PAQ on MIDI

Anonymous Author(s)
Affiliation
Address
email

Abstract

Predicting and compressing data can be seen as two sides of the same coin. Compression mechanisms can be modified to achieve prediction. If one can predict well then the prediction can be used to compress data. Both concepts are tightly intertwined with intelligence. Approaching these problems from an algorithmic perspective reduces an aspect of intelligence to a measurable feature. Human musical perception is strongly influenced by our ability to predict forthcoming musical sequences. This paper investigates the use of the PAQ8 compression algorithm to make predictions about MIDI music, PAQ8 is an algorithm that is known to perform well at text prediction. We make use of the text prediction codebase and expand and manipulate it to perform musical predictions.

1 Introduction

Prediction is closely related to intelligence. The human mind is predictive in tasks that involve anticipation, preparation, prospection, or expectation [1]. In fact, it has been suggested that prediction in the brain might be a major factor in neural computation; when an inaccurate prediction is made revisions to expectations ought to occur (a kind of learning). It is postulated that the human mind uses prediction to better understand and interpret spoken language [2] [3]. Prediction has also been found to be highly relevant in music perception. Human musical perception is greatly dependent on our ability to anticipate forthcoming musical sequences [4]. Anticipation and prediction of music may event contribute to musical appreciation [5].

Prediction can be modeled with compression techniques [6]. Generally speaking, given a compression algorithm and a sequence of data, a good prediction would be an addition that minimally increases the size of the compressed file. By minimally increasing the size of the compressed file the added data conforms with some property of the original sequence. Predictions by compression exploit uniform properties of data, expecting sequential data to preserve those consistencies.

Through a compression mechanism we investigate the successfulness of music prediction. Musical prediction has strong ties to musical perception and therefore intelligence as well. The Hutter prize, a compression competition, is motivated partly by the idea that compression is closely related to intelligence [7]. Explorations in compression reduce the question of intelligence to hard file sizes. Here we are interested in the use of a PAQ compression algorithm to predict Musical Instrument Digital Interface (MIDI) music. The MIDI format is a protocol for specifying the production of computer-generated music. The PAQ model is trained on a set of MIDI files. By feeding the model an introductory sequence of notes we aim to be able to generate a MIDI music file that resembles the training music and the introductory sequence.

2 PAQ

The Hutter prize [7] is rewarded for an algorithm that can improve the state of the art compression ratio of Wikipedia. The motivation for such a competition is the idea that intelligence and compression are two closely related concepts. Since the founding of the Hutter prize in 2006 the PAQ algorithm has been a recurrent front runner for state of the art compression techniques. PAQ is a lossless compression technique that is widely applicable to a variety of domains. Lossless compression algorithms generally consist of two stages. Firstly, a probability distribution that predicts a sequential character given the preceding context is developed. Secondly, the probability distribution is encoded using a scheme such as Arithmetic Coding or Huffman Coding. PAQ8 is a variant of the PAQ algorithm developed by Matt Mahoney [8]. In particular, PAQ8 decomposes the first stage into three modules; the PAQ8 architecture is shown in Figure 1.

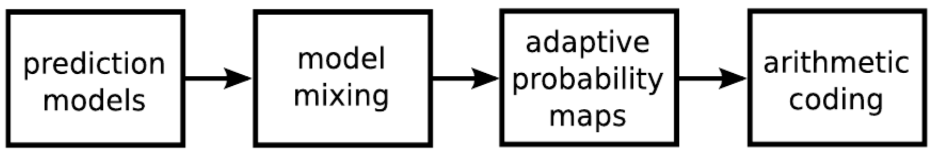


Figure 1: The architecture of PAQ8 [9]. PAQ8 is comprised of four modules; the first three participate in the creation of a probability distribution that describes the data, the last encodes the probability distribution.

The PAQ8 algorithm is an aggregate of many of simple prediction models. This variant of PAQ is an online learning algorithm, which implies that it cyclically and constantly updates the parameters used in its prediction. In Figure 2, parameters can be seen as the weights on links in the network. The predictions of each simple model are combined in a weighted sum to obtain a unique prediction. The combination of these simple model predictions is performed in the model mixing module, as seen in Figure 1. Figure 2 shows that there are 552 simple models used as inputs by the PAQ8 algorithm. The output of each model provides input to the model mixer which trains the links of its network independently and online. Therefore, the parameters of this network generally do not converge to fixed values. PAQ8's network consists of a hidden layer divided into seven sets. From each set a single hidden node is selected based on the context from which the network is making its prediction. The selected node is used in the calculation of the single output prediction. Each set selects a node according to a unique mechanism.

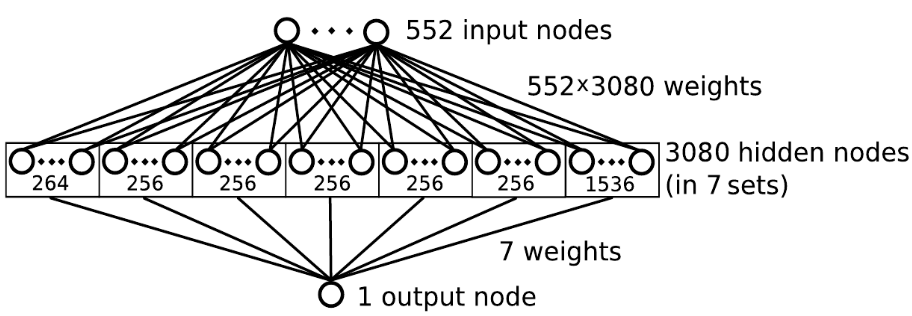


Figure 2: The architecture of PAQ8's model mixing module [9]. The predictions of 552 simple models are aggregated into a single prediction.

The single prediction of the model mixing module is passed through the adaptive probability map before reaching the arithmetic coder, as seen in Figure 1. The adaptive probability map is a post-processing step that in practice reduces the error in prediction by 1%. The predicted character (bit) and a small context are used to make a look-up into a two dimensional table and the entries of the table are updated to reduce error. The arithmetic coder then compresses the probability distribution that describes the data sequence. PAQ8 is most effective on one-dimensional data. The algorithm makes predictions based on context which is interpreted as linear history. A character is evaluated based on its proximity to other characters in the stream of data. Interestingly, PAQ8 is a deterministic algorithm, lacking any stochastic properties.

3 MIDI pre-processing

A MIDI file can be in one of three formats, single track, multi-track synchronous, and multi-track asynchronous. The type of the MIDI file is specified in the header along with metadata such as the timing division and the number of tracks in the file. The MIDI file body is composed of a collection of meta and musical events. Two important musical events are start events and stop events, where a start event indicates the initial striking of a note and a stop event indicates a note off event. Each musical event includes a specification of a time, a type, a note, a channel and a velocity (referring to the speed at which a note was struck).

Meta events provide extra-musical information. Removal of meta events from a song preserves the integrity of the music. To minimize any confusion caused by the misinterpretation of MIDI events, the PAQ8 algorithm was only trained on note events (combinations of start and end events). This limitation in training also transferred to prediction, as all predicted data was interpreted as sequential note events. The restriction to only note events is akin to observing the sheet music for a song. The notes of the music are preserved but other information such as tempo changes and information about the artist are omitted.

The PAQ8 algorithm performs best on data represented as a temporal stream. MIDI files can be interpreted as a temporal stream but their format does not imply such a structure directly. For this reason, some pre-processing of the midi training files was performed. The pre-processing of MIDI files was carried out with the aid of an open source MIDI library [10]. MIDI training files were converted to a bit representation where only the notes played in the file were preserved. This note-only representation is hereafter referred to as Note Encoded MIDI (NEM). The NEM encoding collapses start and end events into a single simple representation specified by a delta start time, a duration, the note, and the velocity. The delta start time is the time difference between the beginning of the current note and the beginning of the previous note. In order to reduce the size of the encoding, further practical constraints were imposed on the MIDI files; no MIDI file was longer than four minutes and no note could play for longer than four minutes. In this way, only 24 bits were used to encode the time and duration each, 8 bits were used to encode the note, and 8 bits were used to encode the velocity. Overall, a single note was described by 64 bits.

In order to simplify the parsing of MIDI files, the training MIDI files were limited to single track MIDI files. This is due to the fact that the demarcation between tracks in the predicted MIDI file would require a precise way to signal the beginning or end of a track. In general, any MIDI file can be converted into a single track MIDI file, therefore, the limitation to single track MIDI files is not a significant loss.

4 Experiment

PAQ8 compression techniques were applied to the task of predicting music. The MIDI file representation was chosen because it consists of discrete values structured in an audio stream. The PAQ8 algorithm was originally developed by Mahoney [8]. Our experiment makes use of a modified version of Byron Knoll's PAQ8 text predictor [9] to perform MIDI file prediction.

The PAQ8 algorithm was used to predict MIDI music, i.e. to continue or forecast a musical sequence. Prediction of MIDI music was broken into two phases. Firstly, a collection of single track MIDI files provided a means to train the PAQ8 model. Secondly, a short note sequence was presented as the introduction of a song and PAQ8 was made to predict a few subsequent notes. The predictions made by PAQ8 are not completely robust. On many input note sequences PAQ8 makes meaningless predictions. However, we found that PAQ8 was capable of making remarkably relevant predictions for select input sequences.

The first phase of the prediction task, the training, consisted of compression over a set of single track MIDI files. There were 148 single track MIDI files in the dataset. It is desirable for the input to the prediction to be novel and as meaningful as possible. For this reason, the short note sequence was taken from the beginning of one of the songs of the dataset. When this was done that song was omitted from the training set to ensure that it was novel to the model. In the end only 147 songs ever survived to be used in the PAQ8 training set. As discussed in the MIDI preprocessing section the raw MIDI files were not fed directly to the PAQ8 algorithm. Each MIDI file of the training set was converted to NEM format before it was passed to the algorithm. PAQ8 created a compressed archive of all the NEM training files thus producing a probability distribution that described the note sequence trends of the training data. The compression probability distribution provided a base distribution for the prediction task.

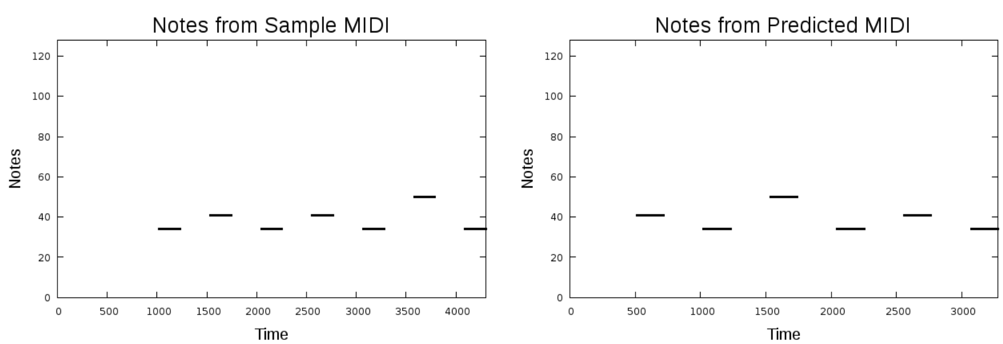
The second phase of the prediction task, the actual prediction, makes use of the probability distribution found during compression to generate music. The PAQ8 algorithm is given one bit of the short note sequence at a time. For each bit received a new prediction is made, resulting in an update of the model parameters. Since PAQ8 learns parameters online the probability distribution is updated for each new bit of the input sequence (as music is generated). As PAQ8 makes predictions it uses those predictions as context. In this way, PAQ8 is able to predict based on trends in the input sequence, trends in the predicted sequence, and trends in the training data.

Once an NEM encoding of notes had been generated by the PAQ8 algorithm the NEM encoding was converted to a MIDI file format. The headers of MIDI files are sensitive to inconsistent information and were not predicted by the paq8 algorithm (since they are not part of the NEM encoding). The header was therefore generated based upon the predicted music. Some meta information was not implicit in the generated notes and had to be manufactured. Predicted music was always interpreted according to the quadruple, $\frac{4}{4}$, time signature. The quadruple time signature is also known as common time and was chosen due to its popularity. The tempo (or beats per minute) was set to 1024 since it allowed for an expression of a reasonably large set of songs. Furthermore, the predicted notes are assumed to belong to a single track. This assumption and interpretation is reasonable since the training MIDI files are all single tracks and PAQ8 had not learnt any way to delimit between tracks of a MIDI file.

5 Results

The predictor took as input a binary file which contains a NEM encoding of a sequence of notes. These notes are meant to be interpreted as the introduction of a song for which the predictor will generate a continuation of the song limited to ten notes.

Successful and meaningful predictions were rare. PAQ8 was given the first ten notes of *Bartok's Suite, Op.14, Allegretto* and asked to predict ten forthcoming notes. The ten notes predicted by PAQ8 were a consistent continuation of the song, in fact they very closely resembled the short note



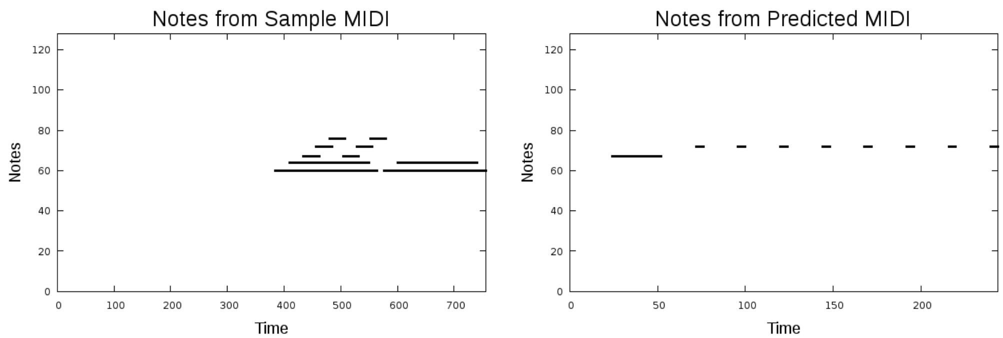
(a) The first ten notes of *Bartok's Suite, Op.14, Allegretto*.

(b) The notes predicted by PAQ8.

Figure 3: PAQ8 was provided the first ten notes of *Bartok's Suite, Op.14, Allegretto*, shown in (a), as a short note sequence defining the introduction of a song to be predicted. The notes predicted by PAQ8 are shown in figure (b). Each horizontal bar represents the duration of a note. The key of that note is represented along the y-axis and the start time and the duration of the note is represented along the x-axis.

sequence provided to the algorithm. In this case, it appears that PAQ8 predicted a continuation of the song primarily based upon the context of the sample input notes it received.

Similarly, when PAQ8 was presented with the first ten notes of *Ave Maria*, it predicted a new sequence seemingly solely dependent on the context of the sample notes. The predictions of both *Ave Maria* and *Bartok's Suite, Op.14, Allegretto* seem to be based on recurrent patterns in the ten sample notes. These predictions are examples of the potential success of PAQ8's ability to forecast. PAQ8 made meaningful predictions based on the note sequence it observed.



(a) The first ten notes of *Ave Maria*

(b) The notes predicted by PAQ8.

Figure 4: PAQ8 was provided the first ten notes of *Ave Maria*, shown in (a), as a short note sequence defining the introduction of a song to be predicted. The notes predicted by PAQ8 are shown in figure (b). Each horizontal bar represents the duration of a note. The key of that note is represented along the y-axis and the start time and the duration of the note is represented along the x-axis.

Unfortunately, PAQ8's performance on non-repetitive music was sub-optimal. For many input sequences the predicted notes are uninterpretable as music. More precisely, predicted notes were often fully described by a series of zero bits. A zero bit note translates into silence. Upon registering the sample notes PAQ8 often failed to make an intial note prediction. As PAQ8 makes predictions it uses those predictions as context; PAQ8 detects patterns in its own predictions. When PAQ8 makes a poor initial prediction, the poor prediction can propagate to further note predictions.

6 Discussion

The choice of the tempo for which to interpret predicted music is arguably arbitrary. Normalizing music to the 1024 signature before encoding and training would eliminate another variant in the data. According to the MIDI format, time steps are required to be round integers. It would be preferable to normalize to a large tempo since a partial note (such as a quarter or half note) could be accurately represented.

PAQ8 has been shown to be very successful for text prediction in that it is capable of producing coherent sentences [9]. This application of PAQ8 to MIDI files provided a few encouraging results and many meaningless predictions. The NEM encoding used to train the PAQ8 algorithm allows for an enormous domain of notes to be described. Over the training set it is highly unlikely that every note event was seen even once. In the case of text prediction, it is very likely that each character is seen at least once and even many times. A note and a character can be seen as atoms of each domain; the prediction algorithm aims to produce a meaningful sequence of atoms. The prediction algorithm provides poor predictions when an introductory sequence has little overlap with the domain of the training set. Improvements upon the MIDI prediction could be attained by using a vast training dataset or by simplifying the encoding used for training. Simplifying the encoding would require restricting the set of describable notes but such a restriction may not be unreasonable.

Here it was shown that PAQ8 was able to make some predictions that generated music according to a trend in the sample notes. This prediction is consistent with the sample notes, but may be inconsistent with the true continuation of the sample song. It would be interesting to see if PAQ8 could be made to successfully predict notes that resembled the continuation of the original sample piece. A continuation consistent with the original music could be interpreted as either accurate recall or creative music generation. To obtain accurate recall of the original piece, the original music could be incorporated into the training set. In this case, PAQ8 would serve as a memory device. To obtain creative music generation it could be beneficial to limit the training set to music consistent with the genre or composer of the original piece.

Compression algorithms can be applied as a mechanism to solve various other problems. In particular, classification can be achieved via compression algorithms. A good compression algorithm can be used to classify data. Over a set of similar data a compression algorithm can take advantage of the homogeneity of the data to obtain better compression ratios. Given a set of data consisting of two classes, partitioning that set according to the data classes and compressing each partition separately should yield better compression ratios than another partitioning. Though PAQ8 was not proven to excel at prediction, it may prove to be a useful classifier. One possible application of PAQ8 would be as a classifier of MIDI music by genre.

Acknowledgements

We would like to thank Byron Knoll for his efforts in explaining PAQ and the feedback he provided on the performance of this model.

References

[1] Andreja Bubic, D Yves von Cramon, and Ricarda I Schubotz. Prediction, Cognition and the Brain. Frontiers in Human Neuroscience. Number 4:25, 2010.

[2] Kara D Federmeier. Thinking ahead: The role and roots of prediction in langage comprehension. Psychophysiology. Number 44, pages 491–505, 2007.

[3] Adrian Staub and Charles Clifton Jr. Syntactic prediction in language comprehension:evidence from either...or. Journal of experimental psychology Learning memory and cognition. Number 32:2, pages 425–436, 2006.

[4] Marcus T Pearce, María Herrojo Ruiz, Selina Kapasi, Geraint A Wiggins, and Joydeep Bhattacharya. Unsupervised statistical learning underpins computational, behavioural, and neural manifestations of musical expectation. NeuroImage. pages 302–313, 2010.

[5] Valorie N Salimpoor, Mitchel Benovoy, Kevin Larcher, Alain Dagher, and Robert J Zatorre. Anatomically distinct dopamine release during anticipation and exerience of peak emotion to music. Nature Neuroscience. Number 14, pages 257–262, 2011.

[6] Joel Ratsaby. Prediction by Compression. Cornell University, 2010.

[7] Marcus Hutter. The Hutter Prize. http://prize.hutter1.net/, accessed December 2011.

[8] Matt Mahoney. PAQ8 Data Compression Program. http://mattmahoney.net/dc/, accessed December 2011.

[9] Byron Knoll and Nando de Freitas. A Machine Learning Perspective on Predictive Coding with PAQ. University of British Columbia, 2011.

[10] Jeff Koftinoff and Vadim Madgazin. C++ MIDI Library - jdksmidi. http://www.jdkoftinoff.com/main/Free_Projects/C++_MIDI_Library/, accessed December 2011.