Artist Detection in Music with Minnowmatch

Brian Whitman
Gary Flake
Steve Lawrence
NEC Research Institute
4 Independence Way
Princeton NJ 08540
{bwhitman, flake, lawrence}
@research.nj.nec.com

ABSTRACT

In this paper we demonstrate the artist detection component of Minnowmatch, a machine listening and music retrieval engine. Minnowmatch (Mima) automatically determines various meta-data and makes classifications concerning a piece of audio using neural networks and support vector machines. The technologies developed in Minnowmatch may be used to create audio information retrieval systems, copyright protection devices, and recommendation agents. This paper concentrates on the artist or source detection component of Mima, which we show to classify a one-in-*n* artist space correctly 91% over a small song-set and 70% over a larger song set. We show that scaling problems using only neural networks for classification can be addressed with a pre-classification step of multiple support vector machines.

Keywords

Machine listening, music retrieval, audio databases, machine learning, neural networks, support vector machines, information retrieval

1. INTRODUCTION

Digital audio is big, uncooperative, and testy: it angers copyright holders, copyright abusers, and information clients. The mass digitization of most culture has brought with it the embarrassing dilemma of *what to do with it*; how to store it, catalog it, and most of all, make sure only those authorized can inspect it. The bit-induced apocalypse feared by consulting firms' clients had nothing to do with the date rolling over, but rather a quorum of college students not wanting or needing but *expecting* their music to be easily accessible, small, and free.

What's a computer to do? Music, as digitally envisioned, takes 10 megabytes of space for every 2-channel minute. The most common music format on the internet, MP3, helps out by chopping this to roughly a tenth of its size. However, its time-encoded sample-stream gives no hints about the contents inside. Even with meta-tags thoughtfully included into almost every audio specification, tags are rarely used, let alone accurate. Audio retrieval systems, such as the maligned Napster, rely solely on the filename to identify music.

In this paper, we demonstrate NECI's machine listening system, Minnowmatch. While Minnowmatch's (Mima's) architecture is suited to many music IR tasks, here we concentrate on the artist classifier component, which 'listens' to an audio stream to make judgments about the creator of the piece of music. From a new piece of music, we can identify the artist, or if we haven't heard anything by the artist, a composite of other artists that define the song. This system uses a combination of machine learning methods and digital signal processing to make the classifications, and the intended result, to treat pieces of sound like lexemes on a web page, borrows much from information retrieval.

2. ARCHITECTURE

Mima aims to create an information system that learns its own meta-data for retrieval. The uses of such a system are obvious, but the three most important are:

-- *Audio IR*, in which users are presented with a search engine-style interface and can find music or other audio data based on meta-data searches such as "Find me a piece that is more than 120 bpm in a rock style that uses a violin," or "Find all the music composed by Bach."

-- *Copyright Protection*, in which copyright holders can crawl the web or other data stores and download parts of audio to determine if it violates their copyright. This can be accomplished via the automatic meta-data retrieval (does this song sound like one of my artists?) or simple audio similarity.

-- *Recommendation Agent*, where a user might know they like Song A and wants to know more songs *like* Song A. This is made possible by the meta-data generation; even if the system does not compute preference-based recommendations directly on the actual frequency content of Song A, it can determine the similarities between its automatically-derived meta-data and other songs already in the database.

We formally interpret the path of a machine listening system as a process of layers. The source layer, which contains the original audio content, is either resampled or 'ripped' to the common ground of PCM encoding. From there, multiple 'perceptual representations' operate on the PCM data, and from those multiple 'learning representations' transform the data into machine-learning friendly packages. An overview follows:

Table 1: Machine Listening Architecture

Layer	Description
Source →	Source content: song from a CD, radio, file
PCM →	Pulse code modulation encoding of audio
Perceptual Representation (PR) →	Usually frequency based representation: wavelets, FFTs, MFCCs
Learning Representation (LR) →	Abstraction of the PR that allows the learning process a maximum of information and minimizes volume of data
Learning Process →	Neural network, support vector machines (SVM), principal component analysis (PCA), k-nearest neighbor (K-NN), etc
Output / Classification	Method's results: MPEG-7 description, XML-encoded meta-data, etc

3. BACKGROUND

There is a recent spate of interest in music retrieval from both the frequency (recorded or digital audio) and 'score' (notes, transcriptions) domains. Since our system operates entirely in the frequency domain, from digitally recorded or

synthesized audio, we will briefly discuss the recent advances and discoveries in the audio retrieval community that led to this work.

There are many references concerning music information retrieval from the frequency / digital audio domain, but as of yet we are not aware of any work on the artist classification problem. Most music-IR research efforts deal with classification of instrument type, and in some cases genre. However, the techniques and representations used for those tasks are used similarly by Minnowmatch to make its artist classifications. Foote [1] gives an overview of audio information retrieval from the starting point of speech recognition and retrieval, referring to the music domain as "a large and extremely variable audio class." As well, Herrera, Amatriain, Batlle and Serra [2] overview the various classification techniques used on digital audio. Their paper discusses such methods as K-nearest neighbor, naïve Bayesian classifiers, discriminant analysis, neural networks and support vector machines.

Tzanetakis and Cook [3] discuss their audio retrieval system MARSYAS, which operates on various representations of audio to predict genre and classify music from speech. The classification techniques used include Gaussian Mixture Models and K-nearest neighbor algorithms. Martin and Kim [11] also use K-NN for instrument identification, with an overall identification success rate of 70%. Marques [4] uses SVMs to classify solo instruments, combining the MFCC representation with the SVM classifier much as Mima does.

Foote [5] discusses using dynamic programming to retrieve orchestral music by similarity, starting with an 'energy representation' composed of the peak RMS value of each one-second slice of music. He then moves onto a purely spectral feature space. The dynamic programming methods used to retrieve similar music (different performances of the same piece, for example) proved adequate for the small corpus used. Our treatment of music data as lexemes for search and classification is also discussed at the score level in Pickens [6]. His discussion of language modeling and query through inference can be applied to Mima's short-time-slice representation, replacing notes with our frames.

Logan [7] describes the use of Mel Frequency Cepstral Coefficients (MFCCs) for the task of music modeling. The MFCC is presented as a 'smarter FFT,' in that it is scaled to a more psychoacoustically-sound frequency growth, and also has a built-in discrete cosine transform (DCT) step to approximate principal component analysis for decorrelation. Her experiments in 'music / not music' separation show promise for the MFCC in high-level music retrieval.

4. REPRESENTATIONS OF MUSIC

The foremost problem that any perception-based system encounters is how it views a set of digital data. In audio's case, there is an especially telling gap between human perception and the computer's representation.

The most common distribution of digital audio, the compact disc, has a time-encoded bit stream of 16-bit samples triggered every 1/44,000 seconds where two

channels interleave each other, uncompressed. For every second of what is known as “full bandwidth” audio data, 176,000 bytes go by (44,000 samples per second * 2 bytes per sample * 2 channels per sample.) This encoding works wonders on digital devices, since most can remain passive, not needing any extra logic but a digital to analog converter and a clock source. But this encoding is itself a representation of what the audio suggests: bands of frequency data mapping to various air disturbances. The time domain function maps this to samples for the machines’ sake, but human listeners don’t eventually hear rapidly played voltages, they hear that oscillation. So a system that attempts to ‘hear’ as a person would needs to get at the encoding that enables ears to vibrate, namely, the frequency domain.

Two encodings of the same short slice of audio, for example, a human voice singing a high note, can relay vastly different information. The digital audio representation can show the waveform and give some indication of total energy and length. On the other hand, the frequency-domain representation, arrived at via the Fast Fourier Transform, can indicate without even listening that the note is high and, via the peaks (overtones), that it is probably a human voice. Just by studying the composite frequencies of monophonic sources, we can easily set up a pitch tracking device or ‘instrument classifier’ with acceptable accuracy.

4.1 Representations for Classification Learning

Once we start dealing with audio with an important time dimension (a song) and with multiple songs, it becomes clear that a simple FFT should not be the sole input to an artist or source classifier. There are many inherent differences in a source classification or clustering problem versus the single time-slice instrument or pitch detection, and the most obvious is our important time domain. Although we can identify the note being sung in the example above, the artist is unrecognizable from the short time slice.

Perhaps by accumulating much more of an artist’s work than just one song or a piece of one, a pattern could emerge that would show heavy weights on certain frequency bands. For example, a vocalist could have a particular ‘voice print’ that is unique, or a banjo-violin-flugelhorn jazz group could be eventually identified by their frequency peaks. But these all assume that the listener has previous experience with the works of the source in question. If the person has yet to hear any of the banjo-violin-flugelhorn’s music, he or she would obviously have a hard time determining the source of such a song.

For this reason, instead of merely analyzing the frequency components of audio to make meta-data determinations, we use machine learning methods that take in as input a second-tier representation of the frequency components (which we call the learning representation, or LR.) By training the system with audio data in such a representation, Mima can learn over time the frequency curves and fingerprints used by different artists. And with the learned meta-data, we can classify and cluster groups of audio data for similarity and retrieval. Assuming that the best

way to represent audio for machine listening then is via the frequency components, we need to formalize what the learning agent will actually 'hear.'

4.1.1 FFT Perceptual Representation

In the case of the Fast Fourier transform, the output is the coefficients of the frequency vectors given an *n*-sampled time series. The choice of *n* can vary depending on performance requirements (larger windows get more done at once but introduce latency into real-time analyses.) We chose a 1024-sample time window, which returns to us 512 magnitude weights of frequency windows comprising 0hz to 22khz.

4.1.2 MFCC Perceptual Representation

The Mel Frequency Cepstral Coefficients method of determining frequency-domain content from audio derives from the speech analysis community. The MFCC approximates an *n*-dimensional PCA by means of the Discrete Cosine Transform, and it then scales the frequency bands along a more logical range for audio perception [7]. We use a tweaked MFCC representation that considers the entire audio bandwidth (up to 22kHz) and plot eight vectors into our perceptual representation.

4.2 Representation Database

These representations are placed into a file-fed database comprised of "PR Chunks." Each chunk maintains the source file indices (which songs it originated from along with their time offset.) We use a square distance metric with a high threshold to determine self-similarity within this perceptual representation database, in effect compressing our song selection to only the non-repeatable data. Therefore, the sound-slices are treated as words in an inverted index.

5. ARITST RECOGNITION LEARNING REPRESENTATIONS

With the perceptual representation of the songs now in the database, we now discuss how we feed our learning processes. There are a few immediate concerns while creating a workable LR:

-- *LRs should have more examples and less dimensionality per example*. The simplest LR would be a mirror of the input PR—that is, each song would be one long vector of frequency coefficients. Instead, we would rather have a learning agent train over multiple instances per song to create a stronger representation, and also, perhaps more importantly, to allow the learning agent to make 'slice-level' decisions.

-- *Consider the entropy / size tradeoff.* Each instance within a song must contain enough information to complete the task assigned to it (here, artist recognition) but also must not contain too much information. Extraneous data could easily hurt the learning stages of a classifier as well as cause overfitting or slow convergence.

-- *"Compression" should be domain-aware*. The methods used in removing, averaging or otherwise collapsing data from the input PR need to consider where

the most salient information usually lies. For example, an LR operating on FFT data could simply 'chop off' all frequency information above 11khz, in effect creating a brickwall filter above most instrument's ranges.

We first decided to alleviate our biggest problem: low example count. Our chosen representation increases the example count while largely decreasing the amount of input nodes to the first layer of the network. Instead of representing a song as a single one-dimensional array of frequency magnitudes, we 'window' the song into short (roughly one second long) slices, treating each slice as a separate example.

The short-piece representation outlined above was not our first representation, but we found an immediate jump in accuracy once switching to this method, low overfitting (if any), and stable test and training scores for the problem of artist recognition. Our example count jumped far over the input node count, which makes the data easier to manage, understand and visualize at small intervals. Whereas previously, a classification could only be per entire song (*song x is by artist y)*, we can now distribute a classification over the entire time space (given a certain resolution, *95% of the slices of song x are predicted to be artist y*) and throw away badly-scoring slices. This method of 'confidence thresholding' is integral to our results.

6. LEARNING ARTIST CLASSIFIER

Most of our learning system was based on NODElib [8], which was extended to accept the new audio data. NODElib was a good choice for its multiple learning tools built-in; with the same architecture we could test various permutations on neural networks, as well as SVMs and principal component analyses. We discuss the following methods in this section: 1) a multi-layer perceptron based neural network; 2) a series of support vector machines combined at the final stage; and 3) a series of support vector machines with outputs fed into a neural network. We discuss the advantages and disadvantages of each approach, and briefly discuss each method's results. A more detailed results chart follows in section 7.

6.1 Choosing an "Artist Space"

Artist classification is subject to bias due to the nature of the problem. The artist space used for classification should at least contain music from varying genres as well as over long time periods. For example, if we merely claimed that one entire album by an artist was the only representation in our database of that artist, we run the risk of classifying the producer of that record or even the mastering techniques. In choosing our training and test set, we tried to chose music in "clusters," and maintain a set of artists in each cluster. For example, we would have a "recent top 40" cluster and choose five artists we felt fit the category, along with an "early 70s psychedelia" cluster and choose five more artists in that cluster. By doing this, our training set will contain clusters of similar and dissimilar music examples.

6.2 Neural Network Approach

To train the system with a neural network, we had to decide on an architecture that would allow various representations to be swapped in, and also an encoding that would allow us to classify unknown songs later on. We chose to have the network have a set of double-precision floating point values as each input node, and *n* output nodes, where *n* is the number of artists present in the training set. Each output node is set to either 0.9 or 0.1 depending on the classification.

For training, the artist classification is determined by the meta-data sent by the representation server. Each slice has a group of meta-data associated with it: the artist, the song title, the length of the song, where in the file this time slice is, the length of the time slice, and a unique identifier. The learning agent also maintains global information about the example sets, such as the number of unique artists, the total number of songs, and the total number of example sets. Once each example set is aligned and ready for training, we choose our training parameters and start the process. When it completes, it writes a binary representation of the network to disk and begins the testing phase. The neural network's activation function, number of nodes in the hidden layer, and learning algorithm could all be altered at run time. Through trial and error we found that a Gaussian activation function performed best over the mostly-noisy audio data, and having at least as many hidden units as input nodes supported high testing accuracies.

One major feature of our learning process is its *confidence evaluation*. Using the more optimal representation of the time slices, we discovered that a large number of example sets were close enough to other artists that the network would become confused and assign low scores to each output unit. Since there are a large number of slices per song, we determined that we could easily "throw away" many of the neural network's decisions if we knew that they weren't as solid as some of the other ones. With that in mind, we developed a *confidence* metric, defining confidence as the maximum classification's output node subtracted by the second-highest's classification output node multiplied by the maximum's again. We only include determinations that have a confidence metric over 0.1. During the testing process, we determine which output node had the most of these over-threshold determinations and declare it the winner.

The NN training and testing results were very high (85%) over our small test set of five artists. However, when the artist count increased, the accuracy rating on the test set decreased. This scaling problem plagued us, and to resolve it we decided to try a different approach.

6.3 Support Vector Machine Approach

Support Vector Machines [9,10] offer a few distinct advantages over neural networks for certain classification tasks. They are designed from first principles to generalize well, and they have been shown to perform better in many cases under

noisy data conditions [12]. However, generalizing SVMs to more than a single output is a current area of research, which is problematic for artist detection since we require multiple outputs to correspond to multiple artists. To counteract this, we developed a system that creates *n* support vector machines, one for each artist. At the testing stage, we run an unknown short time slice through each SVM and assume that the largest returned magnitude is the 'winning' SVM. A more thorough description of this process follows. We use the same input node format as described in 6.2 above; however, instead of choosing all examples in the LR, we iterate through the set of artists and create separate "artist support vector machines" for each one.

Then, for each example in the test set, we compute the possible classification for each artist. Accounting for confidence, we throw away low scoring results immediately and accumulate the highest-scoring artist SVM for each input test slice. The SVM performed nearly perfectly on the training set and slightly worse on the 5-artist test set. However, for larger test sets, where the NN method degraded rapidly, the SVM approach performed significantly better.

6.4 Combined SVM to NN Approach

Since one of our yet unknowns was how to determine the 'winner' of a test set slice using the SVM method, we decided to feed the resulting output of each artist-trained SVM to a new neural network. The process is the same as 6.3, except after training each individual artist SVM, we create a "Metalearner" training machine, which is a neural network that has *n* inputs, one for each output of an artist SVM, and *n* outputs, 0.1 on each except the 'correct' artist, which has 0.9.This neural network is then trained with each example of the LR. We feed each LR through each previously trained SVM, and create the metalearner input vector iteratively. After creating this metalearner dataset, we train it and store the resulting neural network. For the test set, we then use the stored metalearner to compute the probable artist for the input slice example.

7. EXPERIMENTS AND RESULTS

To test each artist classifier, we set up training and test case sets of songs, making sure to choose the ends of ranges both stylistically and over an artists own "time-domain" (collection of albums.) For testing and training purposes, we stuck with a combination of artists traversing such "short-span" music as pop, rock, electronic, and jazz. We immediately pruned and sanitized our sets by removing anomalies such as too-long tracks (> 6 minutes), end-of-CD "hidden" tracks, etc. We ended up with 210 songs covering 21 artists. For test / train set separation, we split each album (or, more precisely "same-time collection of artist's songs" in the case of singles, compilations, etc.) in half. We also created subset lists of varying sizes to give the network differing numbers of artists to classify: along with the full 21-artist list we also had 10 and 5 artist lists. With the representation outlined above (short-time-slice), we switched between using the FFT as a PR and the MFCC. We found that generally the MFCC provides better results for more classification tasks. We consistently used a 1024-window FFT 'averaged' down to

20 frequency slices, mel scaled with a 12-times skip over time slices, and for MFCC, we used 8 vectors, also with a 12-times skip.

Table 2: Artist Classification Results

Artist List (Baseline)	NN Only Per-song accuracy	SVM Only	SVM/NN
5 artists **(20%)**	Train: 25/27 **93%** Test: 23/27: **85%**	Train: 34/34 **100%** Test: 27/32: **84%**	Train: 34/34: **100%** Test: 29/32: **91%**
10 artists **(10%)**	Train: 24/49 **49%** Test: 19/50 **38%**	Train: 48/48: **100%** Test: 34/50: **68%**	Train: 48/48: **100%** Test: 35/50: **70%**
21 artists **(5%)**	Train: 41/127 **32%** Test: 20/131 **15%**	Tr: 113/113: **100%** Test: 57/114: **50%**	Tr: 108/113: **96%** Test: 52/114: **46%**

The combination SVM/NN and the SVM alone help solve the scaling problem with the neural network method. Although our best result over larger artist spaces (70% for 10 artists, and 46% for 21 artists) do not sound very high, we must consider the difficulty of the problem as well as the baseline. We note that the artist sets were intentionally chosen to be difficult and we believe that a human listener under the same circumstances could not do as well, and most definitely not so quickly. Additionally, we also note that many applications may perform well even if the correct artist is one of the top few predictions rather than the top prediction.

8. CONCLUSIONS

We have shown that by using a small amount of frequency content in music data, an automatic artist / source classifier can be built with high accuracy. The set of songs were chosen to be intentionally confusing to a machine listener, and the relatively small set of features extracted by the perceptual and learning representations were enough to separate them. We show that using a combination of two learning techniques helped solve the issue of scaling over large test sets, in effect using support vector machines to cluster the data before using a neural network to perform the final classification.

9. ACKNOWLEDGEMENTS

The authors wish to thank Adam Berenzweig for his helpful contributions in the evaluation stage of this paper.

10. REFERENCES

[1] Foote, Jonathan. An Overview of Audio Information Retrieval. ACM-Springer Multimedia Systems, 1998

[2] Herrera, Perfecto; Amatriain, Xavier; Batlle, Eloi and Serra, Xavier. Towards instrument segmentation for music content description: a critical

review of instrument classification techniques. Proceedings of ISMIR 2000, Plymouth MA, 2000.

[3] Tzanetakis, George and Cook, Perry. Audio Information Retrieval (AIR) Tools. Proceedings of ISMIR 2000, Plymouth MA, 2000.

[4] Marques, J. An automatic annotation system for audio data containing music. BS and ME thesis, MIT, Cambridge, MA, 1999.

[5] Foote, Jonathan. ARTHUR: Retrieving Orchestral Music by Long-Term Structure. Proceedings of ISMIR 2000, Plymouth MA, 2000.

[6] Pickens, Jeremy. A Comparison of Language Modeling and Probabilistic Text Information Retrieval Approaches to Monophonic Music Retrieval. Proceedings of ISMIR 2000, Plymouth MA, 2000.

[7] Logan, Beth. Mel Frequency Cepstral Coefficients for Music Modeling. Proceedings of ISMIR 2000, Plymouth MA, 2000.

[8] Flake, Gary. NODElib: www.neci.nj.nec.com/homepages/flake/nodelib/html/

[9] Vapnik, V.N. The Nature of Statistical Learning Theory. New York: Springer-Verlag.

[10] G. W. Flake and S. Lawrence. Efficient SVM Regression Training with SMO. Machine Learning, 2001. To appear.

[11] Martin, Keith, and Kim, Youngmoo. Musical Instrument Identification: A pattern-recognition approach. Presented at the 136th meeting of the Acoustical Society of America, 1998.

[12] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):955-974, 1998.