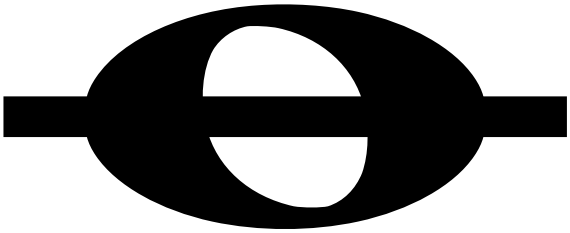
Musical Analysis by Computer from Audio Recordings

Hans Fugal



Prepared in partial satisfaction of the requirements for the doctoral comprehensive exam

Department of Computer Science
New Mexico State University
May 2008

Contents

Introduction

Computer Music is the broad field of "doing musical stuff with computers." It includes synthesis of sounds and musical compositions. It includes beat tracking and instrument or genre identification. It includes automatic accompaniment of jazz soloists, or analytical analyses of musial features. It even includes audio compression algorithms like Ogg Vorbis or MP3.

In this review, we will consider a sample of the literature regarding analysis of music from audio recordings. Therefore synthesis, accompaniment, and many other interesting aspects of computer music will be out of scope.

The starting point for the algorithms discussed in this paper is an audio signal. That is to say, an audio recording that has been (or is in real time being) converted to a digital signal: a single stream of numbers representing the sampled amplitude of the sound wave (two streams for stereo). That we can represent the analog soundwaves as a discrete series of numbers and not lose any perceptual information is an amazing fact, but not one we will concern ourselves with. In this paper I assume a basic familiarity with digital signal processing (DSP) concepts. There is a primer on digital signal processing in Appendix B.

Likewise, I assume a basic familiarity with musical concepts like quarter and half notes, key signatures, chords, etc. There is a primer on music in Appendix A.

Music is not just notes, but also silence. The passage of time is as important as the notes and harmonies. Likewise, the sound and character of the instruments and voices are fundamental to our perception of music. Music has three primary dimensions: rhythm, pitch, and timbre. We will discuss each of these in turn, and look at what research has been done in analyzing each of these dimensions of music. Then we will briefly look at the big picture, what we might call music understanding.

Chapter 1

Rhythm

1.1 Introduction

It can be argued that rhythm alone is musical. Rhythm is perhaps the most accessible aspect of music. Even the most nonmusical among us can tap his foot to the beat (albeit sometimes imperfectly) and appreciate rhythmic music. This fundamental and accessible nature makes metric analysis a natural starting place in the research. It has attracted much attention in the past and will undoubtedly continue to do so.

But metrical analysis from digital audio is not an easy problem. What seems so easy to us is in fact quite difficult to set forth in an algorithm. Many seemingly obvious approaches have been tried, some with more success than others. Almost always, what works well for one set of music does not generalize well to all music. As one example, algorithms aimed at popular music, which is simpler to analyze due to both simpler rhythmic content and consistent features (like drums with easily-detected frequency content and loudness), do not do so well when applied to complex jazz music.

In a review of rhythm description systems, Gouyon and Dixon stated, "The chief goal in automatic rhythm description is the parsing of acoustic events that occur in time into the more abstract notions of metrical structure, tempo, and timing" [GD05]. These rhythmic concepts in which we are interested are indeed abstract. While the abstract musical notation of rhythm is straightforward and agreed-upon, the actual performance and especially the perception of rhythm from performed music is much less clear-cut.

Tempo is the speed of the music. It is generally represented in musical notation as beats per minute. In almost all music (some types of dance music being a notable exception) the tempo varies enough that variation needs to be accounted for in the algorithm. Some music has drastic and immediate tempo

changes from one part to another; the literature so far tends to avoid such music or treat the parts separately.

Timing is the variation from the precise tempo. Some timing is incidental, e.g. the imprecision of human performance. Other timing is subtle but intentional, e.g. the articulation of an organist (playing notes slightly early or late to emphasize or deemphasize them, for musical phrasing). Some timing is not subtle at all, e.g. swing and fermatas. Timing, especially large timing discrepancies, presents the largest difficulty for naïve algorithms.

Metrical structure refers to high-level concepts like time signature, song forms (like verse and chorus), note quantization and durations, bar lines, etc. Although these are important aspects in full transcription, they are relatively unapproached in the literature. Where metrical structure has been approached is primarily in holistic approaches to beat tracking which attempt to detect metrical structure and use it as an aid in more accurate beat tracking. [Dan05] is one such study.

Honing [Hon01] discusses the representation of tempo and timing in depth. He formally analyzes tempo functions, time-shift functions, and time maps under transformation. Tempo functions (tempo as a function of score time) are inadequate for representing issues of timing. Likewise time-shift functions (deviation from a regular pulse) do not carry enough information to separate tempo and timing. Time maps (also known as time deformations or time warping), frequently used in computer music, are a step in the right direction in that they "can, in principle, describe both time-shift and tempo-change." Time maps map (a possibly pre-perturbed) time t to a performance time t':

$$f(t) \rightarrow t'$$

Honing describes several deficiencies of time maps and proposes a new scheme for representing tempo and timing called timing functions. Timing functions combine the two aspects, tempo and timing, into one tuple containing a time map for tempo and a time map for timing. These time maps remain independent through transformation, and are combined when the timing function is evaluated for performance time. This representation avoids the drawbacks of the other methods. He goes on to discuss a generalized timing function that allows timing functions to be related to temporal structure as well.

In [GD05], Gouyon and Dixon give an in-depth review of automatic rhythm description systems. They observe that a direct transcription of metric events to performance time (e.g. MIDI events) is naïve and lacks abstraction. There is a chasm between the abstract notation of the score and the concrete performance, and crossing back in the other direction is all the more difficult.

We generally perceive beats as falling in heirarchical metrical levels. For example, although the quarter note may be the primary beat, we can also tap our feet at twice or half that rate. Sometimes it is not at all obvious to humans, let alone computers, which metrical level is the "correct" primary level.

Gouyon and Dixon briefly discuss the Generative Theory of Tonal Music (GTTM) which attempts to formalize a musical grammar. It defines beats as durationless points in time that are equally spaced, with each metrical level corresponding to a different beat spacing. In this model, all notes fall on a beat in at least one of these metrical levels, so that the smallest metrical level is an abstract quantization of metrical events. Tempo, then, is defined in terms of a metrical level, usually the primary metrical level (which may be somewhat arbitrary). This model is an attractive abstraction, but of limited use when dealing with performed music (as opposed to musical scores), because timing information is absent.

Gouyon and Dixon make an interesting claim, that the ambiguity between tempo and timing "makes causal analysis impossible (i.e., algorithms which do not use information about future events in analyzing present events, as, for example, any real-time algorithm), because with no knowledge of the future, a single "out-of-time" beat could be caused by either a tempo or timing change [CDGW01]."[1] Technically this is true, but it is not the whole story. We cannot see into the future and yet we are able to beat track admirably. I believe that the answer to this paradox is that we, like the computer, do not know at that moment whether it is a tempo or timing change, but we fill in that gap with expectations formed from previous events, and in the near future as more information becomes available we retroactively analyze the meaning of that deviation. Perhaps this is part of the reason why timing and tempo variations are good expressive mechanisms—they introduce surprise and uncertainty which is resolved by expectation and/or near-future events. In computer analysis we may achieve similar advantage by using a holistic approach that emulates that expectation and a not-quite-realtime approach that reserves judgement until a short time later when more information is available.

Almost all studies include a rhythmic event detection step, where rhythmic events (usually note onsets) are detected and enumerated. Earlier work took symbolic input, such as MIDI, and theoretically could be applied to extracted onsets. However onset extraction from audio is not a solved problem. Scheirer [Sch98] denounces this "transcriptive approach" and prefers to work directly from the audio signal with low-level features at the frame level. Gouyon and Dixon unify both approaches as finding a feature list, whether low-level features from the audio signal or extracted (or provided) high-level features. In some cases the low-level features are not explicitly enumerated, e.g. in a real-time algorithm working directly on the signal, but they are implicitly represented nonetheless. Whether or not this unification is valid, or whether or not there is an important difference between the "transcriptive" and "perceptual" approaches, there is no doubt that in both cases rhythmic features are extracted using various signal processing techniques. The efficacy of this

[1] From time to time I will quote passages which contain references, as I did here. I convert the authors' reference to match the rest of this paper and include the reference in the bibliography, although they are not primary sources.

step deeply impacts the overall performance of any algorithm, and they are often tightly coupled—the algorithm depends on the kinds of rhythmic events that the feature detector detects, and on the absence of other kinds that other feature detectors might detect.

Algorithms generally use feature lists to perform either beat induction or beat tracking. Beat induction is the task of finding the tempo of a given section of music. This is usually the basic tempo—the tempo of the primary metrical level that best fits the the entire piece of music. Beat tracking is the task of identifying "the beat," i.e. the regular pulse of the music. It involves tracking that beat throughout the piece, sometimes in the presence of tempo variation and even timing variation. Desain and Honing [DH99] argue that our perception of beat is a combination of these two aspects. We quickly establish a "tempo hypothesis" by tempo induction from the first few notes that we hear, and this hypothesis, or expectation, guides our interpretation of new incoming material. New material also causes us to adapt our hypothesis, so there is a balance between expectation and adaptation.

Two approaches to beat induction are pulse selection and periodicity functions. Pulse selection is to take as an initial hypothesis a pair of available events, then refine this hypothesis when presented with more data. For example, the first pair of notes is a common choice. A periodicity function gives a magnitude (or salience) to a periodicity (reciprocal of tempo). It is obtained in a similar fashion to a histogram, by measuring duration between close event pairs. Sometimes a starting tempo is chosen to prime the algorithm. Sometimes more recent events are given more weight than long past events.

Beat induction assumes a stable tempo. Once the period and phase are determined, the beat can be followed throughought. Beat tracking, on the other hand, assumes that tempo is changing and that the beat must be tracked continuously. Beat induction is usually approached in a bottom-up fashion, while beat tracking is generally approached in a top-down fashion.

Many different approaches to beat tracking have been tried, and we will discuss several of them. Gouyon and Dixon argue that they all can be thought of as state models. A state model has state variables (tempo, phase, etc.), begins in an initial situation, then repeatedly makes observations which may update the state variables via predefined actions. This may be an overgeneralization, but it is clear that the effective approaches all have in common the trait of adaptability. They do the best they can, then continue to adapt as more information arrives (possibly conflicting information, when compared to the past).

Desain and Honing [DH99] review three rule-based beat induction models. The rule-based models discussed take duration values as input, maintain a tempo hypothesis, and update that hypothesis with a set of actions based on the input. The systems being evaluated are from 1982, 1985, and 1994. They are simple and efficient, yet according to Desain and Honing "perform amazingly well." However, the scope is limited—requiring symbolic input

(e.g. MIDI) and assuming stable tempo. In this paper, Desain and Honing are primarily interested in "characterizing the behavior of computational models of beat induction." The results are insightful.

1.2 Onset Detection

Most algorithms work almost exclusively on note onsets. Bello et al. provide "a tutorial on onset detection in music signals" [BDA+05]. They distinguish between three features of a sounded note: transients, onsets, and attacks.

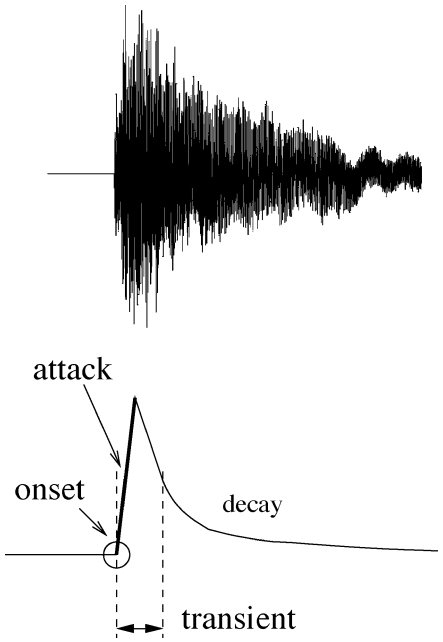


Figure 1.1: " 'Attack,' 'transient,' 'decay,' and 'onset' in the ideal case of a single note" [BDA+05]

They define attack as the ramping up time, where the amplitude envelope is increasing. The onset of a note is a single point in time, probably sometime during the transient and attack. The transient is the period at the beginning of a note where the signal evolves quickly, such as when the instrument is initially excited. Following the attack and transient is the steady phase, which is usually a stable resonant sound. (Bello et al. call this decay, but some instruments can sustain the sound indefinitely, and the decay begins after the sustain ends. On other instruments, decay begins right away) Transients are different—even surprising—on a small time scale, and so transient detection functions are often called novelty functions. The general onset detection

scheme is to optionally preprocess the audio signal, then run a detection function on it to reduce the signal to another signal which has peaks where onsets occur with high probability, and then to use peak-picking to pick the onsets.

Two common preprocessing steps are separating the signal into multiple frequency bands and separating the transient and steady states. Separating the signal into multiple frequency bands helps when the detection function or the beat tracking algorithm works with sub-bands. High frequency and low frequency content both have been useful to various algorithms. "The process of transient/steady-state separation is usually associated with the modeling of music signals... However, there is a fine line between modeling and detection, and indeed, some modeling schemes directed at representing transients may hold promise for onset detection." They briefly describe some methods.

The reduction phase applies a detection function to the (optionally preprocessed) signal. The result is a reduced signal with peaks at detected onsets. Bello et al. review various detection functions. The first type is reduction based on temporal features, usually amplitude envelope. The idea is that the music is louder at the moment of a beat, because loud instruments (like drums) and/or more notes are sounding. This seems natural and so was a popular approach in early systems. However, in reality it is disappointing, especially in the face of modern recordings which often have extreme dynamic compression.

Another approach to detection functions is using spectral features. Because transients are generally broadband, and the energy of the signal is usually concentrated at low frequencies, measuring spectral energy with more weight given to high frequencies is a common detection function. Another approach calculates the distance, or difference, between short-term Fourier spectra. This spectral difference (or flux) will be large at transients. Other approaches are described as well.

Another class of common detection functions is those based on probabalistic models. A probabilistic model is constructed to describe the signal, then the system makes probabalistic inferences about the likelihood of an onset given the observed signal. The effectiveness of this approach is determined by the validity of the model. The model does not need to fully characterize the signal, but what it represents and how that relates to the likelihood of onsets needs to fit well.

One probabilistic system discussed maintains two probabilistic models—one for steady state and one for transients. The system detects the switch from one model to the other which identifies an onset. There are techniques for building up the models from observed data.

Another probabilistic system builds up a model of the signal and then looks for moments of surprise. On a short time scale, the model adapts to the currently sounding note and when a new note sounds (with its transient) the signal no longer matches the model and the system is relatively surprised.

The choice of a detection function will depend on various factors such as the beat tracking algorithm, processing needs, causality requirements, and music type. After reduction, peak picking is performed to extract the onset times from the reduced signal. Desain and Honing discuss a few post-processing and thresholding techniques for the peak-picking stage.

1.3 Transcriptive Approaches

I will discuss several approaches that are what Scheirer would call "transcriptive" approaches, because they use onset detection to identify rhythmic events.

Cemgil et al. developed a system [CKDH01] that models a tempo tracker as a stochastic dynamical system. A dynamical system is characterized by a set of state variables and a set of state transition rules that describe deterministically how the state of the system evolves with time. In this case, the beat and period are modeled. The next beat is the current beat plus the period. Since we are not modeling a perfect metronome, the value of the period changes in time (tempo variation), modeled by a noise parameter. Similarly, the beat is not precise due to timing variation, so the beat is noisy as well. So the actual beat and period are the parameters of the system that we would like to know, but they are hidden. With a linear dynamical system such as this, a Kalman filter can be used to estimate the hidden parameters. The Kalman filter was introduced in 1960 by Kalman [Kal60]. Maybeck gives an excellent conceptual introduction to dynamical systems and the Kalman filter in [May79], and Welch and Bishop give an introduction to the Kalman filter in [WB06]. Cemgil et al. make a few extensions to the basic Kalman filter.

The Kalman filter estimates the state of the beat and period variables given noisy observations of the same. However we cannot observe the beat directly from a musical signal, not even noisily. Cemgil et al. derive these noisy beat observations with a technique they call the tempogram. As the name implies with its similarity to the word spectrogram, it can be viewed as a two-dimensional plot. The x axis is time and the y axis is frequency, and the value at (x, y) is related to the likelihood that there is a beat at time x and that that beat is recurring at frequency y. It is calculated as the inner product between a signal composed of gaussians centered around note events and a windowed pulse train at the given frequency (see Figure 1.2). The tempogram is the primary input to the Kalman filter, which produces an estimate of the beat and period.

This approach is applied to symbolic input, namely MIDI recordings of various pianists performing Yesterday and Michelle by the Beatles. To be applied to arbitrary digital recordings would require generic onset detection. The approach is suited for either causal (real-time) or offline analysis.

Dixon describes a program to estimate tempo and timing in expressively performed music [Dix01]. Expressive performances vary from mechanical per-

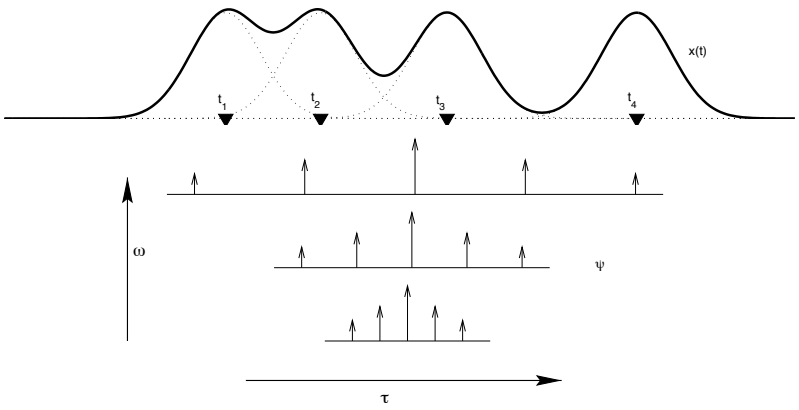


Figure 1.2: "Tempogram Calculation. The continuous signal $x(t)$ is obtained from the onset list by convolution with a Gaussian function. Below, three different basis functions ψ are shown. All are localized at the same τ and different ω. The tempogram at (τ, ω) is calculated by taking the inner product of $x(t)$ and $\psi(t; \tau, \omega)$. Due to the sparse nature of the basis functions, the inner product operation can be implemented very efficiently" [CKDH01].

formances in both tempo and timing. His approach is "based on the belief that beat is a relatively low-level property of music, and therefore the beat can be discovered without recourse to high-level musical knowledge." It works on both symbolic (MIDI) input and audio recordings. Interestingly, it is not causal (it is off-line).

In [Dix01] a rhythmic event is characterized by an onset time and a salience. A key aspect of this algorithm is the salience parameter, and so the imperfection of onset detection actually works to the advantage of this algorithm, as it is assumed that missed onsets are less salient (from the beat-tracking perspective), and that this inherent filter helps rather than harms the algorithm. The onset detection used was a fairly simplistic amplitude envelope tracker, but adequate for for this algorithm for music with instruments with well-defined envelopes.

Once the rhythmic events have been extracted, the inter-onset intervals (IOIs) are examined with a clustering algorithm and given a score, and considered hypotheses for the tempo period. The IOIs of non-adjacent notes are considered as well as adjacent notes, within a window of temporal locality. The scored tempo hypotheses are used to create multiple agents that attempt to track the piece at each tempo. These agents predict beats based on the contained tempo and phase hypothesis, and are adaptive and can adjust their tempo and phase hypothesis to match the observed data. As the song is tracked with varying success by each agent, the agents accumulate a score. At the end of the piece, the agent with the highest score is the final hypothesis

for tempo and phase of the beat.

The system is fairly robust and is able to recover from discontinuities even though no high-level musical knowledge is represented. It is also effective across many styles of music, which have a beat and instruments with sufficiently sharp attacks. The latter restriction could be remedied with a more sophisticated onset detection algorithm.

Goto presents a system for beat tracking from audio recordings of music with and without drum sounds [Got01]. His approach considers onset times, chord changes, and drum patterns. He makes an interesting observation, that it is important to work on real problems from the beginning rather than simplified toy problems. "This is because, as known from the scaling-up problem [Kit93] in the domain of artificial intelligence, it is hard to scale up a system whose preliminary implementation works only in laboratory (toy-world) environments." Therefore he evaluates his work with recordings from CDs.

In Goto's work, onsets are detected in an interesting fashion. The system looks at spectral energy in seven bands and takes into account neighboring bands from the previous frame. Possible chord change events are detected, on the

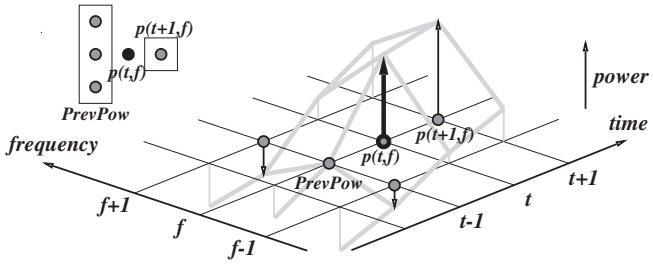


Figure 1.3: Extracting an onset using neighboring bands and the previous frame. [Got01]

premise that chord changes tend to occur on the beat.

Bass and snare drum events are picked out from the audio and the repeating drum patterns are detected. Repeating drum patterns naturally are rich with rhythmic information. Bass drum events are detected by looking for the lowest peaks in spectral histograms, and snare drum by looking for instants of wideband noise.

All of the gathered information is processed by a system of agents based on various heuristics regarding the input data and musical knowledge. They interact with higher-level agents that perform sanity checks. A manager takes the input of all the agents and produces beat information on the quarter-note, half-note, and whole-note levels.

Dannenberg [Dan05] takes a holistic approach, showing improvement in beat tracking by considering musical structure. His baseline beat tracker consists

of two parts. The first computes likely beat events from the audio, which are represented as (time, weight) pairs. The second part looks for an evenly-spaced beat hypothesis with a good match to the observed likely beat events. High frequency content features are employed in detecting likely beat events. The beat detection involves gradient descent over an expanding window, with a goodness of fit function that penalizes abrupt tempo changes in order to prefer a smooth tempo curve.

He applies structure knowledge to improve the beat tracking. A self-similarity matrix is constructed from chroma vectors. A chroma vector is a projection of the Fourier transform magnitude to the 12 chromatic notes (octaves are folded in). The self-similarity matrix will have roughly diagonal paths when a section of music is repeated. By detecting these diagonal paths we can observe elements of song structure. (see Figure 1.4)

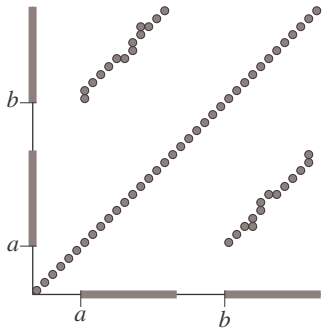


Figure 1.4: "Paths of high similarity in the similarity matrix. Sections starting at a and b in the music are similar." [Dan05]

The beat tracking method is then modified to take advantage of this song structure information. Again gradient descent is used, but develops in a different fashion. As the window expands to overlap one of these alignment paths, analysis jumps to another place in the song, possibly "off the island." Each island is processed in a round-robin fashion. As analysis proceeds, islands grow and merge until there is only one island covering the entire song. The result is the tempo and phase estimate.

The structural holistic approach was substantially better than the baseline beat tracker, indicating that taking advantage of structural information may be beneficial to many other beat tracking approaches as well.

1.4 Perceptual Approaches

Scheirer eschews the onset-based "transcriptive" approach, preferring instead an approach that works directly on low-level audio frames. His method in [Sch98] uses a few bandpass filters and comb filters "to analyze the tempo of, and extract the beat from, musical signals of arbitrary polyphonic complexity and containing arbitrary timbres." His algorithm is based on a psychoacoustic property, that we are able to recognize rhythm when the music is heavily filtered. For example, in one demonstration the music is filtered into amplitude envelopes over a few bands, which is used to amplitude modulate white noise. We are able to recognize the rhythmic content in the resulting signal.

The signal is fed to a filterbank which divides it into six bands. Then the amplitude envelope of each band is calculated, and its derivative taken. This is fed to a filterbank of tuned resonators (comb filters). One of the resonators in each filterbank will phase-lock, and by considering the phase-locked resonators from each filterbank the tempo and phase are estimated. The algorithm is causal.

The system of Sethares et al. [SMS05] breaks an audio signal into various "rhythm tracks," which are reduced signals according to low-level features, and assumes them to be independent then uses either Bayesian analysis (using particle filters) or gradient descent to find the period and phase of the beat. After the rhythm tracks are extracted, the probability of various parameters including phase and period of the beat, given the observed rhythm track, can be calculated using Bayes' theorem[2]. Because modeling these complex distributions is intractable, a particle filter is used to approximate them. The particle filter estimates the probability of the timing parameters given the observed rhythm tracks. At each block of audio, the particle filter predicts the parameters, updates the likelihood of each particle, then resamples the particles from the distribution, which are distributed as the probability of the parameters given the observed rhythm tracks.

Alternatively they used gradient descent in the optimization stage, instead of a particle filter. It was much less computationally intensive, but converges more slowly and undulates about the correct answer.

This algorithm is causal and can potentially be done in real time.

1.5 Conclusion

In our discussion I have been intentionally vague about the results of the systems discussed. There is no established method of evaluating beat tracking systems. One thing that everyone agrees on is that the situation is lamentable.

[2] Bayes' theorem is $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$

This is primarily due to the vague perceptual nature of rhythm and beat. It is a perceptual notion defined by human percpetion, and yet humans are not consistent performers either. Therefore there is no gold standard of truth when it comes to beat tracking. Some papers performed experiments with human subjects as well as computer programs and compared the results. Others constructed a ground truth from musical notation and careful tedious annotation of MIDI or audio files. Unfortunately these corpora are quite small and probably insufficient.

In [Tem04], Temperley presents a survey of metrical model evaluation and proposes an evaluation system. Four requirements of the evaluation system he proposes are: "an agreed-upon way of representing the kind of information to be retrieved; a suitably large and representative corpus of data; correct analyses of the corpus (representing the information to be retrieved); and an agreed-upon way of comparing a model's analyses of the corpus to the correct analyses and scoring the model on its success at matching the correct analyses."

Temperley's proposal for representation of metrical information seems to be adequate to cover the output of the various algorithms. However it is a little quirky in actual implementation. He proposes to encode the beat lists using

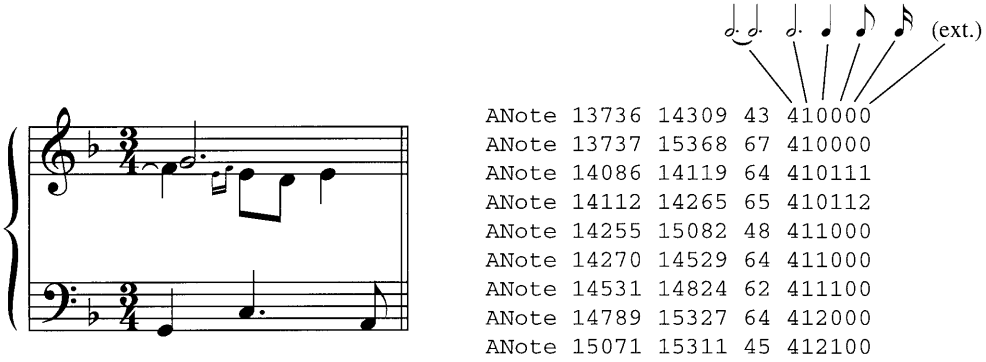


Figure 1.5: "An excerpt from Mozart, Sonata KV 332, I, showing note-address lists at right." [Tem04]

5- or 6-value "note addresses." Each note has a value representing the beat number at the various metrical levels, as well as the onset time and duration. His note address is an ASCII string of 5 or 6 digits, with each digit representing a metrical level. This encoding is unnecessarily limited—it would be better to just say that the representation is a vector of such and such values for each event. The note address system has other problems mentioned in the paper, and doesn't seem to be prepared for wide adoption. Even at a representational level, it is difficult to match all of the varied algorithms. Some systems are primarily concerned with beat induction at one level. Others are

concerned with metrical events at many metrical levels, including non-beat and extrametrical events.

As difficult as it would be to settle on a unified representation, there is nevertheless much value in a large and varied corpus that is tagged with a thorough representation. Studies could compare their results to the relevant subset of the gold standard representation, possibly after some processing to bring it into a compatible representation. This would be of value. This Temperley has done, providing a corpus that has been automatically analyzed but hand-verified using his note address system. However it is a limited corpus—only 46 excerpts from common-practice repertoire ("classical music"), most of it generated mechanically from notation, and that which is performed is only piano music. Although these are serious limitations, it may be useful as a common baseline in comparing algorithms.

Temperley also recommends a method for comparing analyses based on his corpus and goldfiles.

In conclusion, rhythmic analysis is an active field of research. Good results seem to be found for certain classes of music using various of the methods above, but all have their limitations. No method seems to be capable of tracking arbitrary rhythmic content in arbitrary music and genres, though some are more generally applicable than others. Although the problem is still far from solved, there are workable approaches with decent results that can be applied to many problems and situations where certain assumptions hold.

Most of the more recent studies have moved to probabilistic models based on low-level features in the audio (with or without the onset detection step). The simple approaches using amplitude envelopes and looking for repeating frequencies just don't cut it. Interestingly, the more sophisticated approaches tend to be based on more fundamental psychoacoustic and cognitive ideas—simpler in nature though more complicated in mathematical and implementation terms.

Chapter 2

Pitch

2.1 Introduction

Music consists of notes sounding through time. How high or low each note sounds is referred to as pitch. For example, the pitch named A just below middle C has a frequency of 440 Hz (this is known as concert pitch, and is the note that all instruments in an orchestra tune to before a concert).

The actual sound of instruments is not so simple, however. The reason each instrument sounds different is that it has a signature timbre. It has harmonic resonances, noise in attack transients, etc. However it is usually the case that the pitch we perceive for a sounding note is the fundamental frequency—the frequency with the most spectral energy.

The question of pitch is basically, which notes are sounding at any given moment? The difficulty is to accurately detect all sounding notes without detecting spurious notes that are actually just harmonics or noise. Other related questions are about the chord changes or musical key of a piece.

2.2 Transcription

Contrary to beat tracking, which we can do even without thinking, transcribing music from a recording to musical notation is very difficult for people to do. Very few people posess the gift of "perfect pitch" to name the pitch of a sounding note without external reference. With practice the ear can be trained for relative pitch, the ability to recognize an interval between two pitches, and therefore given a starting pitch be able to notate each pitch as it comes. Even musicians well-trained in transcription are limited to only a few voices of music at a time. That is, only a few simultaneously sounding instruments (or

simultaneous notes of a single instrument, like a piano). Identifying pitches precisely does not seem to be important to the enjoyment of music, although the correct performance of them is.

Perhaps it seems strange then that people hope and even expect to be able to do automatic transcription of arbitrary music using a computer. Perhaps it is exactly because this is difficult for us to do, and yet we see a simple if naïve approach: if you can recognize the fundamental frequency of notes then you can just keep track of when which note is sounding and you have transcription! Unfortunately in the real world it is not so easy. In the presence of complex and changing timbre, signal noise, and the spectral confusion due to polyphonic music (music with more than one sounding note at a time), the problem becomes very difficult.

Plumbley et al. [PAB+02] describe the approaches they have taken for the transcription problem. They recognize two classes of approaches: knowledge-based models and learning models. In a knowledge-based model, knowledge such as the physics of vibrating instruments is used to analyze the signal and answer questions. In a learning model, machine learning techniques are applied to adapt adjustable parameters in a system that allows us to extract knowledge from a system, e.g. the frequency content and volume of each note at each time. Machine learning approaches also use some knowledge about the music generation process, often called prior knowledge. The two overlap somewhat, differing primarily in the presence of parameters that are learned from given data.

Plumbley et al. describe a monophonic transcription system that uses autocorrelation together with onset detection and pitch quantization (to MIDI pitch numbers). Autocorrelation is a time-domain technique to detect repetition in the audio signal. The signal is divided into frames of size N, and autocorrelation is calculated as

$$r_{xx}(n) = \frac{1}{N} \sum_{t=1}^{N-n-1} x(t)x(t+n).$$

Autocorrelation measures similarity between shifted versions of the waveform. The delay corresponding to the highest peak gives the period of the waveform, i.e. the pitch of the fundamental frequency.

Brown and Zhang discuss a monophonic transcription system in [BZ91] that uses "narrowed" autocorrelation instead of the traditional autocorrelation. The narrowed autocorrelation function is presented in [BP89] by Brown and Puckette. Narrowed autocorrelation is defined as

$$S_N(\tau)^2 = \langle |f(t) + f(t-\tau) + f(t-2\tau) + \cdots + f(t-(N-1)\tau)|^2 \rangle$$

Where it would seem that the notation $\langle |x|^2 \rangle$ designates the inner product of x with x, i.e. $\langle x, x \rangle$. In the case of $N = 2$, we have traditional autocorrelation.

The narrowing allows for better frequency resolution, creating peaks that are easier to distinguish. One piano they recorded had fairly broad autocorrelation curves, leading to overlapping and the inability to distinguish two adjacent notes. Narrowed autocorrelation made transcription of this instrument possible.

When polyphonic music is considered, autocorrelation is no longer suitable. Instead we must look at the frequency domain, e.g. from a short-time Fourier transform. The fundamental frequency of a single note will have more spectral energy, its harmonics will have less energy, and the noise floor will have even less. During the attack phase of a sounding note, frequency content may be chaotic and high energy in many bands that don't correspond to a fundamental or harmonic of the sounding note. This is the transient phase discussed in the rhythm section. When multiple notes are present, there are often overlapping harmonics which makes it difficult to detect the difference between fundamentals and harmonics based solely on spectral energy, without more information.

Plumbley et al. [PAB+02] discuss two methods, one knowledge-based model using a blackboard system and one multiple-cause model which uses machine learning techniques.[1]

A blackboard system is like a room full of experts collaborating on a blackboard, where each expert pitches in when her area of expertise would help. It consists of a blackboard which contains hypotheses, and a set of knowledge sources (KSs) which are able to make inferences about some hypotheses. Each knowledge source is a kind of expert system. The blackboard is initialized with the information about the signal, e.g. the spectral energy at each frequency band. The KSs are consulted in turn, and one KS may recognize a pattern that may be a fundamental frequency and so updates the blackboard with this new partial hypothesis. Other KSs do the same, and later KSs may integrate the partial hypotheses from earlier KSs into a more confident hypothesis, and so on.

A multiple-cause model [Sau95] is a sort of neural network. It "searches for representations of the underlying causes of the input data, together with amounts of each cause, which take account of the input data as closely as possible." They trained a multiple-cause model with artificial clarinet, oboe, and trumpet sounds and found that it was correctly identify the instrument and note after training. Then they did the same thing with recordings of real instruments with similar but imperfect results. They observed that the output units of the network have found a sparse coding, i.e. a small number of measurement units represent each sound. Not all instruments were completely separted in this encoding. The flute and clarinet are poorly separated, probably because of the similar pure tone during the steady phase. Bello, Monti, and Sandler give the details of these approaches in depth in [BMS00].

[1] Appendix C is a primer on machine learning.

Finally, Plumbley et al. discuss a technique related to blind source separation. The blind source separation problem is to separate individual sources (e.g. a conversation at a cocktail party) using multiple observations in different locations (e.g. two ears). One approach to this problem is to use independent component analysis (ICA). A typical ICA method first removes covariance from the observations by *whitening*, then rotates the axes until the outputs become independent. They apply this in a novel fashion to the transcription problem, by considering each pitch to be a source. The ICA basis will find the spectral shape of each note and each spectral shape should be different (even if some are just frequency-scaled versions of others). Because most notes are not sounding at any given time, sparse coding is used to find a sparse set of independent notes that explain the signal. They have good results transcribing a 3-voice Bach partita on piano with this method. 49 note spectra were learned. One disadvantage is that notes are not arranged by pitch, that is it knows note 48 and note 47 are different but does not know enough to assign the pitches (they are not necessarily adjacent, either). They reordered the sources by hand to create a piano roll representation, but believe that automatic reordering is a possibility in the future. Abdallah and Plumbley discuss the particulars of this method in depth in [AP01].

Cemgil, Kappen, and Barber developed a generative model for transcription [CKB04]. A generative model is a probabalistic approach that generates prior probabilities by simulating generation of music. That is, given a specific transcription, the probable audio signal is generated. The transcription that generates the model most closely matching the observed signal is the most likely correct transcription.

This system is a dynamical Bayesian network. The piano roll transcription defines the currently sounding notes which, combined with the current state of the model, define the generated audio. Each step also affects the next step (i.e. the piano roll at step k has as a subsequence the piano roll at step $k-1$). This Bayesian network graph allows us to calculate the probability of each piano roll using Bayes' theorem: $P(r|x) \propto P(x|r)P(r)$. The likelihood $P(x|r)$ is given by the generative model. They use a time-domain model based on additive synthesis with decay. The parameters of this model can be learned with machine learning techniques, or specified a priori. The prior $P(r)$ allows the introduction of musical knowledge, such as knowledge about likely intervals or less-likely chords. This allows the integration of high-level knowledge of musical structure and low-level acoustic analysis.

Although the generative model is time-domain based, it is related to the frequency domain in that it generates a signal with harmonics and decay, in a sense similar to comparing the spectral footprint of an instrument, but doing so generatively from the time domain. It is these parameters of decay, harmonics, and so forth that can be learned.

Direct calculation of the posterior probability is prohibitive, so they describe various approximation techniques that keep the processing down to polyno-

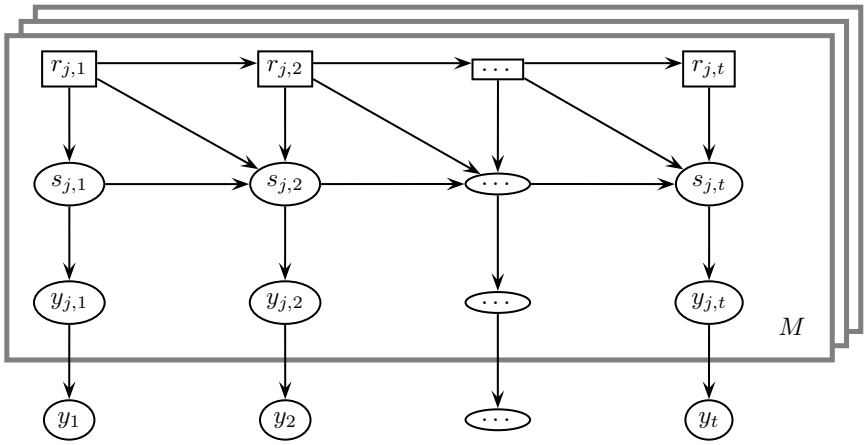


Figure 2.1: "Graphical Model. The rectangle box denotes "plates", M replications of the nodes inside. Each plate, $j = 1, \ldots, M$ represents the sound generator (note) variables through time" [CKB04]. $r_{j,k}$ are piano rolls, $s_{j,k}$ are states, and $y_{j,t}$ are the sound generators.

mial time. The algorithm is causal and sample-accurate. That is, the audio is processed sample-by-sample instead of frame-by-frame. This may not be important, as frame sizes tend to be well within the smallest rhythmic period we distinguish, but the authors felt it was significant.

Raphael [Rap05] describes work that explores automatic transcription of sung melodies using a joint model of pitch, rhythm, tempo, and segmentation. The work incorporates musical knowledge—e.g. rhythmic information—into pitch detection, and vice versa. The model is restricted to melodies in 3/4 time quantized at the eighth note level. The fragment is represented as a list of onset positions (an onset either of a note, or of a rest). This is modeled as a Markov chain. A Markov chain is a sequence of random variables with the property that given the present, the future is independent of the past ($P\{X_t = j | X_0 = i_0, X_1 = i_1, \ldots, X_{t-1} = i_{t-1}\} = P\{X_t = j | X_{t-1} = i_{t-1}\}$). "We simultaneously estimate the partition of audio data into notes, and the labeling of the notes with pitches and rhythmic values that make sense within the metric context. We also simultaneously estimate a (potentially) time-varying tempo process." They calculate the global configuration of these parameters that is an optimal estimate using dynamic programming. Each decision at each rhythmic position is a branching point, and dynamic programming is used to keep the tree pruned. When the end is reached, the optimal set of parameter (rhythm, pitch, onset times, tempo) sequences according to the model is resultant. They added one refinement using the musical key. Initially the prior probability of any note is uniform, but they detect the likely musical key

signature and armed with the knowledge of the tonic use a more appropriate prior (e.g. notes in the key's scale are more likely).

2.3 Other Applications

There are other applications than transcription to which pitch and frequency information can be applied. We have seen one such example in the rhythm chapter with [Dan05], where chroma vectors were used to find self-similar sections of music in order to infer song structure.

Bartsch [BW05] also uses chroma vectors (he calls them chromagrams) to find self-similar song sections in order to extract an "audio thumbnail"—a representative sample of the song, e.g. from the chorus or refrain.

He relates the origin of the chroma formulation: "In the early 1960s, Shepard reported that two dimensions rather than one are necessary to represent the perceptual structure of pitch [She64]. He determined that the human auditory system's perception of pitch was better represented as a helix than as a one-dimensional line, and coined the terms *tone height* and *chroma* to characterize the vertical and angular dimensions, respectively." (See Figure 2.2) One way

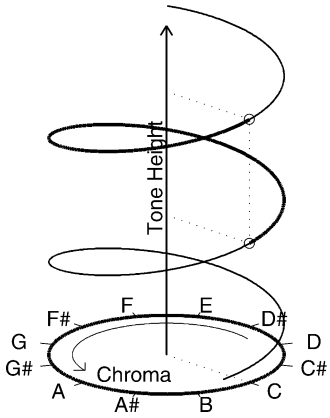


Figure 2.2: "Illustration of Shepard's helix of pitch perception. The vertical dimension is tone height, while the angular dimension is chroma." [BW05]

to calculate chroma is $c = \log_2 f - \lfloor \log_2 f \rfloor$, i.e. the fractional part of the base-2 logarithm of frequency. The chromagram, then, is the time-chroma distribution: the "joint distribution of signal strength over the variables of time and chroma." As with chroma vectors, they discretize the chroma dimension so that there are 12 chroma bands, one for each semitone in the equally tempered musical scale.

After the chromagram is computed, he calculates the self-similarity matrix as $C_{i,j} = v_i^T v_j$, i.e. dot product between the chroma vector at time i and the chroma vector at time j. The musical repetition is visually apparent in this matrix as diagonal lines of high correlation. (See FIgure 2.3) The matrix

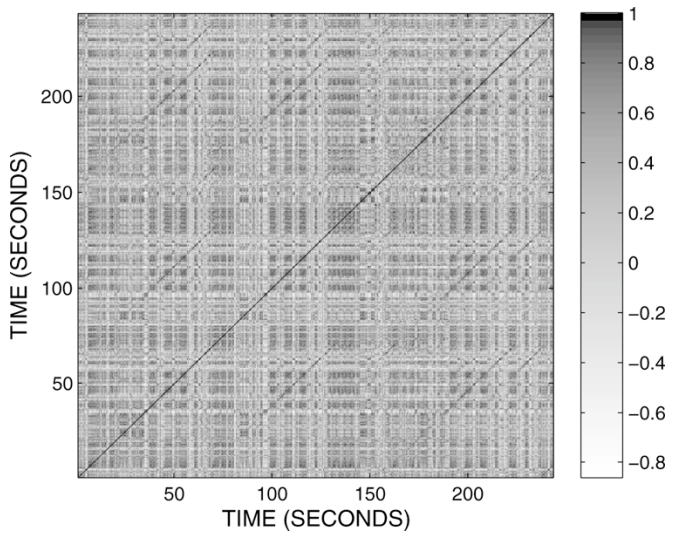


Figure 2.3: "Similarity matrix C for Jimmy Buffet's "Margaritaville," showing the correlation between the features for individual frames of the song." [BW05]

is manipulated in order to facilitate extraction of the information, and the thumbnail is chosen as the segment of the song (of a predefined length) most strongly repeated. The selections show good correlation with hand-chosen selections for music that fits the assumption that there is a repeated section that is a good choice for the thumbnail.

Shenoy and Wang explore a holistic problem in [SW05]. They identify the musical key, the chord progressions, and rhythmic content of popular music. Their approach is a rule-based one, rather than the more common probabilistic and learning approaches.

First a beat detection algorithm using sub-band processing is applied. Salient events are detected based on sub-band intensity. Then IOIs are clustered and beat tracking is done as described in the rhythm chapter. The tracked beats are used to segment the audio, and a chroma-based feature extraction is performed. Chord detection and key determination are based on the chroma information, with extra weight given to primary chords in each key (tonic, dominant, and subdominant). The algorithm up to this point is described in further detail in [SW04].

The chord detection in [SW04] is sufficient for determining the musical key,

but is not accurate enough to stand on its own as a result. So Shenoy and Wang apply further rules for checking and enhancing the chord detection. First, they eliminate chords that are not in the key of the song. They recognize that this is a simplifying assumption, as extra-key chords are possible though less-common. Second they adjust inconsistent or missing chords when the frame is surrounded by the same chord on either side. This corrects a common error where the relative minor chord is detected erroneously between correct detections of a major chord, and other errors of the same nature.

Next the system determines a rhythmic structure based on the chord information extracted so far. Chords are more likely to change at the beginning of a measure than at other positions. The music is assumed to be in 4/4 time, as this is by far the most common meter in popular music. They check for patterns of four consecutive frames (beats) with the same chord, and consider all such patterns as possible measure boundaries. Not all such patterns will be measure boundaries, though, so they are checked against eachother for consistency and the measure phase is chosen. Then the measure boundaries are tracked and interpolated throughout the song.

Now armed with measure boundary information, they perform another chord detection enhancement. If three chords in a measure are the same, then the fourth chord is likely to be the same as the others. If a chord is found in both halves of a measure, then it is likely the other chords in the measure are also the same.

Their results on 30 rock songs spanning 35 years are encouraging. The key and measure detection was correct in 28 out of 30 songs, and the average chord detection rate was 79%. However the restrictions on the algorithm cannot be ignored. The music must be 4/4, and there is probably a bias towards uncomplicated chord progressions. More complex music, such as jazz (even the 4/4 variety) would probably pose a considerable challenge. It seems that rule-based systems tend to be specific and fragile.

2.4 Conclusion

There are many questions that can be asked relating to the pitch of notes in music, but the most common and most focused-on question is simply, "which notes are playing?" Fairly reliable transcription from monophonic music is possible, but transcription from polyphonic audio is much more difficult, and an open problem. On the other hand, it is relatively straightforward to detect pitch and frequency information such as chroma, which has been used with great success in holistic approaches to beat tracking and music understanding problems.

Chapter 3

Timbre

Timbre is defined as "the character or quality of a musical sound or voice as distinct from its pitch and intensity." Timbre is a nebulous and somewhat mysterious concept. We're not really sure of what consists our perception of timbre, though we know that the spectral signature and the attack phase are important factors. Timbre is the difference between a trumpet and a violin, the difference between two voices, the difference between a trumpet with a mute and without.

By far, the most popular application of timbral analysis is instrument identification. There is also interest in instrument separation—e.g. producing one soundfile for each instrument in an ensemble performance—but this is considerably more difficult. I will present a number of studies on instrument identification. They almost all take the same basic machine learning approach[1]. They extract a number of features (often a great number) from the audio signal, then they train one or more classifiers which classify the instrument based on the features. Often, they have a selection step, where features and/or classifiers are evaluated for efficacy and the feature and/or classifier sets are pruned without loss of accuracy. Sometimes the intent is to find out which features and classifiers matter and therefore gain insight into the process and perhaps our own cognitive and perceptual processes at the same time. Other times, it is simply to reduce computation to allow the system to run in real time.

In [HABS00] and [HBPD03], Herrera-Boyer et al. review instrument classification systems. Eronen also gives an extensive literature review in his master's thesis on instrument recognition [Ero01]. I will discuss the aspects of the basic algorithm (features, classifiers, and feature/classifier selection), then discuss any remaining novelties of each approach.

[1] Appendix C is a primer on machine learning concepts used here.

The problem of instrument identification can be subdivided into two main problem classes. First, that of identifying an instrument from a monophonic recording—i.e. one in which there is just one instrument playing. This problem seems solvable, and many studies have done quite well. The second class, which is considerably more difficult, is to identify instruments in polyphonic (ensemble) recordings, and to go even further, to annotate when each instrument is playing or not. It is not at all clear whether research directed at the first (monophonic) problem will be of any help in the second (polyphonic) problem [HBPD03]. The majority of research performed to date has focused on the monophonic problem. Conceivably, if a successful algorithm were developed for instrument separation then it could be applied to an ensemble recording first, and then a monophonic instrument identification algorithm could be applied. Unfortunately separation is also a difficult problem.

In [HBPD03], Herrera-Boyer et al. distinguish between "taxonomic" and "perceptual" instrument recognition problems. Taxonomic systems classify instruments according to an instrument taxonomy, e.g. the violin is a string instrument is a sustained instrument, etc. Perceptual classification does not apply a cultural label but instead compares one sound to another, perhaps giving a kind of distance function or clustering. A perceptual classifier might be able to pick out other piano recordings given a piano recording as input, but would not necessarily know to label them as "piano". Figure 3.1 is an example taxonomy.

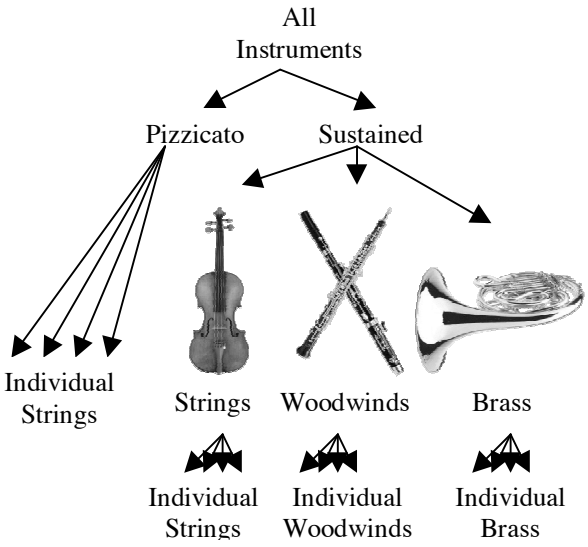


Figure 3.1: "A possible taxonomy of orchestral instrument sounds." [MK98]

They describe several studies where participants rated the similarity of pairs

of sounds, and a technique called Multidimensional Scaling (MDS) was applied to get a low-dimensional "timbre space" (generally two or three dimensions). Qualitatively, one such study assigned as meanings one dimension to spectral energy distribution, one to synchronicity of transients and spectral fluctuation, and one to the temporal attribute of the beginning of the sound. We can draw some vague qualitative conclusions about what timbre is, or might be, from these results. This vagueness, and lack of understanding about our perception of timbre, is a primary motivator in the almost universal choice of machine learning techniques in instrument classification.

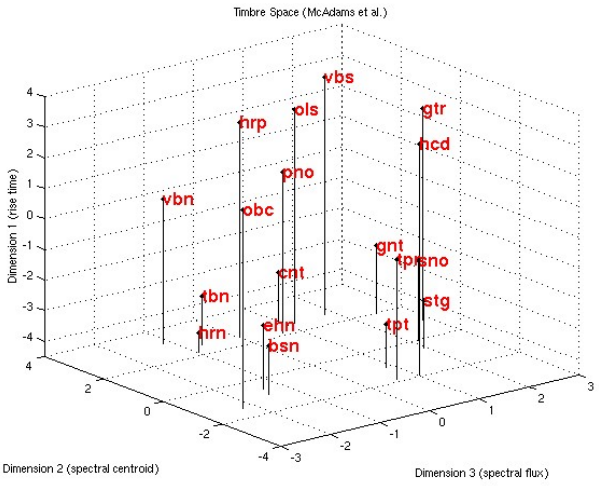


Figure 3.2: "Timbre space coming from the McAdams et al. (1995) experiment." [HBPD03]

3.1 Features

There are a great many features that have been proposed and used for instrument classification. It is not uncommon for a study to consider 50–200 features, pruned down to a reasonable set using a feature selection algorithm. Herrera et al. believe that more than 15–20 features after selection would be suboptimal, and most feature selection experiments seem to bear this out. Too many features is not only computationally expensive, but gives suboptimal results. Pruning unneeded features can sometimes improve the accuracy of classification, on the order of 10% [HBPD03]. Unfortunately, the optimal features are often not what we would expect, and features we may expect to do well do poorly. Also, the relevant features often vary depending on the context. Some features are more relevant for recognizing some instruments,

while other features are more relevant for other instruments. Even the listener (or microphone) position can influence timbre, as evidenced in [SOŠ03], where they objectively and subjectively evaluated the difference in timbre of the pipe organ according to where the listener is in the hall. They found the differences to be substantial.

I cannot hope to enumerate all of the myriad features, but we will look at a representative subset.

Amplitude

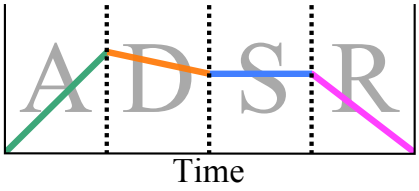


Figure 3.3: ADSR Envelope (Wikipedia)

The attack phase of a note provides vital cues to the listener. Two instruments may be very similar in the steady state, and difficult for people to distinguish in the absence of the attack. This is a well-known effect among electronic musicians, where a very popular amplitude envelope for sound synthesis is the piecewise-linear ADSR envelope (attack, decay, sustain, release). Changing the attack and decay times alone very noticeably changes the perceived timbre of the instrument.

Unfortunately, attack alone is not enough to characterize an instrument, nor is it always a reliable feature in and of itself. The attack of many instruments depends on the volume, the manner of playing and even the note being played.

In addition to the envelope of a sound, there are other amplitude cues we can take. Very loud sounds and very soft sounds are uncharacteristic of some instruments. The dynamic range of the trumpet is quite different to that of the bass, for example. Some wind instruments exhibit tremolo, which is the periodic variation of amplitude (compare with vibrato, which is the periodic variation in frequency around the core frequency).

Amplitude features are used by [Ero01, HDG03].

In addition to amplitude per se, energy features have been used as well, including total energy, harmonic energy, and noise part energy [LR04].

Temporal

Although the frequency domain is the most prolific source of features, some useful features can be extracted from the time domain as well. The most popular of these is the zero-crossing rate (ZCR), which is the rate at which the waveform changes from positive to negative values. Temporal features are used by [HDG03, LR04, KGK+05].

Frequency

Instruments have different frequency ranges. Notes in the lower range of a bass cannot be produced by a piccolo, for example. Thus if we find the fundamental frequency (sometimes denoted F0 or f_0) we can rule out or assign higher probability to instruments based on their range. This is an example of incorporating high-level knowledge, although the same could also be learned by the system. Studies using fundamental frequency features include [Ero01, KGK+05].

Instruments played with vibrato have periodically varying frequency. A vibrato detection feature can steer us toward the violin and away from the piano. The depth and period of the vibrato may also be useful features.

Spectral Shape

Spectral shape is general term for the relative amplitudes of each frequency partial in a spectral analysis. When normalized around a fundamental frequency, the partials generally correspond to the harmonics. There are many features that characterize spectral shape in one way or another. For example, the spectral centroid is analogous to the mean of a normal distribution, and the spectral spread is analogous to the standard deviation of a normal distribution. These features characterize the brightness and richness of a tone. Spectral centroid seems to be one of the most salient features. Spectral centroid and/or spread is used by [Ero01, HDG03, LR04, KGK+05].

Spectral flatness is the ratio between the geometrical mean and the arithmetical mean of the spectrum. This gives an idea of the overal flatness of the spectrum—the more flat it is, the more "white-noise-like" it is. [HDG03]

Spectral skewness and other higher-order statistics are investigated on the understanding that partial spectra are not normally distributed. [LR04]

The temporal variation of spectral shape is often calculated in one form or another. Some common names for this are *spectral flux*, *spectral irregularity*, *spectral fine structure*, and *spectral smoothness*. "Irregularity of the spectrum can indicate a complex resonant structure often found in string instruments" [Ero01].

A related technique is to look at how the spectral shape changes as the tone evolves. During the onset, the spectral shape can indicate the nature of the initial excitement of the instrument. In the transition between onset and steady state, the rise rate of the various harmonics can be observed, and this can be indicative of the instrument. Examining the steady state reveals the harmonics of the note. In order to do this analysis effectively, we need to know the onset and duration times, or guess from the spectral content of a frame that a transition has occured.

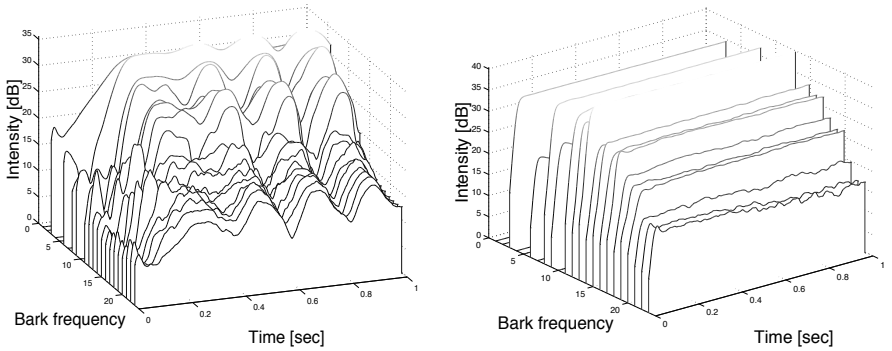


Figure 3.4: "Sinusoid envelopes for flute (left) and clarinet (right), playing the note C4, 261 Hz." [Ero01]

Sinusoid envelopes are related to the Fourier transform, but bins are calculated separately with indpendent time frames, which eliminates the need for windowing and gives a sample-by-sample amplitude envelope. Sinusoidal envelopes have high perceptual fidelity, and are a useful basis for examining onset asynchrony. Their use is limited by difficulty in reliably tracking the fundamental frequency and partials. Eronen uses sinusoid envelopes, but believes that for the future, "using a filterbank instead of sinusoid envelopes would be a simpler and more robust approach" [Ero01].

In [MB96], Masri and Bateman describe an analysis and resynthesis system where the audio frames are aligned with onsets in order to avoid smoothing of the attack during resynthesis ("diffusion"). They detect transients with an algorithm that looks for sudden change and increase in energy, using a spectral energy feature that is linearly weighted to favor high frequency content. They then synchronize the analysis such that the STFT window never crosses an onset boundary. This same technique could be applied to synchronizing analysis of the evolution of spectral shape.

Harmonics and Inharmonics

Related to spectral shape, harmonics-based features look at the relative amplitude of harmonics. A common feature is the odd-to-even ratio of harmonics

[LR04, KGK+05]. Some instruments have primarily odd harmonics with few even harmonics, and others have the opposite. [KGK+05] uses various other harmonic-related features.

Sound from acoustic instruments is rife with inharmonics, i.e. noise components that have no harmonic relationship to the fundamental frequency. This is especially true during the attack transient. The most apparent noise is generally a byproduct of the initial excitation, e.g. the sound of fingers on guitar strings, or due to the manner in which the instrument sustains, e.g. breathiness in a flute sound. In addition there are less-obvious inharmonics that add to the timbre of any instrument.

Aside from transient features, inharmonicity has not been applied to musical instrument recognition [Ero01]. Martin and Kim propose a possible inharmonicity feature, but did not implement it [MK98].

Cepstral Coefficients

The *cepstrum* of a signal $\boldsymbol{x}$ is

$$\boldsymbol{c} = \mathcal{F}^{-1}\left(\log|\mathcal{F}(\boldsymbol{x})|\right)$$

where $\mathcal{F}$ indicates the discrete Fourier transform. So the cepstrum is essentially taking the Fourier transform of a log spectrum. The cepstrum can be seen as information about the rate of change of the spectrum. When applied to audio, the spectrum is usually transformed using the Mel frequency bands, giving the Mel Frequency Cepstral Coefficients (MFCCs). The mel scale is a logarithmic scale where intervals are perceived as equidistant by listeners. It is defined as

$$m(f) = 2595\log(1 + f/700).$$

MFCCs are very popular features. They are used by [Ero01, HDG03, LR04].

Trivia: The word cepstrum is pronounced with a hard c and is an anagram of spectrum, and its independent variable is *quefrency*—an anagram of frequency.

Bark Energy Ratios

Bark bands are approximations to the first 24 critical bands of human hearing. Herrera et al. add two more bands for low frequency resolution. They extract a number of features from the Bark bands [HDG03].

Equivalent Rectangular Bandwidth

Vincent and Rodet [VR04] use the equivalent rectangular bandwidth (ERB) scale, which is similar to Bark bands and the mel scale. It is defined as

$$f_{\text{ERB}} = 9.26\log(0.00437 f_{\text{Hz}} + 1).$$

They also take the log of the power to better fit human perception.

Log-lag Correlogram

In [MK98], Martin and Kim use the log-lag correlogram as a representation instead of the STFT. The correlogram is "not based on the assumption that the signal is periodic; it may therefore be better suited than previously studied representations to analysis of inharmonic signals."

To calculate the correlogram, the signal is first filtered and separated into channels, in a manner modeled after the cochlea and inner hair cells. Then autocorrelation is run on each channel with logarithmic lag spacing.

The three dimensions of the correlogram are cochlear position (a frequency, in Hz), autocorrelation lag (logarithmic representation of periodicity), and time. It is usually visualized as a snapshot in time with cochlear position on the vertical axis and lag on the horizontal, as in Figure 3.5.

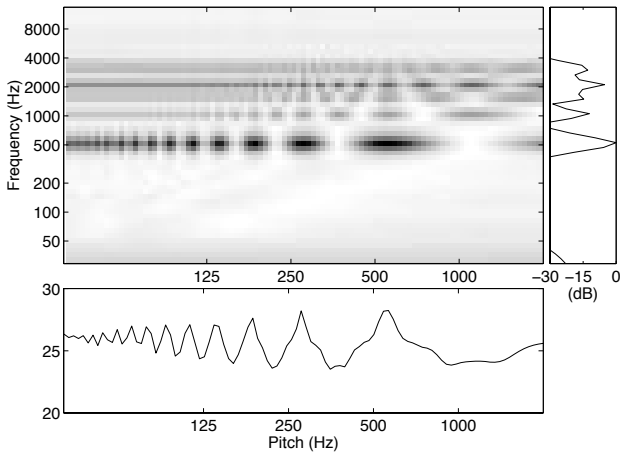


Figure 3.5: "Correlogram snapshot of a violin tone. The horizontal axis, labeled "pitch," is the inverse of autocorrelation lag. The vertical axis, labeled "frequency," corresponds to cochlear position. The lower panel displays the summary autocorrelation (the correlogram integrated over the cochlear dimension). The right-hand panel displays the zero-lag energy, which for isolated periodic sources is equal to the spectral envelope." [MK98]

As with the spectrogram, many features can be extracted from a correlogram. Martin and Kim extract 31 features including pitch, spectral centroid, onset asynchrony, ratio of odd-to-even harmonic energy, and the strength of vibrato and tremolo.

3.2 Classification

Classifiers classify the signal based on the features that were extracted. During the training phase, they are given training examples and usually a label for classification (this is supervised learning). Sometimes the learning is unsupervised, i.e. there are no labels. In those cases the classifier clusters examples automatically according to likeness, but does not apply labels. Most classification systems use supervised learning.

k-Nearest Neighbor

The *k*-NN algorithm stores feature vectors of all the training examples, and when asked to classify a new instance it finds the *k* nearest training examples (neighbors) in the feature space. The new instance is then classified in the class that contains the most of the neighbors. *k*-NN is popular because it is easy to implement.

However, there are some significant drawbacks. It is a lazy algorithm, meaning all it does during the training phase is store the training examples for later. Thus, all the training examples need to be in memory. Finding the *k* nearest neighbors for each query can be computationally expensive as well. In addition, it is sensitive to irrelevant features, which can dominate the distance metrics.

Herrera-Boyer et al. [HBPD03] reviewed the use of *k*-NN, and based on the results obtained by researchers estimate that the algorithm is limited to about 80% accuracy. Indeed, almost every study that uses *k*-NN uses it as a prototype or just includes it for completeness. It is often outperformed by other classifiers in the study.

Studies employing *k*-NN classification include [MK98, Ero01].

Naïve Bayes

The naïve Bayesian classifier calculates the maximum a posteriori (MAP) hypothesis (the most likely hypothesis). Let $a_1, a_2, \ldots, a_n$ be the n features under consideration, and V is the set of classes. Then we wish to find

$$v_{\text{MAP}} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j | a_1, a_2, \ldots, a_n).$$

Using Bayes' theorem and assuming independence of the features, we can rewrite this as

$$v_{\text{NB}} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j).$$

$P(v_j)$ is the prior probability of the classification v_j, and $P(a_i|v_j)$ is the probability of seeing feature a_i given classification v_j. In essence, we count the feature attributes seen in each class while training and use those counts to calculate the posterior probability.

Naïve Bayes is naïve because of the assumption that the features are independent, which they almost certainly are not. However, the approach works well in practice. [KGK+05] uses naïve Bayes inference.

Discriminant Analysis

Discriminant analysis provides linear, quadratic, or logistic (s-curve) functions of the variables that best separate the data into two or more predefined groups. Martin and Kim use multiple discriminant analysis at each decision point in the taxonomy to reduce the high-dimensional feature space (31 features) to a space of fewer dimensions (the number of data classes minus one), greatly reducing the amount of training data needed [MK98]. The actual classification is done with a naïve Bayesian classifier. See Figure 3.6 for a visual example of LDA.

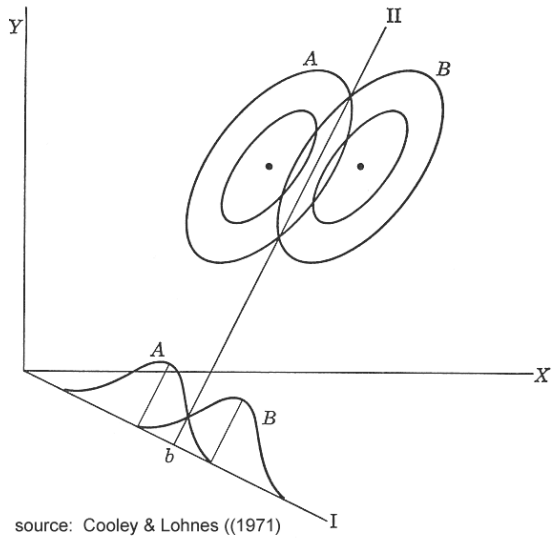


Figure 3.6: An example of linear discriminant analysis. The data is in the X, Y plane, and LDA finds the line b (the II axis), which allows projection onto the I axis.

Herrera et al. note that "surprisingly, there have been very few studies using these techniques" [HBPD03]. In an earlier paper, they give a possible reason for this:"perhaps it is commonly assumed that the classification problem is

much more complex than that of a quadratic estimation, but it means taking for granted something that has not been experimentally verified, and maybe it should be done" [HABS00]. Herrera et al. use canonical discriminant analysis as a classifier in [HDG03].

Decision Trees

A binary decision tree is constructed by first considering that binary (yes/no) feature (at the root of the tree) that gives the most information (i.e. maximally reduces entropy). This process continues recursively, and then the tree can be pruned to avoid overfitting the data. Classification is performed by traversing the tree and finding the answer at the leaf node. Building a binary tree is faster than training a neural network [HBPD03].

The C4.5 algorithm is a decision tree algorithm that tries to ignore irrelevant features and focus on relevant ones. Herrera et al. [HDG03] use C4.5 and a variant called PART (an acronym for *partial decision trees*). "It yields association rules between descriptors and classes by recursively selecting a class and finding a rule that "covers" as many instances as possible of it. The models derived by PART usually contain fewer rules than those generated by C4.5, and are easier to interpret."

Support Vector Machines

Support Vector Machines (SVMs) [DSN01, HABS00, HBPD03] find the optimal linear hyperplane that minimizes the expected classification error. They are based on the principle that a function that classifies the training data correctly and has the lowest complexity will generalize better than functions with higher complexity.

When dealing with data which is not linearly separable, SVM can nonlinearly map the data to a high-dimensional feature space where a linear hyperplane can be found. The function performing this mapping is called a kernel function.

Some drawbacks to SVMs include, the kernel function found is not guaranteed to be optimal, it is computationally expensive, and the classification is binary. Binary classification can be worked around by performing a concatenation of binary learning procedures.

Artificial Neural Networks

An artificial neural network (ANN) is a network of "neurons" organized into a graph with at least two layers: input and output. The input nodes are fed with

features and in the case of classification the output nodes generally correspond to classifications. In addition there may be one or more hidden layers in between. A node takes many inputs and produces a single output, which is then possibly fed as input to other nodes and so forth. The behavior of each node is determined by the weights given to its inputs, and these weights are learned during the training phase.

ANNs are very popular. They perform well, are robust to noisy training data, and evaluation is quick. On the other hand, training an ANN can take a long time, and the trained network is a black box—it does not reveal any meaning or structure that we can make sense of.

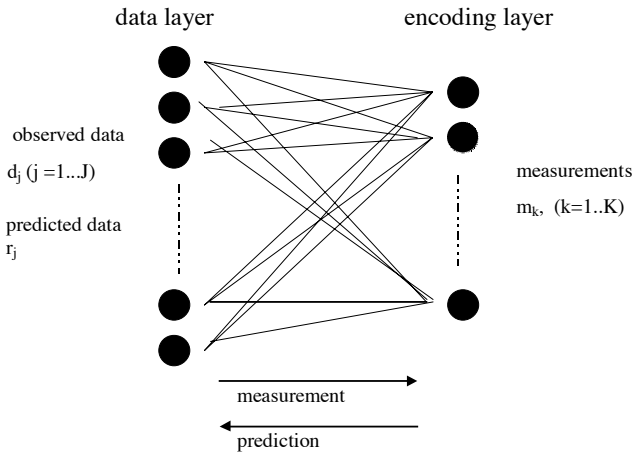


Figure 3.7: "Multiple cause model architecture" [KP00] Note the biderctional (two-step) flow of processing. There are J nodes on the left and K nodes on the right.

Klingseisen and Plumbley [KP00] use a variation on Saund's Multiple Cause Mixture Model [Sau95], which is a two-layer neural network. The J observations are fed into the data layer, and the K measurements are read on the encoding layer. Training is done differently, though. The network is fed forward, and then fed backward providing predicted observations on the data layer. The prediction is compared with the observations and the weights are adjusted to minimize error. Saund's model is binary, so observations and measurements are either on or off. Klingseisen and Plumbley modified the model to use continuous data and measurements. The input data is spectral magnitude and the output is a related to the probability of membership to class k. This is an unsupervised learning algorithm, so the correspondence of each output m_k needs to be associated with the corresponding class separately (this can be done automatically).

Rough Sets

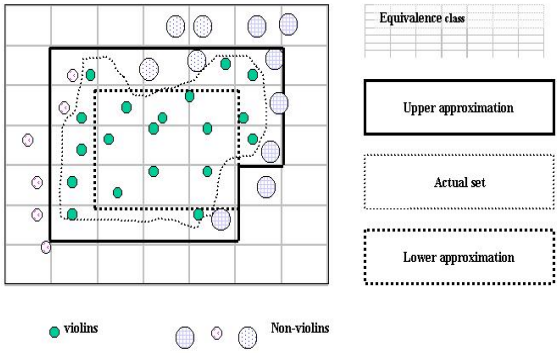


Figure 3.8: "An illustration of rough sets concepts" [HBPD03].

Rough sets can be used to reason about vague concepts which cannot be characterized in terms of information about their elements. An upper approximation and a lower approximation are found, and the region between them is the boundary region of the concept. The lower approximation of the set contains all objects that are definitely in the set, and the upper approximation contains all of the objects that may belong to the set. Probabalistic inferences about membership can then be made. [HBPD03]

Hidden Markov Models

A Markov model is a Bayesian network of states. At each time step, the current state is changed according to the probabilities along the edges from that state to other states, and an observation is generated according to the observation probabilities for the current state. In a hidden Markov model (HMM), the states, state transition probabilities, and observation probabilities are hidden, and only the observations are available. HMM methods such as Expectation Maximization (EM) learn an estimate of the state parameters [GH96].

The number of states is a regulation parameter that is not estimated. It is a tradeoff between not enough states and too many states (which leads to overfitting).

For classification an HMM for each instrument is considered and the model that gives the highest probability of observing the signal is chosen.

Gaussian Mixture Model

A Gaussian mixture model (GMM) is a linear combination of Gaussians. In instrument classification, a GMM can be used to model the spectrum of an

instrument, or with similar bases like the constant-Q transform (a transform related to the Fourier transform but which gives log-frequency bins, which closer matches human perception). The parameters of the component Gaussians are estimated using a maximum likelihood (ML) process which maximizes the likelihood of the GMM given the training data. The ML estimates of the model parameters are iteratively estimated using the EM algorithm [Ero01]. The EM algorithm is also used in estimating Kalman filter parameters. It is described in detail in [GH96]. Ribeiro gives a tutorial on Gaussians in [Rib04].

Independent Subspace Analysis

Linear independent subspace analysis (ISA) explains observed polyphonic power spectra as a combination of typical power spectra and additive Gaussian noise. This gives the generative model

$$\boldsymbol{x}_t = \sum_{h=1}^{H} e_{ht} \boldsymbol{\Phi}_h + \boldsymbol{\epsilon}_t$$

where $\boldsymbol{x}_t$ is the polyphonic power spectra at time t, $\boldsymbol{\Phi}_h$ is a normalized typical spectra, H is the number of typical spectra, e_{ht} is the scaling of typical spectra $\boldsymbol{\Phi}_h$ at time t, and $\boldsymbol{\epsilon}_t$ is Gaussian noise. Essentially, it is a sum of typical spectra at different amplitudes, plus noise.

Vincent and Rodet [VR04] develop a nonlinear variation of ISA. They observe that it is best to model noise as multiplicative (additive in the log-power domain) instead of as additive on the power spectrum. "Experiments show that the absolute value of [the noise] is usually correlated with [the polyphonic power spectra.]" They use fixed $\log(\cdot)$ and $\exp(\cdot)$ nonlinearities to transform the power spectra into log-power spectra and vice versa. In doing this they introduce variation spectra that model variations in the spectrum around the typical spectra, variation scalars which modify the variation spectra, and the power spectrum of the stationary background noise.

These parameters, along with the usual ISA parameters, are learned by maximizing the likelihood of the probability of observing the spectra given a set of instruments (the orchestra). However the direct computation is intractable so estimation techniques are employed.

3.3 Selection

It is common to extract numerous features, and sometimes try several classifiers, and analyze the results to determine which features and classifiers are the most salient. There are several methods for doing this.

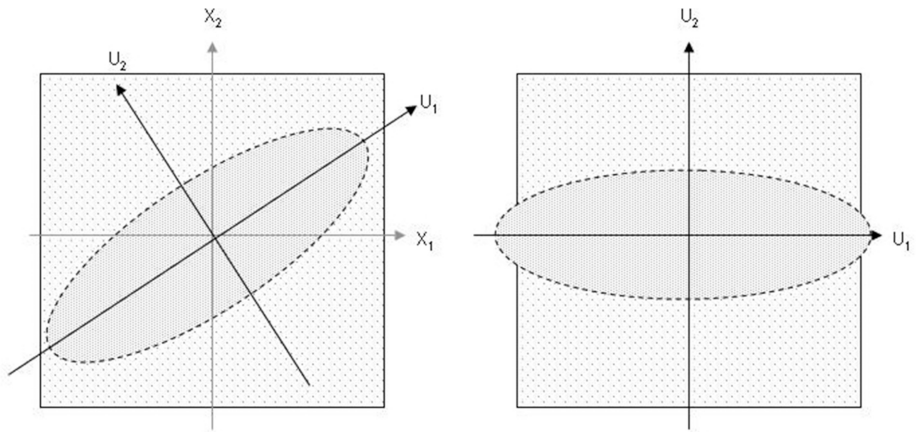


Figure 3.9: "The left depicts two-dimensional data concentrated in an ellipse. The right shows the data rotated according to its two principal axes, computed using principal components analysis." [CFPT06]

One technique is pricipal component analysis (PCA). PCA finds the eigenvectors of the covariance matrix and projects the data onto the n principal eigenvectors (those with the highest eigenvalues). This can be used to reduce the dimensionality while keeping the information loss at a minimum. Smith gives a tutorial on PCA in [Smi02].

Another technique is discriminant analysis, described above. In instrument recognition, discriminant analysis is used in this capacity more frequently than as a classifier. Kitahara et al. [KGK+05] employ PCA and then linear discriminant analysis (LDA) to select important features in their mixed-sound templates.

Sequential forward generation (SFG) and sequential backward generation (SBG) start with none or all of the features and add or remove features one at a time, respectively. The feature to be added or removed is the most or least relevant feature. Eronen reports that SFG tends to converge to a suboptimal solution, and that SBG gives better results [Ero01]. SBG will never do worse than all of the features, but that may not have been optimal to begin with (irrelevant features can worsen performance). SBG will often not give the minimal set of features.

Livshin and Rodet [LR04] use a technique similar to SBG that they call gradual descriptor elimination (GDE). It uses linear discriminant analysis to choose the most irrelevant feature which is then removed. This process is repeated until all features are removed, and the system recognition rate is estimated at every step.

3.4 Conclusion

Herrera, Dehamel, and Gouyon [HDG03] quite successfully classify percussive instruments. They also are able to classify percussion (snare, kick, tom, hihat) by manufacturer with high accuracy. They use MFCC- and Bark-based features, along with features like spectral centroid and zero crossing rate.

Livshin and Rodet [LR04] consider not just one-note samples but longer "continuous" ecological recordings (solos). They used over 60 features, and pruned that down to 20 features for real-time operation using GDE. They experimented on polyphonic music, recognizing the dominant instrument in a duet, both with unmodified signal and with a signal where one instrument has been attenuated.

Vincent and Rodet's [VR04] nonlinear ISA classifier does well on monophonic tests even in the presence of reverberation and noise, and on preliminary difficult polyphonic tests.

Kitahara et al. [KGK+05] aim to address three issues they see with polyphonic instrument recognition: the problem of feature robustness, the problem of pitch-dependent timbre, and the desire to make use of musical context. It would seem that only the first is unique to polyphonic music. To address the first issue they train on "mixed-sound templates," representative polyphonic samples that have been labeled with onset, duration, pitch, and instrument information. The signal goes through a harmonic structure extraction and features are applied, then they use PCA and LDA to determine the robustness of features. Thus they find features that are relevant in polyphonic music, and they train on common combinations of instruments which gives better results than relying on current instrument separation technology. It appears that their classification algorithm also requires labeling of notes, onsets, and durations. If that is the case, it would have to be coupled with an effective transcriptor to be of use (hand-labeling would not be practical—those doing the labeling could identify the instruments as well).

To address pitch-dependent timbre, they developed a F0-dependent multivariate normal distribution model. It has two parameters: the F0-dependent mean function $\mu_i(f)$, and the F0-normalized covariance Σ_i. A Bayesian decision is made based on these parameters and the prior probability, which is influenced by the musical context step.

To account for musical context, they make two passes. In the first pass, the prior probabilities are uniform, and on the second pass the posterior probabilities from the first pass are used to calculate a prior probability that takes neighboring notes into account. That is, if the notes around are thought to be played by the clarinet, then the probability that this note is a clarinet is higher, and the probability that it is a flute is lower.

It seems that there is a bimodal distribution in results reported for instrument recognition. [Ero01] observes this as well. On the one hand some systems tend

to have about 40%–50%, and on the other hand some systems have 80%–100% accuracy. Experiments reported involving humans follow the same distribution. The explanation may be in the choices of instruments to classify. It is common to try to distinguish between the violin and viola, for example, which is a feat that people and machine alike have a difficult time doing. Distinguishing a trumpet and a violin is much easier, and corresponds to the better results. It seems best to focus on distinguishing instruments that we are able to reliably distinguish before confounding the results with violin/viola comparisons.

Instrument recognition is an active area of research. Although good results have been obtained in monophonic music, there are still many limitations and assumptions to be eliminated. Ecological solo music needs to be investigated more in depth, following the lead of Livshin, Rodet, and Vincent [LR04, VR04]. There is much work yet to be done in polyphonic instrument recognition. The polyphonic studies discussed have promising results but have just begun to scratch the surface.

There are other questions that can be asked that involve analysis of timbre, but by far the instrument recognition question is the most active right now. Most other timbre-related problems are more holistic in nature and will be considered in the next chapter.

Chapter 4

Music Understanding

We have treated the three major axes of music: rhythm, pitch, and timbre. But music is more than just the sum of these three. It evokes emotion and memory. It is instantly recognizable. It communicates.

We may not be able to make the machine feel emotion, but can we somehow recognize a sad song from a happy song? When, as musicians, we ask ourselves what makes a sad song sad, there are various concrete cues that definitely do contribute: the minor key, the tempo, the instrumentation, the ambiance (a word that can take on a meaning for a whole piece of music that is analogous to the word timbre for an instrument).

There are almost limitless questions that we can ask, and if we apply the computer to them we may in the process learn more about our own understanding of music.

There is also the aspect of using the computer directly to help us understand music better. Not to ask the computer to determine what is a sad song, but to use it as a tool to show us information about the music that is otherwise hidden in our perceptive subconscious.

We will consider a small set of papers that tackle a few questions like these. What is anticipation and surprise in music and how can we put it to use? How can we visualize sound and music in new and informative ways? What is the style of this music? Is it frantic, or the blues? Lyrical, or syncopated?

In Spectral Anticipations [Dub06], Dubnov investigates the role of surprise, randomness, and anticipation. He ties in information theory, namely entropy and information rate. There is a natural "inverted-U" structure to the interest of music. If not enough is happening (silence at the extreme), it is boring. If too much is happening, it is just noise (white noise at the extreme). Consider music as an information source, communicated to a listener. The listener forms expectations about what is coming, and the information is transmitted

in the surprise. He defines *structure* as that which the listener can predict, and *noise* as what carries no information to the listener.

Dubnov develops the equivalence between spectral flatness and information rate (IR), and then considering a vector-IR method which represents the information rate for a complex signal containing several components, he develops a generalized spectral flatness which is equivalent to vector-IR.

Take a signal and transform it with a STFT or MFCC or other such transformation, then perform basis decomposition using e.g. PCA to reduce the dimensionality of the data. The IR of each component is estimated using vector-IR. They are summed together to give a single IR value for that frame of music. This value over the time evolution of the signal gives an *anticipation profile*.

The anticipation profile seems to carry relevant information about the signal, the music structure, and emotional content.

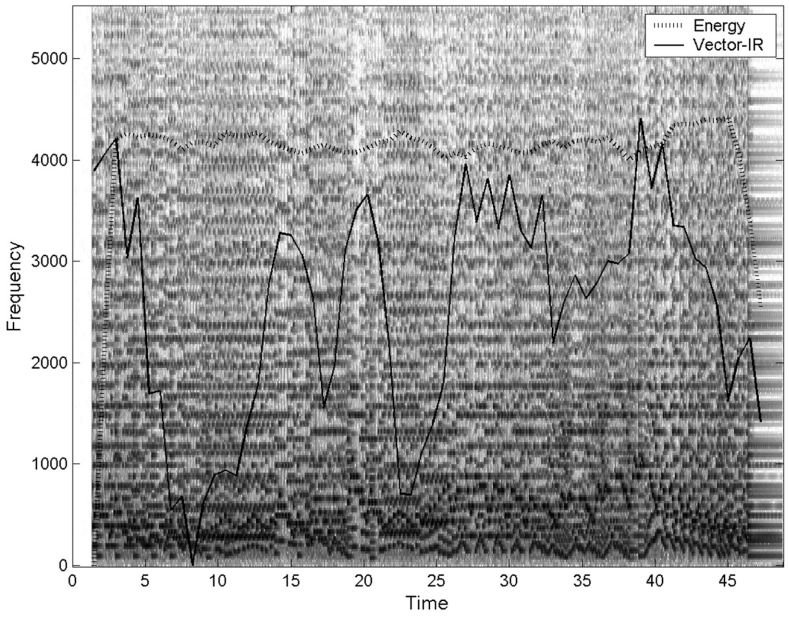


Figure 4.1: "Graph of anticipation profile (estimated by vector-IR) using 30 cepstral features, displayed over a spectrogram of a musical excerpt (Bach's Prelude in G Major *from Book I of the* Well-Tempered Clavier*). The acoustic signal was created by computer rendering of a MIDI file." [Dub06]*

Cooper et al. [CFPT06] review a number of music visualizations. The *similarity matrix* we have discussed already. It reveals temporal structure of the music. The *beat spectrum* is a measure of self-similarity as a function of lag. The beat

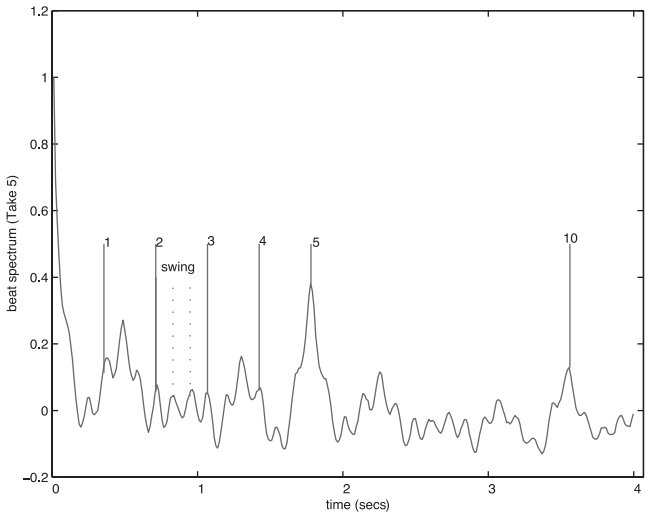


Figure 4.2: "Beat spectrum of the jazz composition Take Five*." [CFPT06]*

spectrum is a telling visualization about the temporal structure of the frame of music. The *beat spectrogram* is obtained with the addition of the time axis, as the spectrogram from the spectrum. The beat spectrogram gives even more visual cues to the temporal structure, and, for example, makes it easy to see changes in time signature. The *beat histogram* is similar to the beat spectrum, but calculated in a different way (and usually over the entire song).

Timbregrams use vertical bars of color to represent similarity in timbre. They are commonly made using PCA from feature vectors. If the song has an ABA structure, the ABA structure will be visible also in the timbregram.

Songs as well as samples of instruments, etc. may be organized into timbre spaces (either 2-dimensional or 3-dimensional) and plotted. Music of similar style or instrumentation will be clustered together. A related visualization clusters music using self-organizing (Kohonen) maps, and clusters are depicted as islands on a map. These automatically clustered Islands of Music provide an insightful exploratory landscape.

These and other visualizations allow us to see patterns in music due to our excellent capacity for pattern recognition. These visualizations can be fun and informative, and they can also guide us to insight about features of music that we can then exploit in a programmatic fashion.

Deshpande et al. [DSN01] take this one step further, and use image processing techniques to classify music into broad categories (rock, classical, jazz) using visualizations. The spectrogram and MFCC of music in these categories has certain tendencies. Rock has strong vertical lines corresponding to the pow-

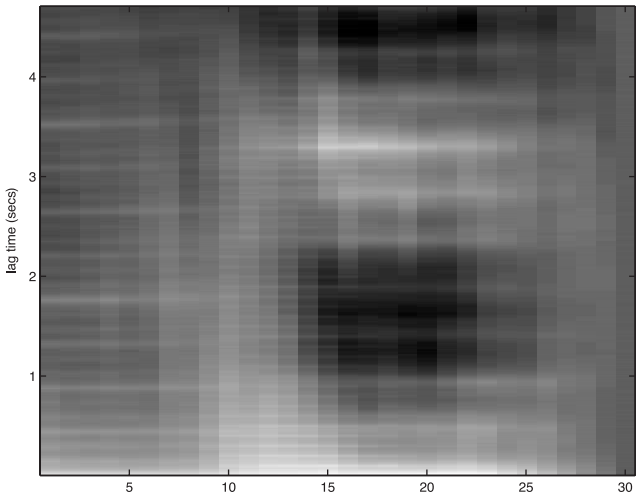


Figure 4.3: "Beat spectrogram of Pink Floyd's Money *showing the transition from 4/4 time to 7/4 time around 10 seconds (on the x-axis)." [CFPT06]*

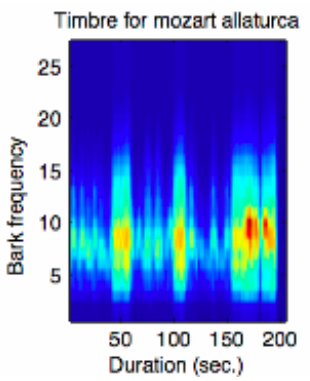


Figure 4.4: Timbregram for Mozart's Alla Turca*, from [KJ08]*

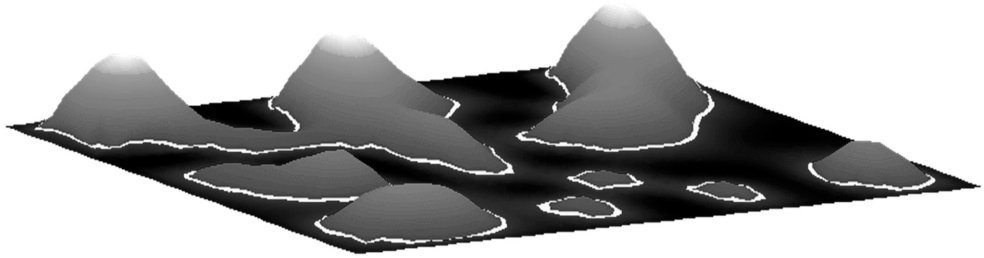


Figure 4.5: Islands of Music from [CFPT06]

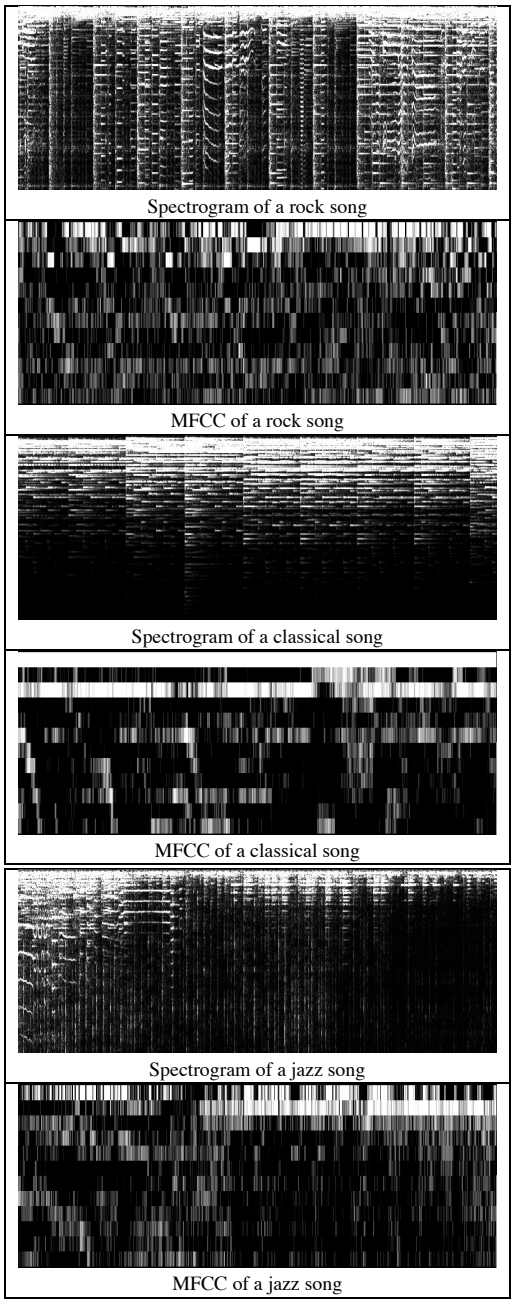


Figure 4.6: "The above figures show images of spectrogram and MFCC data for rock, classical, and jazz music." [DSN01]

erful drum beat. Classical music tends to be smooth because most classical instruments have relatively pure tones. Jazz tends to look much more random, and some characteristic instruments (e.g. saxophones) have distinctive patterns in the spectrogram.

The spectrogram and MFCC are decomposed into a vector in $\mathbb{R}^{k^d}$, by convolving with k filters recursively d times. This is the texture-of-textures approach. The feature vectors were then classified using k-NN, GMM, and SVM and compared. They found k-NN worked best, but believe it may be due to not finding the optimal parameters or not having a large enough set of data.

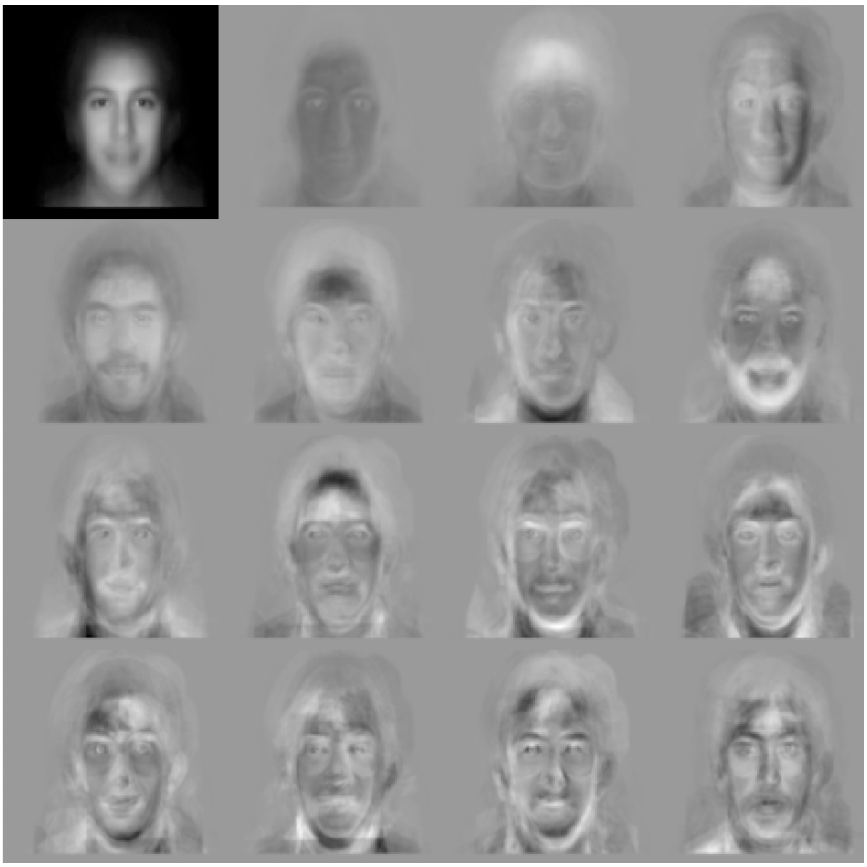


Figure 4.7: Eigenfaces from [ZC06]

A similar approach, though not one discussed by Deshpande et al. would be to use an *eigenface* technique instead of texture-of-textures. At a high level they are very similar. An eigenface is to an image as an eigenvector is to a matrix. A set of eigenfaces is derived using PCA, each representing some orthogonal aspect of the set of training data. A face is recognized by a combination of

eigenfaces that gives the highest correlation. This technique could be used to capture the principal components of visualizations like the spectrogram of MFCC. It has also been adapted to sound in speech recognition. Eigenfaces are discussed at length in [Abd88, PMS94, TP91].

Recht and Whitman [RW03] "abstract away" auditory events (using PCA) in the incoming signal to reveal the *eigensound*—a generalized spectral template. This generalized audio is then resynthesized producing interesting and pleasing textures. For a time Whitman ran an "eigenradio" that analyzed multiple radio stations at once and resynthesized generalized audio in real time. "One song on Eigenradio is worth at least twenty songs on old radio."

Dannenberg et al. [DTW97] review their work on musical style recognition using machine learning. They look at the problem of recognizing the style in which a piece of music is played (e.g. on the keyboard). Is it played lyrically or energetically, or with syncopation? "Although it may appear obvious how one might detect these styles, good musical performance is always filled with contrast. For example, energetic performances contain silence, slow lyrical passages may have rapid runs of grace notes, and syncopated passages may have a variety of confusing patterns." Their work is based on symbolic (MIDI) input, but uses some of the same machine learning techniques discussed earlier. They used 13 features extracted from the MIDI data (statistics of the MIDI note, duration, duration to IOI ratio ("duty factor"), pitch, volume, etc.), then classified using naïve Bayes, a linear classifier that looks for a separating hyperplane in a Gaussian mixture, and a nerual network. They report excellent results, and include a discussion on why machine learning "works so well when hand-coded approaches have consistently failed."

Dannenberg [Dan91] discusses three music understanding applications as of 1991. They take symbolic (MIDI) input, but the techniques are related to current techniques (after onset detection or transcription). The first is score following, which is done using a matching algorithm modified from the longest common subsequence algorithm. It finds the best correspondence between performed notes and the score. The second follows improvisation on the 12-bar blues. It recognizes the beat and looks for likely chord changes, and jumps in at the beginning of the cycle. Finally he describes a rhythm tracker using beam search. Beam search keeps track of alternative representations and continuously updates its estimates of the best tracker. It prunes off unlikely alternatives to avoid exponential growth. It is instructive to compare these approaches with modern approaches that are contending with more difficult data, not the least of which is that it begins as audio data instead of symbolic.

Scheirer and Slaney [SS97] developed a robust speech/music discriminator. It uses 13 features that are particularly relevant to discriminating speech from music, including 4 Hz modulation energy, percentage of low-energy frames, spectral skewness, spectral centroid, spectral flux, ZCR, a cepstral feature, and a pulse metric which determins the amount of "rhythmicness" in a 5-second window. For classification they use a GMM. They report excellent and robust

results.

Conclusion

We have seen some very interesting audio analysis. There has been exciting progress in this field, but the field is still young and there is still much room for improvement. There is a clear trend from direct procedural algorithms to probabalistic and machine learning algorithms. It is apparent that in almost every aspect the analysis of music is a holistic endeavor, not easily captured with simple rules. The abstractions we make when we listen to music are complex, with multiple causes. One day we may understand our own cognitive processes enough to instruct the computer to mimic them. Until then, the best approach seems to be algorithms which can make their own abstractions, which at least partially coincide with ours. In the process we have been and will be surprised at what we learn about ourselves.

Appendix A

Music Primer

The essence of musc is sound. The essence of sound is time. Appendix B will address sound itself in more depth. For this primer, it will suffice to notice the different characteristics of sounds produced by instruments.

Some are bright, some are mellow. These and other descriptions of the texture of sound is called *timbre* (sounds like "amber"). Some sounds are loud, and some are soft. This distinction we call *dynamics*. Some sounds are high and some are low. This is called *pitch*. A coherent sound such as that coming from an instrument is called a *note*.

Rhythm is the pattern of sounds in time. Rhythms are usually repetitive. Almost all music has a *beat*, a recurring pulse that drives the music along. The beat is like a rhythmic foundation to the music. It gives structure and direction. The beat has a period, and therefore a frequency, on the order of 50–200 beats per minute (bpm). We call this *tempo.* 120 bpm is a typical tempo.

But the beat isn't the last word. Rhythmically interesting music plays around the beat. Some notes are held longer, some come between beats, some are "swung" or *syncopated*. The variation from the beat adds surprise and interest. But the beat is always there, even if it is not explicitly expressed.

When notes of different pitches sound together, they can sound good or they can sound bad. When they sound good, it is *harmonic*. When they sound bad, it is *dissonant* or *inharmonic* (technically speaking inharmonicity is subtly different from dissonance).

A *melody* is a pleasing sequene of notes. Melodies often repeat—sometimes with slight variations—throughout the piece. The other notes happening around the melody—often lower in pitch but not always—make up the *accompaniment*.

The *composer* writes music, and the *performer* reads and performs it. Musical *notation* has been devised over the centuries as a common language of communication between musicians.

The central element of musical notation is the note. A note has a pitch and a *duration*. Durations are rational, i.e. they are all integer ratios from a basis. Usually, a quarter note gets one beat and there are four beats in a measure. The *time signature* in this case is written as 4/4, meaning 4 beats to the measure (numerator) and the quarter note gets one beat (denominator). Another common time signature is 3/4, where there are 3 quarter notes to a measure. Measures are delineated by bar lines. A half note is twice as long as a quarter note, i.e. 2 beats. A whole note is 4 beats. In 4/4 time there are 2 eighth notes to a beat, 4 sixteenth notes to a beat, and so forth.

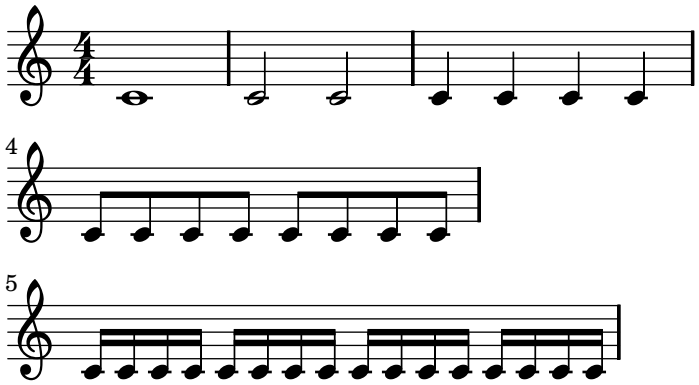


Figure A.1: A whole note, two half notes, four quarter notes, eight eighth notes, sixteen sixteenth notes. Each make up one measure—four beats.

The vertical position of the note denotes its pitch. Pitches are named A, B, C, D, E, F, and G. There are seven notes in a *scale*, and then we begin again. A span of eight notes in the scale is called an *octave*. Sometimes octaves are given numbers, so we may talk about A4 and A5. Octaves double or halve the frequency. A4 is 440 Hz, so A5 is 880 Hz and A3 is 220 Hz. To our ears, the same note in different octaves sounds like the same pitch, but higher or lower.

The scale is not divided into 7 equal pieces, but 12 *semitones*. In musical notation, notes are arranged on a *staff*. A note can be either on a line, or on a space. Each *major scale* is a shifted version of the C major scale, which is C, D, E, F, G, A, B, and back to C (the white keys on a keyboard). From a line to the adjacent space is one step on the C major scale. The *accidentals* are the semitones between some of these notes (the black keys on a keyboard), e.g. C sharp (which is also D flat), D sharp, F sharp, etc. There is no E sharp—the next semitone up from E is F (the same goes for B). So the *chromatic scale* from C4 to C5 is C, C sharp, D, D sharp, E, F, F sharp, G, G sharp, A, A sharp, B, and back to C. So each scale begins with a *whole step*, another whole step, then a half step, etc.

A *rest* indicates silence. There are quarter rests, whole rests, etc.

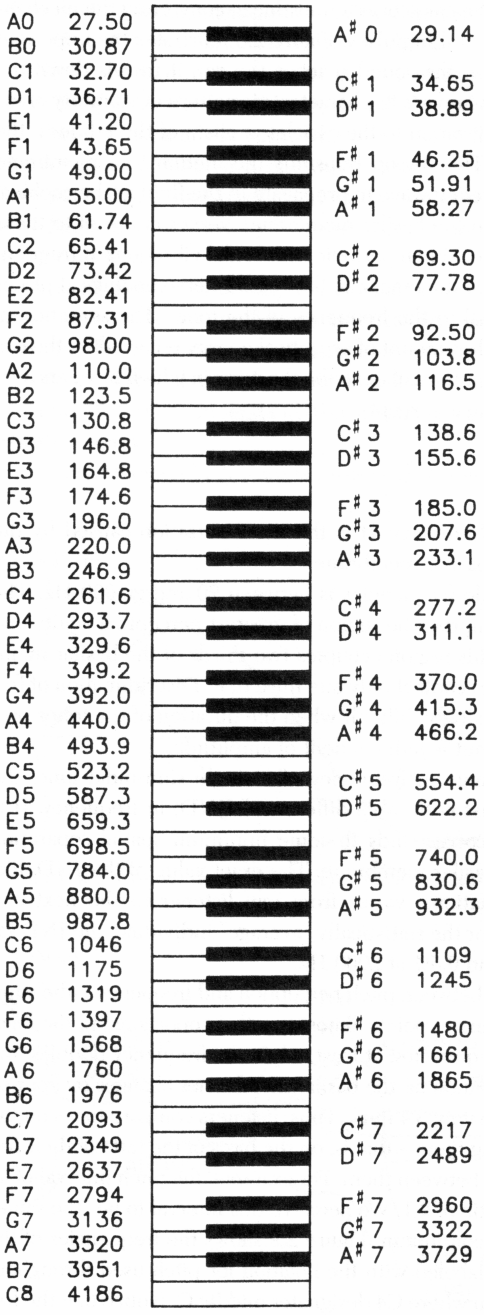


Figure A.2: "Frequencies and pitches of the equal-tempered piano keyboard" [DJ97, pg 34].

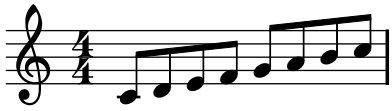


Figure A.3: C major scale from C4 to C5.

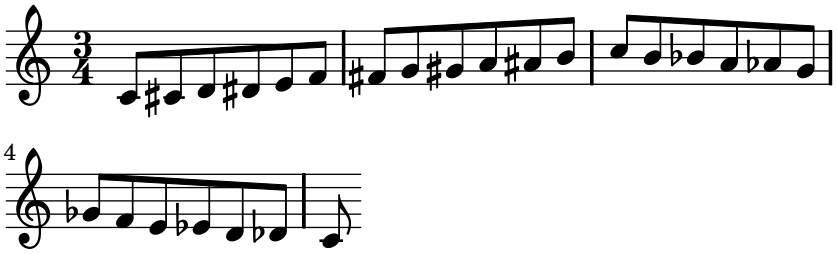


Figure A.4: The chromatic scale from C4 to C5 and back to C4. Sharps are used going up, and flats are used going down.

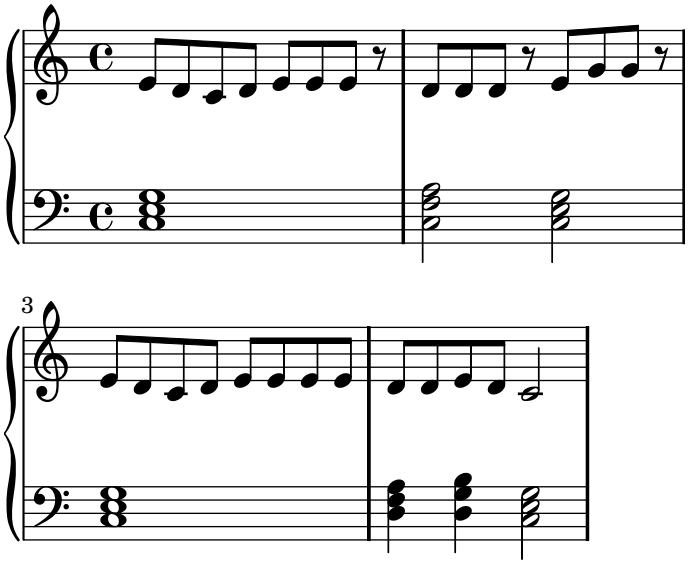


Figure A.5: Mary had a little lamb

A major scale (the *diatonic* scale) is a nice-sounding progression through an octave. There are also minor scales, which also sound nice but sad. *Chords* are simultaneous groupings of notes. Some chords sound stable and harmonic, like the *tonic*, which is the major chord starting on the *root* of the key. Other chords lead the listener in anticipation of *resolution* to e.g. the tonic chord. For our purposes, it is enough to know that in a given key there are various chords which go hand in hand with melody. Chords tend to progress in certain patterns that sound nice, and *chord progressions* are used by composers and improvising musicians to aid them in deciding which notes to play (they play the notes that are members of the current chord).

Musical works, especially songs, are generally organized into a few repeating parts. A common song structure is AABA, meaning the A section repeats twice, the B section once, and then the A section again. ABA is another common structure. Different sections are often called *verse*, *chorus*, *bridge*, etc.

MIDI stands for Musical Instrument Digital Interface. It is a protocol for communicating with electronic musical instruments, like keyboards and synthesizers. It is a simple protocol that sends events when they happen. Each event contains information such as "play a C4 with velocity 52" or "stop playing C4," or information about various state parameters called *controllers*. MIDI events can be stored in a file with associated timestamps, and the resulting MIDI files can be played back over a MIDI interface (or of course by software that interprets MIDI data). The important distinction about MIDI is that it does not transmit an audio signal, but instead symbolic instructions. This symbolic information is less abstract than a musical score, but much closer to musical notation than it is to sound.

There is much more to music and its notation, but these basics should be sufficient to understand this paper.

Appendix B

Digital Signal Processing Primer

This will be a very brief primer on DSP, completely lacking in theoretical rigor. For a more thorough treatment of DSP, see [OS89].

Sound is the result of vibration. Something vibrates (a tuning fork, for example), and this causes the air around it to vibrate. The compression wave thus generated propagates until it reaches our ears, where it is perceived as sound. So we speak of sound as a wave, and although it propogates as a compression wave we think of it as a traditional mathematical wave.

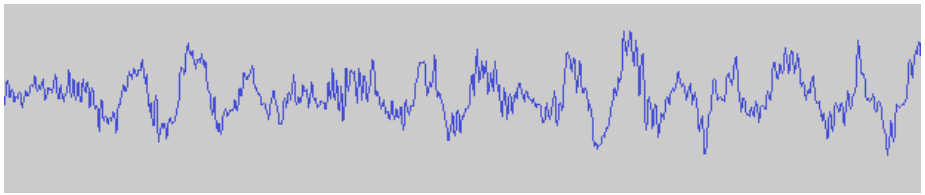


Figure B.1: A sound wave

In order to process sound with computers, we need to convert that analog sound wave into ones and zeroes. We do this by *sampling* the sound at a fixed rate, and recording the *amplitude* of the sound at each moment. The result is a list of numbers, which we call a *digital signal.*

A sine wave with period T has frequency $f = \frac{1}{T}$ and is a pure tone at that frequency. Period is measured in seconds, and frequency in Hertz (Hz).

The rate at which we take samples is called the *sample rate*, sometimes denoted f_s. A discrete-time sampling of a signal faithfully represents that signal if the sample rate is at least twice the highest frequency in the signal. This minimal

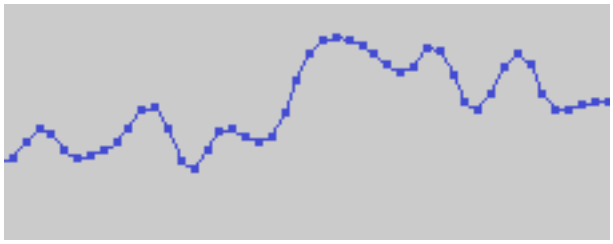


Figure B.2: A sampled audio signal. The dots are samples, and the lines are just interpolated.

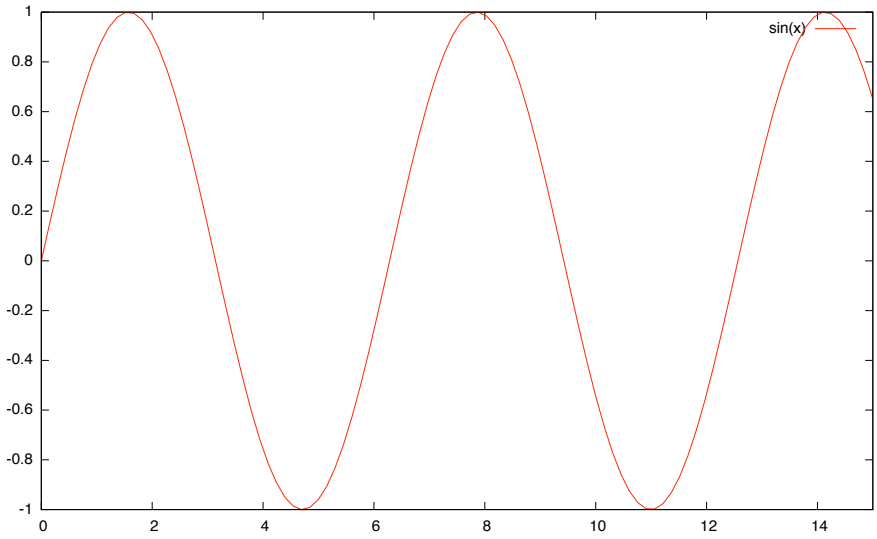


Figure B.3: Sine wave

sample rate is named the *Nyquist frequency*. Humans can hear sound from roughly 20–20000 Hz, so in order to reproduce the sounds we can hear, we would have to sample at at least 40000 Hz. Indeed, the sample rate of CD-quality audio is 44100 Hz.

In addition to sampling in time, another discretization occurs because we are limited in the precision with which we can represent amplitude. CD-quality audio uses 16 bits of resolution.

The *discrete Fourier transform* (DFT) is a very useful mathematical tool which allows us to analyze the frequency content of a signal. It transforms the signal from the *time domain* into the *frequency domain*. A pure sine wave in the frequency domain is (ideally speaking) a spike at the sine wave's frequency, and 0 everywhere else[1].

Ecological signals—those signals we find in the "real world"—are generally much more interesting in the frequency domain. Any sound can be mathematically decomposed into component sinusoids of different frequencies and amplitudes. In the frequency domain, we see these these component sinusoids at their frequencies. We call the 2-dimensional plot of frequency and amplitude the *spectrum*.

It is common to apply the Fourier transform to a short *frame* of the signal at a time (on the order of a few to a few tens of milliseconds). This is called the short-time Fourier transform (STFT). A fast algorithm for computing the DFT is the fast Fourier transform (FFT), and when we talk about the Fourier transform in computer music we generally are thinking of the FFT unless otherwise noted[2].

A series of STFT in time gives a 3-dimensional space: frequency, amplitude, and time. A popular music visualization is the *spectrogram*, which puts time on the x axis, frequency on the y axis, and represents amplitude with intensity or color.

When working with musical instruments, it is common to talk about *partials*, which are the component sinusoids that add up to the (ideal) instrument waveform. They are often *harmonics*, i.e. integer multiples of the fundamental frequency.

Our hearing mechanism is nonlinear. We hear frequencies as being the same distance apart, which are actually closer to logarithmic (the octave 220 Hz to 440 Hz is perceptually the same distance as the octave 440 Hz to 880 Hz, etc.). Likewise, we perceive loudness differently based on the pitch of the sound.

[1] Actually, this is only half true. Specifically, only over half of the frequency range of the Fourier transform (from 0 to the Nysquist frequency). The distinction is not important in our discussions, and it is common practice in computer music to look only at the frequencies from 0 to the Nyquist frequency.

[2] We are also ignoring the *phase* and dealing solely with the *amplitude* or *power spectrum* (the Fourier transform gives both). Although phase can be important in computer music, we will not need it in this paper.

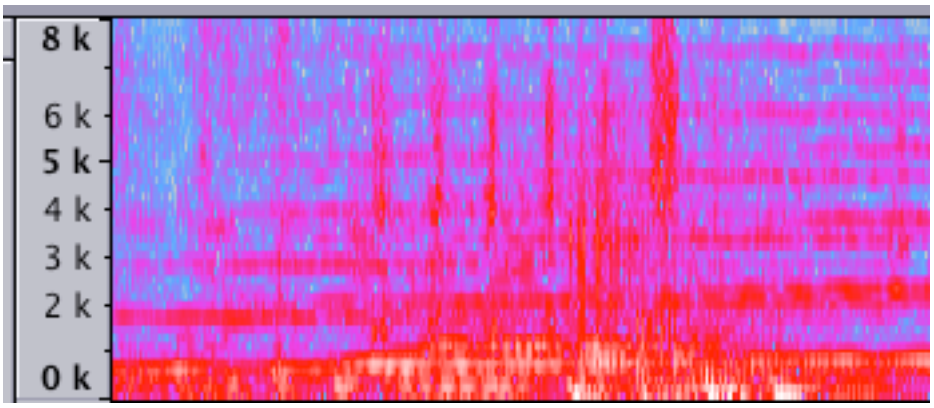


Figure B.4: A spectrogram. Blue is low amplitude, red is middle amplitude, and white is high amplitude.

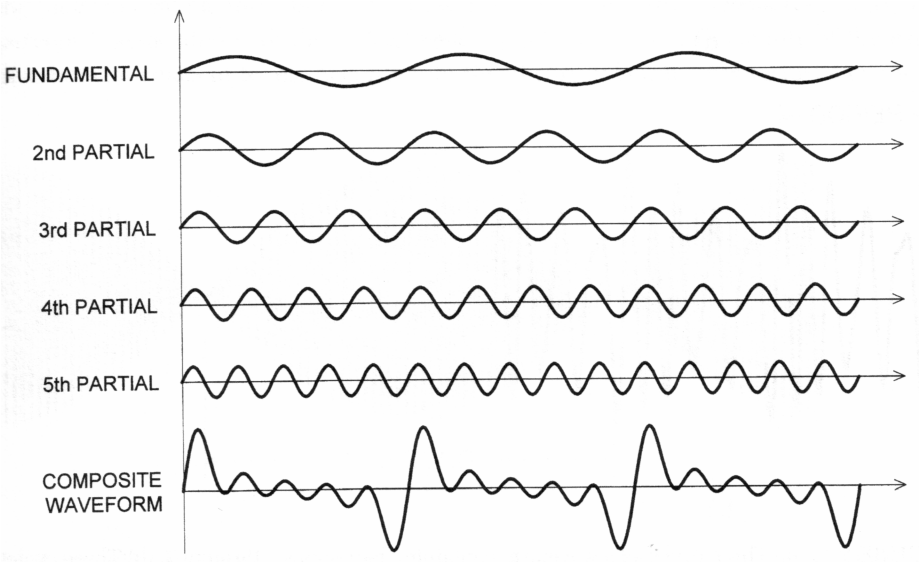


Figure B.5: "Complex periodic waveform and the five harmonic partials of which it is comprised" [DJ97, pg 35].

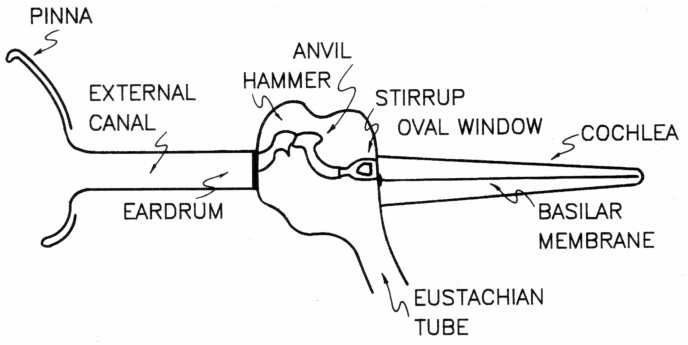


Figure B.6: "Schematic of the hearing apparatus of the ear" [DJ97, pg 32].

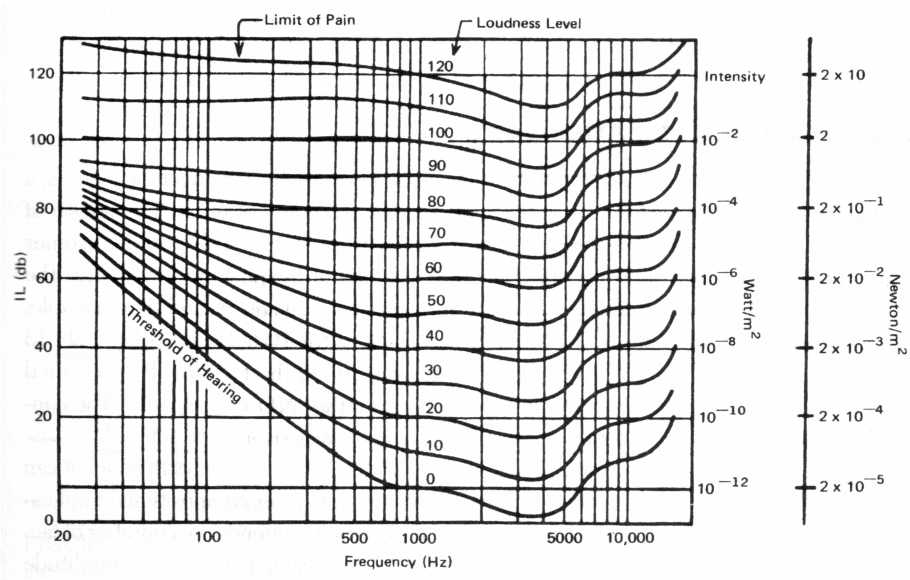


Figure B.7: "Fletcher-Munson diagram of equal loudness for tones of different frequencies. (Reprinted from Introduction to the Physics and Psychophysics of Music, by Juan C. Roedere with the permission of Springer-Verlag, Heidleberg, Germany)" [DJ97, pg 43].

Appendix C

Machine Learning Primer

Computers are dumb. They do just what we tell them to do. Humans are smart and adaptive. We are able to learn from experience and from teaching, gaining knowledge and understanding. We would like to help the computer to also learn, because that adaptability is useful and robust, and then they may be able to learn to do things that we can't figure out how to program directly.

Computers can't learn nearly as well as humans, but there are algorithms that can learn if applied correctly. We will discuss a few concepts and algorithms here and in the paper. For a more thorough treatment, see [Mit97].

Mitchell gives this definition of machine learning: "A computer program is said to *learn* from experience E with repsect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [Mit97, pg. 2]. So to apply machine learning to a problem we need to well define the class of tasks, the performance measure, and the experience.

It is important to understand the role of *generalization* in machine learning. If the problem is classification, then given a set of training examples it is easy to perfectly classify them by simply remembering. This doesn't help us to classify new examples that we haven't seen. We need not to memorize, but to generalize. Proper validation of machine learning algorithms using test data that demonstrates the generality of the algorithm is important. One of the more important aspects is to be sure that you have a large enough and diverse enough data set to sufficiently train and test the algorithm.

Machine learning classification algorithms can be either *supervised* or *unsupervised*. A supervised algorithm is told the correct labels during the training phase, and it adjusts its state so that it performs better the next time it sees an example like that. An unsupervised algorithm is given no external feedback about the "correct" answer, and instead the performance measure is internal.

This is not very useful for learning human labels, but it is able to e.g. cluster examples by similarity, even though the similarity measure is not known a priori.

Machine learning is diverse and multidisciplinary field. It brings together artificial intelligence, statistics, search, and more. There are many different algorithms that can learn many different tasks from many different kinds of experience. In this paper we are primarily concerned with learning to classify.

In the classification problem, the experience is a set of examples and their correct classification (for supervised learning). The performance measure is whether or not it correctly classified the example, and the task is choosing a class to which the example belongs. Because of the nature of the examples (audio signals), every example is very different from the other in a direct comparison. For this reason, *features* are extracted from the audio signal which better represent aspects of the signal that are similar. These features include the spectrum (which is itself also a good basis for more features), power, and so forth.

Classification algorithms then take these features and choose a classification. Often, the output of the classification algorithm itself is a probability distribution, i.e. it is probable that it belongs to class X, but there is also a non-zero probability that it may belong to other classes.

Often multiple algorithms are applied in a single study, and their results compared or even combined to give a more confident answer.

Algorithms vary in how they approach the problem. Each has its own advantages and disadvantages. Neural networks provide fast classification and are robust to noise, but they take a long time to train and the trained network is a black box. k-Nearest Neighbor is slower to classify, fast to train, and very easy to implement. Some algorithms are more prone to overfitting the data (not generalizing enough), and others are prone to be too general and not provide specific classifications with high enough accuracy. Likewise some features are more useful or robust than others, and some combinations of features with algorithms are more useful than others.

Applying machine learning is an exercise in learning itself—we try to pick features and algorithms that will perform well, then we try it out and see where things fall short, then we try tweaking a parameter, feature, or algorithm, and this iteration process continues. In the process we may learn a little about how we learn and the nature of the task.

Bibliography

A

[Abd88] Hervé Abdi. A generalized approach for connectionist auto-associative memories: Interpretation, implications and illustration for face processing. *Artificial intelligence and cognitive sciences*, pages 149–165, 1988.

[AP01] Samer Abdallah and Mark Plumbley. Sparse Coding of Music Signals. Submitted to Neural Computation, March 2001.

B

[BDA+05] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B. Sandler. A Tutorial on Onset Detection in Music Signals. *Speech and Audio Processing, IEEE Transactions on*, 13(5 Part 2):1035–1047, 2005.

[BMS00] Juan Pablo Bello, Giuliano Monti, and Mark B. Sandler. Techniques for Automatic Music Transcription. In *ISMIR*, 2000.

[BP89] Judith C. Brown and Miller S. Puckette. Calculation of a "narrowed" autocorrelation function. *The Journal of the Acoustical Society of America*, 85:1595–1601, 1989.

[BW05] Mark A. Bartsch and Gregory H. Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on Multimedia*, 7(1):96–104, February 2005.

[BZ91] Judith C. Brown and Bin Zhang. Musical frequency tracking using the methods of conventional and "narrowed" autocorrelation. *The Journal of the Acoustical Society of America*, 89(5):2346–2354, May 1991.

C

[CDGW01] Emilios Cambouropoulos, Simon Dixon, Werner Goebl, and Gerhard Widmer. Computational models of tempo: Comparison of human and computer beat-tracking. *Proceedings of VII International Symposium on Systematic and Comparative Musicology and III International Conference on Cognitive Musicology*, pages 18–26, 2001.

[CFPT06] Matthew Cooper, Jonathan Foote, Elias Pampalk, and George Tzanetakis. Visualization in audio-based music information retrieval. *Computer Music Journal*, 30(2):42–62, 2006.

[CKB04] Ali Taylan Cemgil, Bert Kappen, and David Barber. A Generative Model for Music Transcription. *IEEE Transactions on Speech and Audio Processing*, 2004.

[CKDH01] Ali Taylan Cemgil, Bert Kappen, Peter Desain, and Henkjan Honing. On tempo tracking: Tempogram representation and Kalman filtering. *Journal of New Music Research*, 28(4):259–273, 2001.

D

[Dan91] Roger B. Dannenberg. Recent work in real-time music understanding by computer. In *Proceedings of the International Symposium on Music, Language, Speech and Brain*, pages 194–202, 1991.

[Dan05] Roger B. Dannenberg. Toward automated holistic beat tracking, music analysis, and understanding. In *ISMIR 2005 6th International Conference on Music Information Retrieval Proceedings*, pages 366–373, London, 2005. Queen Mary, University of London.

[DH99] Peter Desain and Henkjan Honing. Computational Models of Beat Induction: The Rule-Based Approach. *Journal of New Music Research*, 28(1):29–42, 1999.

[Dix01] Simon Dixon. Automatic Extraction of Tempo and Beat From Expressive Performances. *Journal of New Music Research*, 30(1):39–58, 2001.

[DJ97] Charles Dodge and Thomas A. Jerse. *Computer Music: synthesis, composition, and performance*. Schirmer, second edition, 1997.

[DSN01] Hrishikesh Deshpande, Rohit Singh, and Unjung Nam. Classification of music signals in the visual domain. In *Proceedings of the COST-G6 Conference on Digital Audio Effects*, 2001.

[DTW97] Roger B. Dannenberg, Belinda Thom, and David Watson. A machine learning approach to musical style recognition. In *International Computer Music Conference*, 1997.

[Dub06] Shlomo Dubnov. Spectral Anticipations. *Computer Music Journal*, 30(2):63–83, 2006.

E

[Ero01] Antti Eronen. Automatic musical instrument recognition. Master's thesis, Tampere University of Technology, 2001.

G

[GD05] Fabien Gouyon and Simon Dixon. A review of automatic rhythm description systems. *Computer Music Journal*, 29(1):34–54, Spring 2005.

[GH96] Zoubin Ghahramani and Geoffrey E. Hinton. Parameter Estimation for Linear Dynamical Systems. Technical Report CRG-TR-96-2, University of Toronto, February 1996.

[Got01] Masataka Goto. An Audio-based Real-time Beat Tracking System for Music With or Without Drum-sounds. *Journal of New Music Research*, 30(2):159–171, 2001.

H

[HABS00] Perfecto Herrera, Xavier Amatriain, Eloi Batlle, and Xavier Serra. Towards instrument segmentation for music content description: a critical review of instrument classification techniques. *International Symposium on Music Information Retrieval*, 2000.

[HBPD03] Perfecto Herrera-Boyer, Geoffroy Peeters, and Shlomo Dubnov. Automatic Classification of Musical Instrument Sounds. *Journal of New Music Research*, 32(1):3–21, 2003.

[HDG03] Perfecto Herrera, Amaury Dehamel, and Fabien Gouyon. Automatic labeling of unpitched percussion sounds. *114th AES Convention*, 2003.

[Hon01] Henkjan Honing. From Time to Time: The Representation of Timing and Tempo. *Computer Music Journal*, 25(3):50–61, 2001.

K

[Kal60] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[KGK+05] Tetsuro Kitahara, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Instrument Identification in Polyphonic Music: Feature Weighting with Mixed Sounds, Pitch-Dependent Timbre Modeling, and Use of Musical Context. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[Kit93] Hiroaki Kitano. Challenges of massive parallelism. *Proc. of IJCAI-93, the Thirteenth International Joint Conference on Artificial Intelligence, Chambery, France, August*, pages 813–834, 1993.

[KJ08] Ole Kuhl and Kristoffer Jensen. Retrieving and recreating musical form. In R. Kronland-Martinet, S. Ystad, and K. Jensen, editors, *Computer Music Modeling and Retrieval 2007*, Lecture Notes in Computer Science, pages 263–275, 2008.

[KP00] J. Klingseisen and Mark D. Plumbley. Towards musical instrument separation using multiple-cause neural networks. In *Proc. ICA*, pages 447–452, 2000.

L

[LR04] Arie A. Livshin and Xavier Rodet. Musical instrument identification in continuous recordings. In *Proceedings of the 7th International Conference on Digital Audio Effects (DAFX-4)*, October 2004.

M

[May79] Peter S. Maybeck. *Stochastic models, estimation, and control, Volume 1*, volume 141 of *Mathematics in Science and Engineering*, chapter 1, pages 1–15. Academic Press, Inc., 1979.

[MB96] Paul Masri and Andrew Bateman. Improved modelling of attack transients in music analysis-resynthesis. In *Proceedings of the 1996 International Computer Music Conference*, pages 100–103. International Computer Music Association, 1996.

[Mit97] Tom M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, 1997.

[MK98] Keith D. Martin and Youngmoo E. Kim. 2pMU9. Musical instrument identification: A pattern-recognition approach. In *Proceedings of the 136th meeting of ASA*, October 1998.

O

[OS89] A.V. Oppenheim and R.W. Schafer. *Discrete-time signal processing*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1989.

P

[PAB+02] Mark D. Plumbley, Samer A. Abdallah, Juan Pablo Bello, Mike E. Davies, Giuliano Monti, and Mark B. Sandler. Automatic music transcription and audio source separation. *Cybernetics and Systems*, 33(6):603–627, September 2002.

[PMS94] Alex Pentland, Baback Moghaddam, and Thad Starner. View-based and modular eigenspaces for face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, WA, June 1994.

R

[Rap05] Christopher Raphael. A graphical model for recognizing sung melodies. In *ISMIR*, pages 658–663, 2005.

[Rib04] Maria Isabel Ribeiro. Gaussian Probability Density Functions: Properties and Error Characterization. Technical report, Institute for Systems and Robotics, February 2004.

[RW03] Ben Recht and Brian Whitman. Musically expressive sound textures from generalized audio. *Proceedings of the 6th International Conference on Digital Audio Effects*, pages 8–11, 2003.

S

[Sau95] Eric Saund. A multiple cause mixture model for unsupervised learning. *Neural Computation*, 7(1):51–71, 1995.

[Sch98] Eric D. Scheirer. Tempo and beat analysis of acoustic musical signals. *The Journal of the Acoustical Society of America*, 103:588, 1998.

[She64] R.N. Shepard. Circularity in Judgments of Relative Pitch. *The Journal of the Acoustical Society of America*, 36:2346, 1964.

[Smi02] Lindsay I. Smith. A tutorial on principal component analysis. Student Tutorial, University of Otago, February 2002.

[SMS05] William A. Sethares, Robini D. Morris, and James C. Sethares. Beat Tracking of Musical Performances Using Low-Level Audio Features. *IEEE Transactions on Speech and Audio Processing*, 13(2):1, 2005.

[SOŠ03] Václav Syrovỳ, Zdeněk Otčenášek, and Jan Štěpánek. Subjective evaluation of organ pipe timbre in the standard listener positions. In *Proceedings of the Stockholm Music Acoustic Conference 2003*, pages 333–336, 2003.

[SS97] Eric D. Scheirer and Malcolm Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *ICASSP*, pages 1331–1334, 1997.

[SW04] A.M. Shenoy and R.Y. Wang. Key determination of acoustic musical signals. *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*, 3, 2004.

[SW05] Arun Shenoy and Ye Wang. Key, Chord, and Rhythm Tracking of Popular Music Recordings. *Computer Music Journal*, 29(3):75–86, 2005.

T

[Tem04] David Temperley. An evaluation system for metrical models. *Computer Music Journal*, 28(3):28–44, Fall 2004.

[TP91] Matthew A. Turk and Alex P. Pentland. Face recognition using eigenfaces. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991.

V

[VR04] Emmanuel Vincent and Xavier Rodet. Instrument identification in solo and ensemble music using independent subspace analysis. In *Proceedings of ISMIR*, pages 576–581, 2004.

W

[WB06] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill, July 2006.

Z

[ZC06] W. Zhao and R. Chellappa. *Face Processing: Advanced Modeling and Methods*. Academic Press, 2006.