MULTIMODAL SENTIMENT ANALYSIS ON SONGS USING ENSEMBLE CLASSIFIERS

Draft of April 27, 2015 at 11 : 48

BY

ESTEBAN GOMEZ

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science in Electrical and Computer Engineering
in the Undergraduate College of Engineering of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Adviser:

Minh N. Do

ABSTRACT

We consider the problem of performing sentiment analysis on songs by combining audio and lyrics in a large and varied dataset, using the Million Song Dataset for audio features and the MusicXMatch dataset for lyric information.

The algorithms presented on this paper utilize ensemble classifiers as a method of fusing data vectors from different feature spaces. We find that multimodal classification outperforms using only audio or only lyrics. This paper argues that utilizing signals from different spaces can account for inter-class inconsistencies and leverages class-specific performance. The experimental results show that multimodal classification not only improves overall classification, but is also more consistent across different classes.

Keywords: Music Information Retrieval; Sentiment Analysis; Multimodal Classification; Classification Algorithms; Multimodal Fusion

ACKNOWLEDGMENTS

I want to thank Prof. Minh N Do, Ben Chidester and the rest of the Prof. Minh Do's research group for all the guidance and help they have provided through the development of this project.

TABLE OF CONTENTS

1

INTRODUCTION

In recent years, music based services have been trying to make their applications more user-centered. One way to make sure that the user experience is foremost, is to provide services that match the user's current emotional state. This can be implemented by understanding the emotional content of the songs and playlists that are being played.

Sentiment analysis is the portion of Music Informational Retrieval (MIR) where an algorithm recognizes the main emotions that a song evokes. Emotions are subjective, so classifying media into distinct groups is a challenging problem. Most human subjects agree in broad strokes on emotional classifications. However it is not uncommon to find media inputs where there is no consensus, leading to inconsistencies in class groupings. As a result, sentiment analysis is still an open problem that requires the investigation of alternative methods that employ different modalities to counteract class inconsistencies. Success in this approach would allow a more robust classification algorithm that is better suited for real-world applications.

The motivation behind this research is to find a method that accounts for these inter-class inconsistencies across a large dataset. During initial testing it became apparent that certain classifiers perform well on specific sentiments and fail to learn features that represent others. This research seeks to account for this difference by combining the classifiers to output classifications in a more consistent manner, thus improving the reliability of the overall system.

This paper will be focusing on the employment of Support Vector Machines (SVM), Gaussian Naive Bayes and Multinomial Naive Bayes classifiers to recognize emotional information. The features that will be employed are MFCC-like vectors obtained from the response of the song's windowed spectrogram to different base functions to represent the audio portion of the song and a bag of words vector that contains the lyric information of the song. The modal fusion will be tested through both feature fusion and classifier

fusion.

The dataset used for this paper is called the Million Song Dataset, and was compiled by Labrosa [1]. This dataset contains a million different songs represented by their pitch, loudness and timbre. The songs are also accompanied by plenty of metadata such as artist, release date and tags. Sentiment classification was obtained from these tags. If a sentiment was used to described a song, it is assumed that the song conveyed that sentiment. The lyric information was obtained from the musicXmatch dataset [2], which provides lyric information out of order in a bag-of-words format. Since there was no way to obtain semantic information from an unordered bag-of-words representation, this research did not focus on the impact of semantics on sentiment classification.

This document starts with a brief overview of previous work performed on this topic, followed by a description of the methods and the algorithms developed, then a section detailing the actual experiments, followed by the results and corresponding analysis and closes with suggestions for future work.

2

PREVIOUS WORK

There has been considerable amount of work done in the field of multimodal sentiment analysis. This section will briefly cover a portion of the relevant research that was considered during the development of the presented methodologies. The relevant topics that were researched for this paper were: Audio Sentiment Analysis, Text Sentiment Analysis and Multimodal Classification.

Audio Sentiment Analysis

The study of the relationship between emotional content and audio signals is a very mature field. Researchers have expanded on the success found in the speech recognition community while using Mel-Frequency Cepstral Components (MFCC) to explore their uses in music modeling [3]. MFCCs are currently a staple in audio processing and are commonly used in MIR applications such as genre classification [4], since it is a quantifiable method for comparing the timbral texture of songs. Timbre has been used with some success to classify the emotional content of songs [5], however class inconsistencies proved to be a difficult challenge that create substantial misclassification between edge cases. Timbre has also been used to generate songs that evoke particular emotions [6]. These vectors have been commonly classified using Support Vector Machines (SVM) and Naive Bayes classifiers.

Text Sentiment Analysis

Similarly the study of the relationship between text and emotional content is quite developed. From predicting Yelp ratings based on the sentiment expressed on the review [7] to extracting the emotional progression of major

literary pieces [6]. There are many methods to represent and extract emotional information from texts. The Yelp experiment uses statistical word vectors to capture word semantics and emotions as a probability. Other researchers have represented textual information in a bag-of-features framework and classified them using Naive Bayes, SVMs and Maximum Entropy classifiers to recognize positive or negative valances [8].

Capturing the semantic nuisances has also been an area of great interest, which is the study of how a given word might have different emotional value depending on its context. This level of analysis requires the creation of complex sentiment vectors that encode how meanings change based on semantics [9]. Similarly researchers have improved classification accuracy by preprocessing text [10], and use the cleaned data to capture emotional subtleties like the use of negation and modifiers to emotional words [11].

Although there is a great body of research on how to obtain rich sentiment vectors from text, the goal of this paper is demonstrate the added advantage of a multimodal approach. For that end, the lyric vectors were kept simple to clearly underline the benefit of combining them with audio information. In addition, it is necessary to point out that obtaining a large enough dataset of lyrics is difficult due to legal restrictions. As a result, the features used will be the unordered representation of the lyrics in a bag-of-words vector provided by the musicXatch dataset [2].

Multimodal Classification

Multimodal classification is the task of using feature vectors from different spaces, for example text and audio, to reach a single classification. There are two main methods of combining the information from both vector spaces: feature fusion and classifier fusion [12].

Feature Fusion is the technique that takes signals from different feature spaces and joins them to train a single multimodal classifier. The standard fusion method is called series fusion, which consists on concatenating the vectors together and training the classifier on the union of both spaces. Several alternatives have been suggested to maintain the same amount of expressibility in the vector while keeping the vector space as small as possible. Instead of concatenating the vectors together, it is possible to join vectors in parallel

[13] by making vectors from the linear combinations of a real-valued feature with another complex-valued feature. The benefit of the series fusion over the parallel method is that many diverse features can be fused together to obtain more robust data. As seen in the research by Liang et al., genre classification was improved by joining five different vectors all resulting from different preprocessing methods for text and audio vectors [14].

Classifier fusions consists on training an array of unimodal classifiers and using some function to consolidate the predictions [15]. This method seamlessly fuses features from very different spaces. Caridakis et al. combined facial expressions, body gestures and speech by having a classifier voting system where the class with most votes and higher probability was chosen amongst all the decisions [16]. The final decision-making process can be taken a step further by adding an additional classifier that learns from the decisions provided from classifier array [17]. The algorithms presented on this paper were largely based on this last approach.

Multimodal classification has been successful in improving the accuracy of classification [18] [12]. However, some of the previous work either ran the experiments on highly homogenous datasets where all the music was in the same language, belonged to the same genre or were carefully classified by a single subject thus eliminating class inconsistencies. The goal of this research is to obtain improved classification and reliability from a varied dataset through ensemble classifiers.

3

METHOD

Emotions are highly nuanced since there are many experiences that do not fall neatly within a category. The difference between bittersweet and nostalgic, for example, is very different to quantify. The six Ekman emotions are joy, sadness, anger, fear, disgust and surprise [6]. For this paper we will focus on classifying songs based on the first three emotions (joy/happiness, sadness and anger), mainly because there is not enough sample songs whose main emotions are the last three (fear, disgust and surprise).

Features Used

The audio features used for the experiment are the timbre data from the EchoNest dataset. These features are zero-mean vectors of length 12, extracted by windowing the spectrogram in constant-width segments. Although the segment width varies from song to song, it is typically under a second long and it is determined so that the timbre and harmony are relatively uniform throughout.

The song is broken up into a series of non-overlapping windows $\mathcal{S}_i$ for the i^{th} segment. The images in Figure 1 are a visual representation of the basis functions g_j that were applied to each segment. An audio feature f is extracted from each segment and has the form of a vector of length 12, where each element of the vector is defined as follows:

$$f_j(x) = \sum_{p \in \mathcal{S}_i} g_j(p) \bullet x(p) \qquad j \in \{1, 2, ..., 12\}$$

The lyric information was obtained from the musicXmatch database [2] that provides the songs in a bag of words format. This format consists of a vector of length 5000 representing a stemmed dictionary where the value at each element is the number of times that a particular word is used in the

song. Data is distributed in this form so that it respects artists' rights over the ordering of the words while still being able to represent the content.

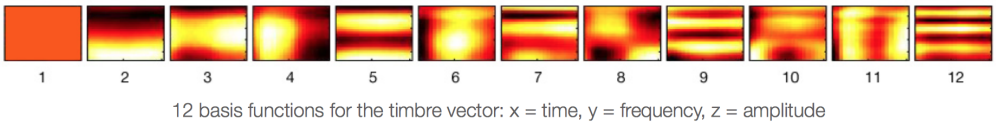

Figure 3.1: EchoNest's basis functions used to generate MFCC like vectors.[1]

Classifiers

As mentioned in the Chapter 2, similar problems have been address with certain amount of success using Naive Bayes and SVMs. For the experiment the classifiers selected were the Gaussian Naive Bayes, Multinomial Naive Bayes and Kernel SVM with histogram intersection as the kernel.

Naive Bayes

The Naive Bayes classifier is a statistical classifier that selects the most-likely class given a particular data point. The classifier builds a statistical model for each class that follows a given distribution. Under the assumption that each feature is statistically independent, it computes the conditional probability for the input vector given each model and returns the model with the highest probability.

$$\hat{y} = \text{argmax}_y P(y)\Pi_{i=1}^{n} P(x_i|y)$$

The distinction between the Gaussian Naive Bayes and the Multinomial Naive Bayes is the distribution that is assumed for the conditional probability $P(x_i|y)$. The Gaussian Naive Bayes classifier, as the name suggests, assumes that data is normally distributed and that the covariance matrix are diagonal. It is a good starting point when little information is known about the data.

The Multinomial Naive Bayes supposes that data follows a multinomial distribution, which is a generalization of the binomial distribution. The

[1] `http://developer.echonest.com/docs/v4/_static/AnalyzeDocumentation.pdf`

difference between binomial and multinomial is that in binomial the outcome of each of the k trials is either "yes" with a probability of p or "no" with a probability of $(1 - p)$. Under the multinomial distribution, the outcome of each of the k trials can be one of n different classes such that $\sum_{i=1}^{n} p_i = 1$. Since an event cannot occur a negative number of times, the multinomial distributions is well suited for problems where the vectors are non-negative such as for histogram analysis.

Support Vector Machines

Support Vector Machines are classifiers that construct hyperplanes that maximize the margin to the labeled data points. The margin is defined to be the shortest distance from hyperplane to any of the points. As a result a linear SVM finds a function that linearly separates the data points into clusters for classification. Soft SVMs exist that tolerate data that is not linearly separable by allowing some misclassification.

An SVM can be non-linear if the data points are preprocessed non-linearly before creating the hyperplane. Non-Linear SVMs rely on functions called kernels that map data points into a higher dimensional space where the points can be separated by a single hyperplane.

For the experiments detailed on this paper, we used a non-linear SVM with a histogram intersection kernel. Histogram intersection is a method that finds the minimum intersection between two histograms [19], which means that it is generally more accurate similarity metric than euclidean distances for histograms. The histogram intersection kernel is defined by the function below:

$$k_{HI}(h_a, h_b) = \sum_{i=1}^{n} min(h_a(i), h_b(i))$$

The formula above details the metric used to compare the intersection of two histograms. The distance of a particular pair of histograms is the sum of minimum value of each column of the histogram. By computing the distance between every pair of histograms using this metric, the SVM can easily group similar data points together.

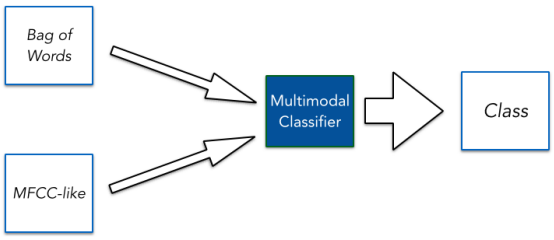

Figure 3.2: Series Fusion

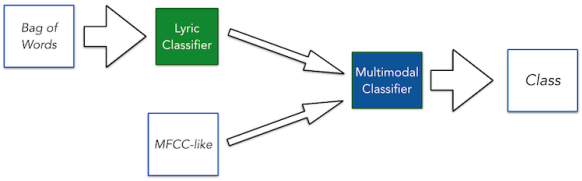

Figure 3.3: Lyrics Only Partial Ensemble

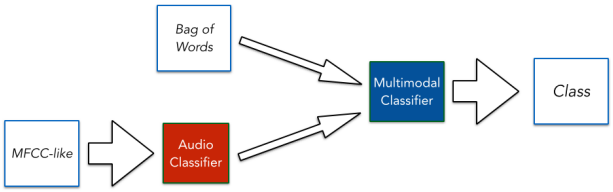

Figure 3.4: Audio Only Partial Ensemble

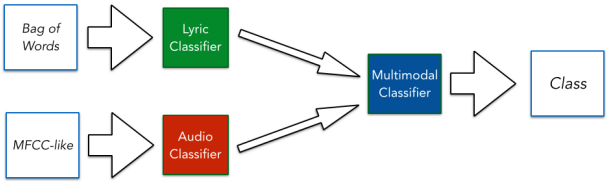

Figure 3.5: Full Ensemble

Fusion Methods

Using the features and the classifiers detailed above, four different fusion methods were used for classification: Series Fusion, two separate Partial Ensemble and Full Ensemble. These were defined by concatenating the feature vectors at different levels of processing. The figures above show a graphical representation of the different structures implemented.

It is important to note that out of the classifiers detailed above, only Gaussian Naive Bayes is suited to classify negative-valued features. As a result not all the classifiers can be used for all the vectors. The Multinomial Naive Bayes classifier requires that audio segments be normalized to be non-negative to be fitted to the multinomial distribution. We introduce the following classifier sets:

$\mathcal{A}$ = set of audio classifiers = {Gaussian NB, Multinomial NB}

$\mathcal{L}$ = set of lyric classifiers = {Gaussian NB, Multinomial NB, SVM}

$\mathcal{M}$ = set of multimodal classifiers = {Gaussian NB, Multinomial NB, SVM}

The set $\mathcal{A}$ is used for audio features, the set $\mathcal{L}$ is used for lyric features and the set $\mathcal{M}$ is used for multimodal features. In the figures above, the squares labeled as Multimodal Classifier, Audio Classifier and Lyric Classifier represent a sampled classifier from the corresponding set.

Series Fusion

The Series fusion method is the basic approach discussed in the Chapter 2 of this document. This method concatenates each raw audio vector with the corresponding raw lyric vector. As a result each song has hundreds of segments, each of which is a vector of length 5012. This large dataset can be computationally prohibitive for certain classifiers. The classifier set for this structure is :

$$\text{Series} = \mathcal{M} \setminus \{SVM\}$$

The reason why SVMs were not included in this test was because the runtime complexity of training an linear SVM is at worst $O(d^2 n)$ where d is the number of dimensions and n is the number of samples [20]. With a hundred vectors per song where each vector has 5012 dimensions, it becomes impractical to run this test.

This fusion method involves taking a classifier from the set *Series* and training it on the dataset resulting from the concatenation of the audio vectors and bag of words.

Audio Partial Ensemble

The Audio Partial Ensemble methods come from noting that the complexity of the Series Fusion method stemmed from the fact that the amount of vectors for each of the modes was lopsided. Each song has exactly one bag of words vector of length 5000, while having hundreds of audio segments of length 12. Audio Partial Ensemble explores preprocessing those hundreds of segments and representing them as a single vector that can be combined with the bag of words.

This method initially classifies all the song's audio segments, and builds a histogram for each song containing how many of these segments were classified for each emotion. As a result the audio data goes from being represented by hundreds of vectors of length 12, to a single histogram of length 3 that provides some concrete information of the emotional content of the song.

The classifier set for Audio Partial Ensemble is defined as all the combinations between the Multimodal classifier set and the Audio classifier set. The resulting set is as follows:

$$\text{Audio Partial Ensemble} = \mathcal{A} \times \mathcal{M}$$

For this fusion method a classifier is taken from $\mathcal{A}$, and is used to build the emotional histogram from the audio segments. This histogram is then concatenated to the bag of words to create the multimodal vector, which is used to train the classifier from $\mathcal{M}$.

Lyric Partial Ensemble

The Lyric Partial Ensemble is very similar to the Audio Partial Ensemble, with the modification that the preprocessing is performed to the bag of words vector instead of the audio.

The classifier set for Lyric Partial Ensemble is defined as all the combinations between the Multimodal classifier set and the Lyric classifier set.

$$\text{Lyric Partial Ensemble} = \mathcal{L} \times \mathcal{M}$$

This fusion method takes a classifier from $\mathcal{L}$ and determines the likelihood that each bag of words belong to each class, resulting in a vector of length 3. This likelihood vector is concatenated to the MFCC-like vectors from the audio data to create the multimodal features. A classifier from $\mathcal{M}$ is used to make the final decision based on this vector.

Full Ensemble

Taking both Partial Ensembles to the next logical step, Full Ensemble preprocesses both the audio and the lyric vectors and concatenates the the intermediate results into a single multimodal vector. The audio vectors are converted into the same emotional histogram used in the Audio Partial Ensemble method. The bag of words, like in Lyric Partial Ensemble, is used to create a likelihood vector. The emotional histogram and the bag of words classification are concatenated and used as a single vector for multimodal classification.

The classifier set for Full Ensemble is defined by all combinations between the Audio, Lyric and Multimodal classifier sets. The resulting set is defined

as follows:

$$\text{Full} = \mathcal{A} \times \mathcal{L} \times \mathcal{M}$$

4

EXPERIMENT

Given the size of the dataset, it was necessary to perform significant preprocessing for each test to run in a sensible amount of time. The experiments require that every song has audio vectors, a lyric vector and a true emotional label. To get the subset of the Million Song Dataset that had these characteristics, it was necessary to perform two levels of preprocessing. First we needed to identify which songs had sentiment labels. From this subset, we needed to identify which songs were also included in the musciXmatch dataset. As a final step, the dataset was truncated so that each class had the same number of samples to avoid having any misrepresentation.

Song labeling was performed by looking at the mbtags provided in the metadata provided by the MSDS. If the tags contained the word 'happy', 'angry' or 'sad', the corresponding song was considered to be part of that class. This method resulted in three distinct sets. These sets were cross referenced with the musicXmatch dataset to determine which songs contained both an emotion label and a bag of words. Out of the intersection between the two sets, 1000 songs for each sentiment were randomly selected.

The algorithms detailed in the Chapter 3 were implemented on python using the sklearn toolkit[1]. A python module was developed to handle sampling and extracting features and class vectors, training classifiers and creating the emotional histogram. These functions were built into a single module to ensure that all the tests ran on the same code, thus avoiding any issues with accuracy difference due to code conflicts. To streamline the training and testing processes, 100 randomly selected audio segments were sampled for each song. This allowed the test program to quickly identify where the prediction for one song ended and the next one started without much overhead.

The module was then used to implement five different programs: Series Fusion, Partial Ensemble, Full Ensemble, Audio Only and Lyrics Only. It was important that each program could take user input to select which classifier

is to be used for what portion of the algorithm. A script was written around these programs to try all classifier combinations as described in Chapter 3.

It is important to mention that the Audio Only benchmark showed that Gaussian Naive Bayes consistently provided better results than the Multinomial Naive Bayes, which allowed us to trim a subset of the combinations used for the Full Ensemble tests by requiring audio preprocessing to be done with a Gaussian Naive Bayes classifier. As a result, we ran six different unimodal benchmarks, nine different configurations for Full Ensemble tests, eighteen different Partial Ensemble tests and two Series fusions, for a total of thirty-five different runs.

Each run consisted of training the algorithm four times with different sized training sets to understand the associated learning curves. The training sets were of size 60, 150, 300, 1500 songs and were comprised of equal number of songs for each class. After training the algorithm, it was tested against a set of 300 different songs which was also comprised of an equal number of songs for each class. Each run was repeated thirty times to have many data points for each configuration, to be able to compare performances more accurately. The confusion matrices were recorded for post processing, and the results will be discussed in Chapter 5.

[1]The histogram intersection kernel code was obtained from kuantkid's branch of the open source scikit-learn project.
`https://github.com/kuantkid/scikit-learn/commit/16c82d8f2fe763df7bfee9bbcc40016fb84affcf`

5

RESULTS

This section will present the results of the experiments detailed in the Chapter 4 while providing some insight into what differentiates the fusion algorithms from the unimodal benchmarks. To do so, two different metrics are used to compare the algorithms. Average accuracy is the standard method for comparing classification algorithms. Since one of the goals of this paper is to improve the reliability of the overall system in diverse datasets, it is not beneficial to have near perfect accuracy for one class and sub-random choice on another class. This gap is smoothed out while comparing average accuracies, which is why these algorithms are reinterpreted as Montecarlo random algorithms.

Montecarlo algorithms are randomized algorithms that have a deterministic runtime and a probability of returning the correct result. Random algorithms are generally measured in a worst-case scenario. So in this application the Montecarlo metric is defined as recognition accuracy for the worst class.

Table 5.1 contains a summary of the accuracies of all the configurations. Each row represents an experiment, and it is summarized with the average of all the class accuracies, the standard deviation of the class accuracies and the Montecarlo metric mentioned above. The naming convention used to described each algorithm is as follows:

{Method AudioClassifier - LyricClassifier - MultimodalClassifier}

Where Method can be a unimodal configurations (Audio or Lyrics) or a multimodal configuration (Full Ensemble, Audio Partial, Lyric Partial or Series).

Table 5.1: Accuracy Measured in Average Accuracy and Montecarlo

		Average	StdDev	Montecarlo
1	Audio Gaus	0.5356	0.2447	0.2423
2	Audio Multi	0.5409	0.2318	0.379
3	Audio SVM	0.3508	0.2803	0.218
4	Lyrics Gaus	0.4904	0.2788	0.265
5	Lyrics Multi	0.5671	0.1630	0.3693
6	Lyrics SVM	0.623	0.0864	0.5283
7	Full Gaus-Gaus-Gaus	0.5438	0.2476	0.283
8	Full Gaus-Gaus-Multi	0.5807	0.1342	0.4243
9	Full Gaus-Gaus-SVM	0.5733	0.1303	0.4317
10	Full Gaus-Multi-Gaus	0.6047	0.1115	0.461
11	Full Gaus-Multi-Multi	0.5733	0.1334	0.421
12	Full Gaus-Multi-SVM	0.5872	0.0984	0.483
13	Full Gaus-SVM-Gaus	0.6339	0.0682	0.5717
14	Full Gaus-SVM-Multi	0.5888	0.1293	0.4363
15	Full Gaus-SVM-SVM	0.6388	0.0698	0.5697
16	Partial Audio Gaus-Gaus	0.4811	0.2964	0.2453
17	Partial Audio Gaus-Multi	0.6377	0.1436	0.466
18	Partial Audio Gaus-SVM	0.6846	0.0809	0.5893
19	Partial Audio Multi-Gaus	0.4812	0.296	0.263
20	Partial Audio Multi-Multi	0.5992	0.1763	0.5537
21	Partial Audio Multi-SVM	0.6677	0.1198	0.587
22	Partial Audio SVM-Gaus	0.4888	0.2850	0.2663
23	Partial Audio SVM-Multi	0.6024	0.1656	0.3873
24	Partial Audio SVM-SVM	0.6517	0.0966	0.5403
25	Partial Lyric Gaus-Gaus	0.4902	0.2839	0.251
26	Partial Lyric Gaus-Multi	0.4802	0.2801	0.2623
27	Partial Lyric Gaus-SVM	0.3828	0.3109	0.1433
28	Partial Lyric Multi-Gaus	0.5796	0.1492	0.3857
29	Partial Lyric Multi-Multi	0.5662	0.147	0.3853
30	Partial Lyric Multi-SVM	0.5646	0.1725	0.3763
31	Partial Lyric SVM-Gaus	0.6286	0.0783	0.5533
32	Partial Lyric SVM-Multi	0.636	0.0789	0.553
33	Partial Lyric SVM-SVM	0.3598	0.2920	0.1063
34	Series Gaus	0.478	0.2964	0.2663
35	Series Multi	0.5651	0.153	0.3717

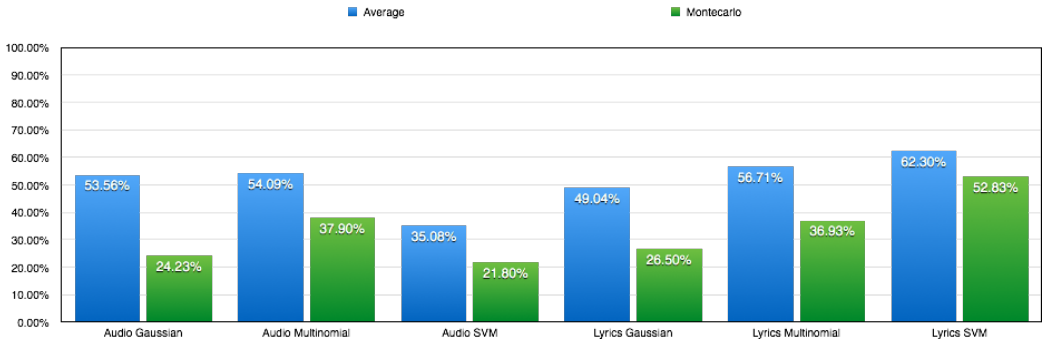

Figure 5.1: Benchmark Summary

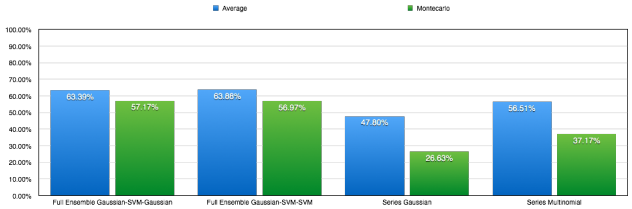

Figure 5.2: Top of Full Ensemble and Series

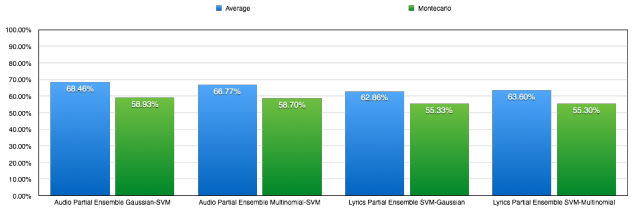

Figure 5.3: Top of each Partial Ensemble

It is interesting to notice some trends in Table 5.1. First note that overall accuracy for Fusion algorithms tend to be slightly higher than the accuracy for the unimodal benchmarks at the top of the table. This is further explored by looking at the standard deviation of these accuracies; the benchmarks tend to have a higher variance the fusion experiments. This trend suggests that the audio only and the lyric only algorithms were very successful on a particular class while performing poorly on another one. Note that the Montecarlo metric tends to be higher when the standard deviation is lower. Algorithms with a high Montecarlo value tend to do better overall.

Figure 5.1 contains a representation of all the benchmarks, which can be compared with Figure 5.2 and Figure 5.3 that contain the two best fusion methods for each configuration. Figure 5.2 shows that Series methods do not perform very well. Looking at the other fusion methods, it becomes apparent that some preprocessing before fusing the data had drastic benefits. Out of all the configurations with some preprocessing, Lyrics Partial Ensemble with SVM-Gaussian had the lowest overall accuracy of 62.8% which is comparable to the highest unimodal accuracy of 62.2%. Note that the majority of the Montecarlo values for the benchmarks are under 40% while the Montecarlo value for all the fusion methods with preprocessing were over 55%.

	H	S	A
h	**58.93%**	15.60 %	16.97%
s	19.93%	**73.07 %**	9.67%
a	21.13%	11.33 %	**73.37%**

(a) Audio Partial Gaussian-SVM

	H	S	A
h	**58.70%**	19.00 %	17.80%
s	18.70%	**67.10 %**	7.70%
a	22.60%	13.90 %	**74.50**

(b) Audio Partial Mutimodal-SVM

	H	S	A
h	**52.83%**	19.77 %	17.27%
s	23.13%	**63.93 %**	12.60%
a	24.03%	16.30 %	**70.13%**

(c) Lyrics SVM

	H	S	A
h	**36.93%**	13.37 %	7.87%
s	32.23%	**59.27 %**	18.20%
a	30.83%	27.37 %	**73.93**

(d) Lyrics Multinomial

Figure 5.4: Confusion Matrices for Top Fusion and Top Benchmarks

Better Reliability

The Montecarlo value serves as an indication of how reliable the system is. A multi-class classification algorithm could possibly have strong average recognition accuracy as a result of performing really well for one class but poorly for the others. Figure 5.4 shows the confusion matrices for the top two fusion algorithms and the top two benchmarks. The columns represent the true label "H", "S", "A" for the classes Happy, Sad and Angry. The rows represent the algorithm's prediction "h", "s", "a" for Happy, Sad and Angry.

Figure 5.4 is very informative in terms of what classes are harder to recognize. Clearly, happy songs are harder to identify that angry songs. This makes sense because angry words tend to only be present in angry music. For example words like "wrath" or "rage" are more indicative of emotion than words like "sunny" or "cheer". The same occurs with the Sad classification, but to a lesser extent.

The benefit of the Montecarlo measurement becomes clear in Figure 5.4.d. This classifier has an average accuracy of 56.71%, as seen in row 5 of Table 5.1. This classifier has some success for Sad and Angry classes, but makes a random guess for Happy songs.This type of behavior would be undesirable for a real application

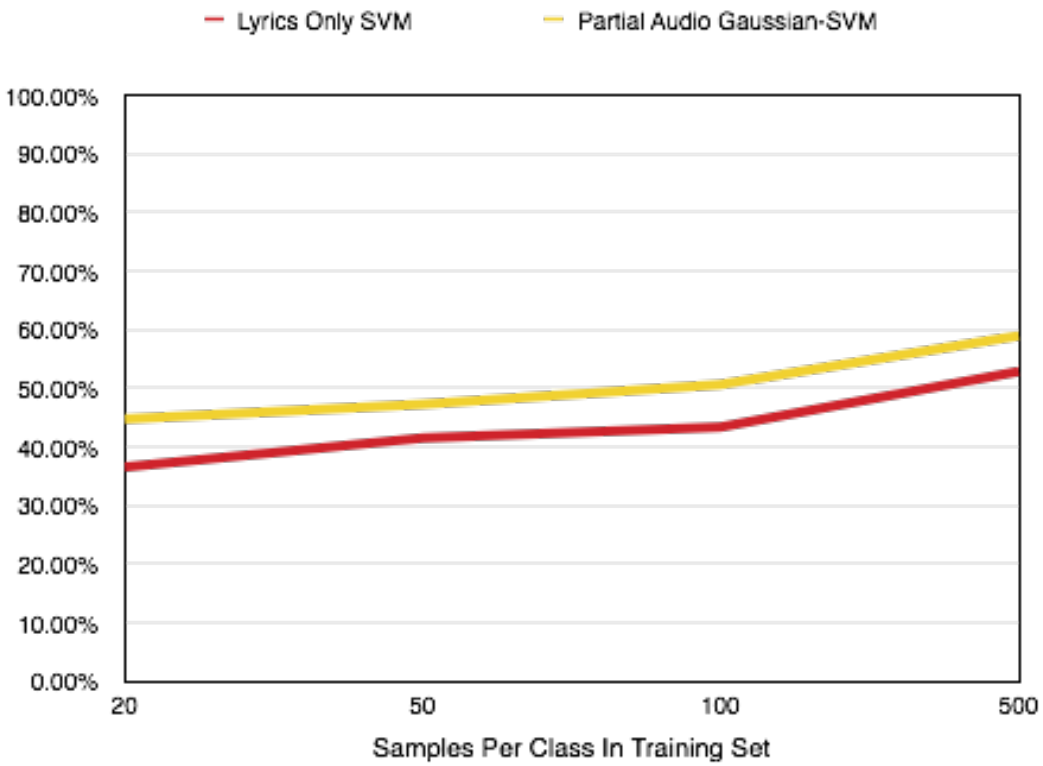

Figure 5.5: Learning Curves for Partial Audio Gaussian-SVM and Lyrics SVM

Smaller Datasets

Multimodal classifiers in general also tended to outperform unimodal classifiers on smaller training sets. Figure 5.5 shows the learning curves of the top fusion method and the benchmark using the Montecarlo metric. The training sets were compiled by taking 20, 50, 100, 500 samples of each class. All the runs tested against a testing set containing 100 samples of each class. As seen in the figure, the multimodal classifier was consistently more reliable than the unimodal counterpart. For the smallest training set, Lyrics SVM had an accuracy of 36.5% which is barely above the random choice accuracy of 33%. On the other hand, the Partial Audio Gaussian-SVM classifier had an accuracy of 44.7% on the smallest training set. It is clear that even with very small datasets, multimodal outperforms unimodal classifiers.

6

FUTURE WORK

This paper explored the benefits of combining classification methods in different configurations. It became apparent that the strongest configuration was the Partial Audio architecture with Gaussian preprocessing of Audio data and an SVM to classify the multimodal vector. The features used for the paper were relatively simple to underline the added benefit of multimodal classification. Possible future work includes used more complex features that take into account sentence semantics to capture word modifiers. Another possible route for improvement on this paper would be to extend the number of classes used to all six Ekman emotions.

REFERENCES

[1] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[2] musicXMatch, "The musixmatch dataset," 2011. [Online]. Available: http://labrosa.ee.columbia.edu/millionsong/musixmatch

[3] B. Logan, "Mel frequency cepstral coefficients for music modeling," in *In International Symposium on Music Information Retrieval*, 2000.

[4] G. Tzanetakis and G. Essl, "Automatic musical genre classification of audio signals," in *IEEE Transactions on Speech and Audio Processing*, 2001, pp. 293–302.

[5] T. L. University and T. Li, "Detecting emotion in music," 2003.

[6] H. Davis and S. M. Mohammad, "Generating music from literature," presented at the EACL in Gothenburg, 2014.

[7] J. Jong, "Predicting rating with sentiment analysis," 2003.

[8] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: Sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, ser. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002. [Online]. Available: http://dx.doi.org/10.3115/1118693.1118704 pp. 79–86.

[9] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ser. HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002472.2002491 pp. 142–150.

[10] E. Haddi, X. Liu, and Y. Shi, "The role of text pre-processing in sentiment analysis," *Procedia Computer Science*, vol. 17, no. 0, pp. 26 – 32, 2013, first International Conference on Information Technology and Quantitative Management. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050913001385

[11] Y. Xia, K. fai Wong, L. Wang, and M. Xu, "Sentiment vector space model for lyric-based song sentiment classification."

[12] J. Zhonga, Y. Chenga, S. Yanga, and L. Wena, "Music sentiment classification integrating audio with lyrics," 2012.

[13] J. Yang, J.-y. Yang, D. Zhang, and J.-f. Lu, "Feature fusion: parallel strategy vs. serial strategy," *Pattern Recognition*, vol. 36, no. 6, pp. 1369–1381, 2003.

[14] D. Liang, H. Gu, and B. O?Connor, "Music genre classification with the million song dataset," *Machine Learning Department, CMU*, 2011.

[15] K. Ahmadian and M. Gavrilova, "A multi-modal approach for high-dimensional feature recognition," *The Visual Computer*, vol. 29, no. 2, pp. 123–130, 2013.

[16] G. Caridakis, G. Castellano, L. Kessous, A. Raouzaiou, L. Malatesta, S. Asteriadis, and K. Karpouzis, "Multimodal emotion recognition from expressive faces, body gestures and speech," in *Artificial intelligence and innovations 2007: From theory to applications*. Springer, 2007, pp. 375–388.

[17] S. Li and C. Zong, "Multi-domain adaptation for sentiment classification: Using multiple classifier combining methods," in *Natural Language Processing and Knowledge Engineering, 2008. NLP-KE'08. International Conference on*. IEEE, 2008, pp. 1–8.

[18] X. Hu and J. S. Downie, "Improving mood classification in music digital libraries by combining lyrics and audio," in *Proceedings of the 10th annual joint conference on Digital libraries*. ACM, 2010, pp. 159–168.

[19] S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[20] O. Chapelle, "Training a support vector machine in the primal," *Neural Computation*, vol. 19, no. 5, pp. 1155–1178, 2007.