimedia databases*, pages 94-99. ACM Press, 2004.

[Jiang *et al.* 2005] H. Jiang, J. Bai, S. Zhang, and B. Xu. Svm-based audio scene classification. In Proceedings of the IEEE International Conference on Natural Language Processing and Knowledge Engineering, pages 131-136, Wuhan, China, Oct. 2005. IEEE.

[Tzanetakis 2002] G. Tzanetakis. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293-302, Jul. 2002.

[Tzanetakis 2005] G. Tzanetakis. Audio-based gender identification using bootstrapping. In *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 432433, Victoria, Canada, Aug. 2005. IEEE.

[Li *et al.* 2004] T. Li and M. Ogihara. Content-based music similarity search and emotion detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 705-708, Montreal, Quebec, Canada, May 2004. IEEE.

[Wang 2003] A. Wang. An industrial strength audio search algorithm. In *Proceedings of the International Conference on Music Information Retrieval*, pages 7-13, Baltimore, Maryland, Oct. 2003.

[Wang 2006] A. Wang. The shazam music recognition service. *Communications of the ACM*, 49(8):44-48, Aug. 2006.

[ANSI 1995] ANSI. Bioacoustical Terminology, ANSI S3.20-1995 (R2003). *American National Standards Institute*, New York, 1995.

[Hess 1983] W. Hess. Pitch determination of speech signals : *algorithms and devices*. Springer, Berlin, Germany, 1983.

[Bartsch *et al.* 2001] M.A. Bartsch and G.H. Wakefield. To catch a chorus: using chromabased representations for audio thumbnailing. In *Proceedings of the IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*, pages 15-18, New Platz, New York, Oct. 2001. IEEE

[Goto 2003] M. Goto. A chorus-section detecting method for musical audio signals. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 437-440, Hong Kong, China, Apr. 2003. IEEE.

[Zhu *et al.* 2006] Y. Zhu and M.S. Kankanhalli. Precise pitch profile feature extraction from musical audio for key detection. *IEEE Transactions on Multimedia*, 8(3):575-584, Jun. 2006.

[Agostini *et al.* 2001] G. Agostini, M. Longari, and E. Pollastri. Musical instrument timbres classification with spectral features. In *Proceedings of the IEEE Workshop on Multimedia Signal Processing*, pages 97-102, Cannes, France, Oct. 2001. IEEE.

[Cai *et al.* 2006] R. Cai, L. Lu, A. Hanjalic, H.J. Zhang, and L.H. Cai. A flexible framework for key audio effects detection and auditory context inference. *IEEE Transactions on Speech and Audio Processing*, 14:1026-1039, May 2006

[Bogert *et al.* 1963] B. Bogert, M. Healy, and J. Tukey. The quefrency alanysis of time series for echoes: cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe-cracking. In *Proceedings of the Symposium on Time Series Analysis*, pages 209-243. New York: Wiley, 1963

[Noll 1964] A.M. Noll. Short-time spectrum and cepstrum techniques for vocal pitch detection. *The Journal of the Acoustical Society of America*, 36(2), 1964.

[Bridle *et al.* 1974] J.S. Bridle and M.D. Brown. An experimental automatic word recognition system. JSRU Report No. 1003, Ruislip, *England: Joint Speech Research Unit*, 1974.

[Davis *et al.* 1980] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357-366, Aug. 1980.

[Xu *et al.* 2004] M. Xu, L. Duan, L. Chia, and C. Xu. Audio keyword generation for sports video analysis. In *Proceedings of the ACM International Conference on Multimedia*, pages 758-759, 2004.

[Wang *et al.* 2000] X. Wang, Y. Dong, J. Hakkinen, and O. Viikki. Noise robust chinese speech recognition using feature vector normalization and higher-order cepstral coefficients. *In Proceedings of the 5th International Conference on Signal Processing*, volume 2, pages 738-741, Aug. 2000

[Zwicker 1961] E. Zwicker. Subdivision of the audible frequency range into critical bands (frequenzgruppen). *The Journal of the Acoustical Society of America*, 33:248, 1961.

[Moore *et al.* 1990] C.J. Moore, R.W. Peters, and B.R. Glasberg. Auditory filter shapes at low center frequencies. *Journal of the Acoustical Society of America*, 88(1):132-140, 1990.

[Ogihara *et al.* 2003] T. Li, M. Ogihara, and Li Q. A comparative study on content-based music genre classification. In SIGIR 03: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 282-289, Toronto, Canada, 2003. ACM Press.

[Li *et al.* 2004] T. Li and M. Ogihara. Content-based music similarity search and emotion detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 705-708, Montreal, Quebec, Canada, May 2004. IEEE, IEEE.

[Li *et al.* 2006] T. Li and M. Ogihara. Toward intelligent music information retrieval. *IEEE Transactions on Multimedia*, 8(3):564-574, Jun. 2006.

[Umapathy *et al.* 2005] K. Umapathy, S. Krishnan, and S. Jimaa. Multigroup classification of audio signals using time-frequency parameters. IEEE *Transactions on Multimedia*, 7(2):308-315, Apr. 2005.

[Rabiner *et al.* 1978] L. Rabiner and R. Schafer. Digital Processing of Speech Signals. Prentice Hall Inc., Englewood Cliffs, New Jersey, 1978.

[Tremain 1982] T. Tremain. The government standard linear predictive coding algorithm: Lpc-10. *Speech Technology Magazine*, 1:40-49, Apr. 1982.

[Zwicker *et al.* 1999] E. Zwicker and H. Fastl. *Psychoacoustics: Facts and Models*. Springer, Berlin, Heidelberg, Germany, 2nd edition, 1999.

[Pampalk *et al.* 2002] E. Pampalk, A. Rauber, and D. Merkl. Content-based organization and visualization of music archives. In *Proceedings of the tenth*

ACM international conference on Multimedia, pages 570-579. ACM Press, 2002.

[Pfeiffer 1999] S. Pfeiffer. The importance of perceptive adaptation of sound features for audio content processing. In *Proceedings SPIE Conferences, Electronic Imaging Storage and Retrieval for Image and Video Databases VII*, pages 328-337, San Jose, California, Jan. 1999.

[Pfeiffer 2001] S. Pfeiffer. Pause concepts for audio segmentation at different semantic levels. In *Proceedings of the ACM International Conference on Multimedia*, pages 187-193, Ottawa, Canada, 2001. ACM Press.

[ISO-IEC 2002] ISO-IEC. Information Technology Multimedia Content Description Interface part 4: Audio. Number 15938. ISO/IEC, Moving Pictures Expert Group, 1st edition, 2002.

[Zhang *et al.* 2001] T. Zhang and C. C. J. Kuo. *Content-Based Audio Classifcation and Retrieval for Audiovisual Data Parsing.* Kluwer Academic Publishers, Boston, Massachusetts, 2001.

[Lloyd *et al.* 1982] Lloyd, Stuart P. (1982), "Least squares quantization in PCM", *IEEE Transactions on Information Theory* 28 (2): 129–137, doi:10.1109/TIT.1982.1056489.

[Reynolds *et al.* 2000] D. Reynolds, T.F. Quatieri, and R.B. Dunn, "Speaker Verification using Adapted Gaussian Mixture Models," *Digital Signal Processing*, vol. 10, pp. 19–41, 2000.

[Reynolds *et al.* 1990] Reynolds, D. A. Rose, R. C. and Smith, M. J. T., A Mixture Modeling Approach to Text-Independent Speaker Identification, *Journal of the Acoustical Society of America*, Suppl. 1, Vol. 87, p. 109, 1990.

[Wold *et al.* 1996] T. Wold, D. Blum, and J. Wheaton. Content-based classification, search, and retrieval of audio. *IEEE Multimedia*, 3(3):2736, 1996.

[Ramalingam *et al*. 2005] A. Ramalingam and S. Krishnan. Gaussian mixture modeling using short time fourier transform features for audio fingerprinting.

In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1146-1149, Amsterdam, The Netherlands, Jul. 2005. IEEE.

[Scheirer 1998] E. Scheirer. *Tempo and beat analysis of acoustic musical signals*. Joint Acoustic Society of America, 103(1):588-601, Jan. 1998.

[Tzanetakis 2002] G. Tzanetakis. Manipulation, analysis and retrieval systems for audio signals. *PhD. Thesis. Computer Science Department, Princeton University*, 2002.

[Foote 2000] J. Foote. Automatic audio segmentation using a measure of audio novelty. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, volume 1, pages 452-455, New York, NY, Aug. 2000. IEEE.

[Foote *et al*. 2001] J. Foote and S. Uchihashi. The beat spectrum: a new approach to rhythm analysis. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 881-884. IEEE, 2001.

[Kurth *et al*. 2006] F. Kurth, T. Gehrmann, and M. MuÌller. The cyclic beat spectrum: Tempo-related audio features for time-scale invariant audio identification. In *Proceedings of the 7th International Conference on Music Information Retrieval*, pages 35-40, Victoria, Canada, Oct. 2006.

[Grimaldi *et al.* 2003] M. Grimaldi, P. Cunningham, and A. Kokaram. A wavelet packet representation of audio signals for music genre classification using different ensemble and feature selection techniques. In *Proceedings of the ACM SIGMM international workshop on Multimedia information retrieval*, pages 102-108, Berkeley, California, 2003. ACM Press.

[Mallat 1999] S. Mallat. *A wavelet tour of signal processing*. Academic Press, San Diego, California, 1999.

[Rauber *et al.* 2002] A. Rauber, E. Pampalk, and D. Merkl. Using psycho-acoustic models and self-organizing maps to create a hierarchical structuring of music by sound similarity. In *Proceedings of the International Conference on Music Information Retrieval*, Paris, France, Oct. 2002.

[Meng et al. 2011] Meng, H. and Bianchi-Berthouze, N., Naturalistic. Affective Expression Classification by a Multi-Stage Approach Based on Hidden Markov Models, in *Proceedings of 1st International Audio/Visual Emotion Challenge and Workshop (AVEC 2011)*, LNCS (6975): 378- 387.