Machine Learning Course Project Report

Peter Snyder, Xiang Huo

May 10, 2013

1 Introduction

Our problem is to generate a machine learning algorithm that writes original songs in the style of soundtracks to Nintendo Entertainment System games. The songs this algorithm produces should more closely resemble the original, human written songs as the amount of training data available to the algorithm increases.

For the purposes of this project we define a song to be an ordered sequence of bundles of notes, where a bundle refers to a set of notes intended to be played simultaneously. This definition of a song intentionally excludes consideration of tempo and percussion from our problem scope. We choose not to consider tempo because the Nintendos audio hardware only allows a single tempo per song, which makes the prediction problem uninteresting. We choose to disregard percussion in order to simplify the problem in front of us, though with the expectation that the learning and synthesis methods used in this project could be extended to to percussion at a later date.

2 Background

2.1 The Nintendo Sound System

The original Nintendo Entertainment System included a sound chip that, in its most common configuration, could play five channels of monophonic tones. Each of the channels have a fixed sound or instrument, meaning that composers were limited to the kinds and combinations of tones they could produce. The five types of tones the NES could produce are as follows:

2 pulse wave channels
1 triangle wave channel
1 white noise channel
1 low resolution sample channel, able to play low resolution, 7 bit PCM audio data

While technically all five of these channels could produce musical tones, typically only the first three channels were used to produce melodic, melody or a harmony sounds. The white noise channel was typically clipped and played for very brief durations to produce a rudimentary percussion sound. The sample channel, when used at all (the limited storage space on the NES cartridges made this channels use infrequent) generally played sound effects like explosions or speech recordings.

Songs were performed by the NES hardware with a fixed tempo. The speed at which the audio hardware moved from one bundle of notes to the next was fixed for for the duration of each song. Composers wishing to have the effect of fast and slow moving sections of songs could only do so by increasing the length that each note was played for during slow sections, and decreasing it again to create the feel of a fast moving section. The effect in traditional scored music would be keeping the tempo of the piece constant, while switching the notes from quarter notes to whole notes (in the case of moving from a fast to slow section).

2.2 Data Set

Our dataset consists of thirty nine NES games, for a total of 529 songs and 552101 unique observations. All of the selected games were published between 1985 and 1995, the time during which the NES was being popularly commercially developed for. The games in our dataset were hand selected, and so not scientific. We selected games that have well known, well regarded and / or particularly memorable soundtracks.

Similarly, we avoided games with atonal, sparse, or a-musical soundtracks. While we selected games from a range of genres, years, and publishers, we realize that this selection process is subjective and imperfect.[1]

2.3 Format of Audio Data

Audio data is stored on NES cartridges in solid state ROM chips. Audio data for the first four audio channels is stored in a symbolic format, with the audio hardware needing to looking up the correct frequency and waveform for a given symbol. Audio data for the fifth, sample channel was stored as low resolution PCM audio data.

2.4 Audio Preprocessing

We extracted the audio data from each of the thirty nine games in the format of Nintendo Sound Format (NSF) ROM data. Then, in order to make it easier to parse, process and write, we converted the data to standard MIDI format. This transformation was lossless in the features relevant to this project.

MIDI is a music format popularly used for music composition, performance and recording. Like NSF, it is a symbolic representation of audio information, consisting of instructions for how to play a song, rather than a recording of a previous performance. Also like NSF, it has a concept of channels, or independent instruments performing simultaneously. In all features we considered for this project, MIDI can be though of as having a strict superset of the features of NSF. This allowed us to train our models on MIDI representations of the NSF data using existing MIDI tools, rather than needing to write new parsers and generators for MIDI data.

MIDI records notes on a scale of 0 to 127, with 0 being C0 on the piano (effective silence) and 127 being G10. Since each NES song contained three instruments playing simultaneously, our problem space consisted of series of one of 128^3 combinations of notes.

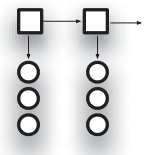
3 Methodology

We created two types of models to attempt to solve this problem. One set of models was based around different orders of hidden Markov models (HMM), the other a more complex Bayes network. The details of each of these models are provided below, followed by an explanation of how we used our dataset to train these models.

3.1 HMM Models

In all of our HMM models, we consider the observed states to be a sequence of tuples, with each tuple representing the notes being played on the first, second and third audio channel. The emission probability in this model is always equal to 1, and the transition probability of moving from state n to state n+1 is equal to the count of the number of times the set of notes being performed at time state n+1 follows the set of notes being played at time state n in the training collection, divided by the count of the number of times the combination of notes being played at time state n appears in the training collection.

Figure 1: HMM model



[1]Complete listing of included games: 1942, Adventure Island, Adventures of Lolo 1, Adventures of Lolo 2, Adventures of Lolo 3, Bad News Baseball, Bakushou Jinsei Gekijou 1, Bakushou Jinsei Gekijou 2, Bakushou Jinsei Gekijou 3, Baseball Star 1, Baseball Star 2, Bionic Commando, Castlevania, Castlevania 2, Castlevania 3, Chip N Dale Rescue Rangers, Contra, Double Dragon, Double Dragon 2, Double Dragon 3, Duck Tales, Duck Tales 2, Final Fantasy, Mega Man, Mega Man 2, Mega Man 3, Mega Man 4, Mega Man 5, Mega Man 6, Metal Gear, Metroid, NARC, Ninja Gaiden, Shanghai, Shanghai 2, Super Mario Bros, Super Mario Bros 2, Super Mario Bros 3, The Legend of Zelda

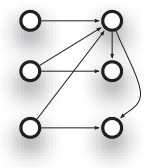
For higher order models, where began finding series of sets of notes in our validation set that were not in our training set, we used LaPlace smoothing to avoid the zero probabilities. In these cases (HMM models with orders 6 and above) we simply added 1 to the numerator and 128^3 to the numerator for all observations.

We tried several different orders of HMM models, where order means the number of previous steps that were examined in each song (first order being unigrams, second order being unigrams, etc.). We considered HMMs of order 1 through 7.

3.2 Bayes Network Models

We also created a more complex Bayes network, expecting to be able to take advantage of commonalties we expected to find in the NES songs. We expected to find that the first audio channel would carry the melody most of the time, so the rest of the notes in the song would move to accommodate the melody track. Similarly, we expected that the combination of notes in the previous time state would have been selected to accommodate the melody in the next time state.

Figure 2: Bayes network model



We tried to capitalize on this 'insight' in several ways. First, we calculated transition probabilities of each not independently, instead of only considering the probability of each set of three notes transitioning to the next. This is captured in the network by the edges in the network moving for each channel, moving from channel x in the previous time state to the note on channel x in the current time state.

We next hoped to take advantage of the idea that the notes in a given time state would be chosen to 'set up' the melody in the next time state by adding edges from each note in each channel to the melody note in the next time state.

Finally, we tried to take advantage of the idea the the melody note in a given time state would govern or control the other two notes in the same time state by adding edges from the note in the first channel to the notes in the second and third channels.

The probabilities of these seven transitions do not sum up to 1, and thus do not create a true probability. However, the sums of these transitions are in proportion to a probability, with a larger sum representing greater total likelihood, and a smaller sum always being less likely. While this would not be the case if there were ever any zero probabilities in the network, we address this through LaPlace smoothing.

Probabilities for each transition / edge were calculated as bigrams, the 'tendency' of any transition between time states is equal to the sum of the below terms:

Table 1: Average scores given by HMM

Let:	Tendency of O_{x+1} following O_x:
$o_{x,y}$ represent the note being played at time state x, channel y $P(x_{x,y}, o_{a,b})$ equal $Count(o_{x,y} \rightarrow o_{a,b})/Count(o_{x,y})$ And the LaPlace smoothed version being : $Count(o_{x,y} \rightarrow o_{a,b}) + 1/Count(o_{x,y}) + 128$	$P(o_{x,0} \rightarrow o_{x+1,0})+$ $P(o_{x,1} \rightarrow o_{x+1,1})+$ $P(o_{x,2} \rightarrow o_{x+1,2})+$ $P(o_{x,1} \rightarrow o_{x+1,0})+$ $P(o_{x,2} \rightarrow o_{x+1,0})+$ $P(o_{x+1,0} \rightarrow o_{x+1,1})+$ $P(o_{x+1,0} \rightarrow o_{x+1,2})+$

4 Experiment

We used the songs from the training set (2/3rds of our data) to train both the HMM and Bayes net models. We divided the set of real songs into two subsets. One, which contains 388 music files, was used to train models as training set. The other, which contains the rest songs, is used as the test set.

In this section, we design a two part experiment to determine how well our model wells. For the first part, we use both HMM and Bayes net models to score the midi songs in our training set. The purpose of this is to see whether our models can distinguish real songs from noise (random songs) clearly.

The first experiment is to use our models to score songs. In this part, we use the two types of model to score 'real' songs and random songs. To get the results of our experiment, we used our testing subset of 218 songs from our larger dataset of NES soundtrack songs, along with a set of 100 randomly generated songs. These random songs are created by a random function. They have the same number of tracks as real songs have, but the pitch and duration of each their each note is decide randomly, and independently. For our model to do a good job distinguishing random songs from 'real' songs, our models should give much lower likelihood scores to the randomly generated songs than from the real songs'.

The second experiment is to use our models to synthesize new songs.

4.1 Scoring Music Files

In this part, we use the models trained by our training data to score existing songs from the training set, along with randomly generated songs. The expected result is that there is obvious difference between the scores given to the 'real' songs and the scores given to the randomly generated songs.

When trying to evaluate each song, we treat all notes in the song as a sequence of observation. We then score the likelihood of the transitions between the each of these observations. The likelihood scoring function of HMM is defined as following:

$$Score = Length_{song} \log[p_{transition}] = Length_{song} \log[\frac{1+count(o_{i-n}o_{i-n-1}...o_i)}{128^{3^n}+count(o_{i-n}o_{i-n-1}...o_{i-1}o_i)}]$$

where n means HMM's order and 128 is count of possible values for one note in each channel. We use the logarithmic value of possibility for this sequence as the song's score. So the closer to zero the score is, the higher possibility the song has. Higher possibility of observations also implies that this sequence accord with the rule behind training data, therefore this sequence is more likely to be considered as a song.

Another issue about HMM model is the order of the HMM used, or the number of previous states we consider when trying to predict the next state. To get the value which can classify music best, we range its value from 2 to 6 on the same data set. The result we get shown in Fig.3

Based on this result, we can see almost all random songs get the same score. The reason of this is that we are unable to get data set large enough, so we cannot find even 1 example in data set for almost all observations in random songs. In this case, we execute the following method to calculate transition probability:

$$p_{transition} = \frac{1+count(o_{i-n}o_{i-n-1}...o_i)}{128^{3^n}+count(o_{i-n}o_{i-n-1}...o_{i-1}o_i)}$$

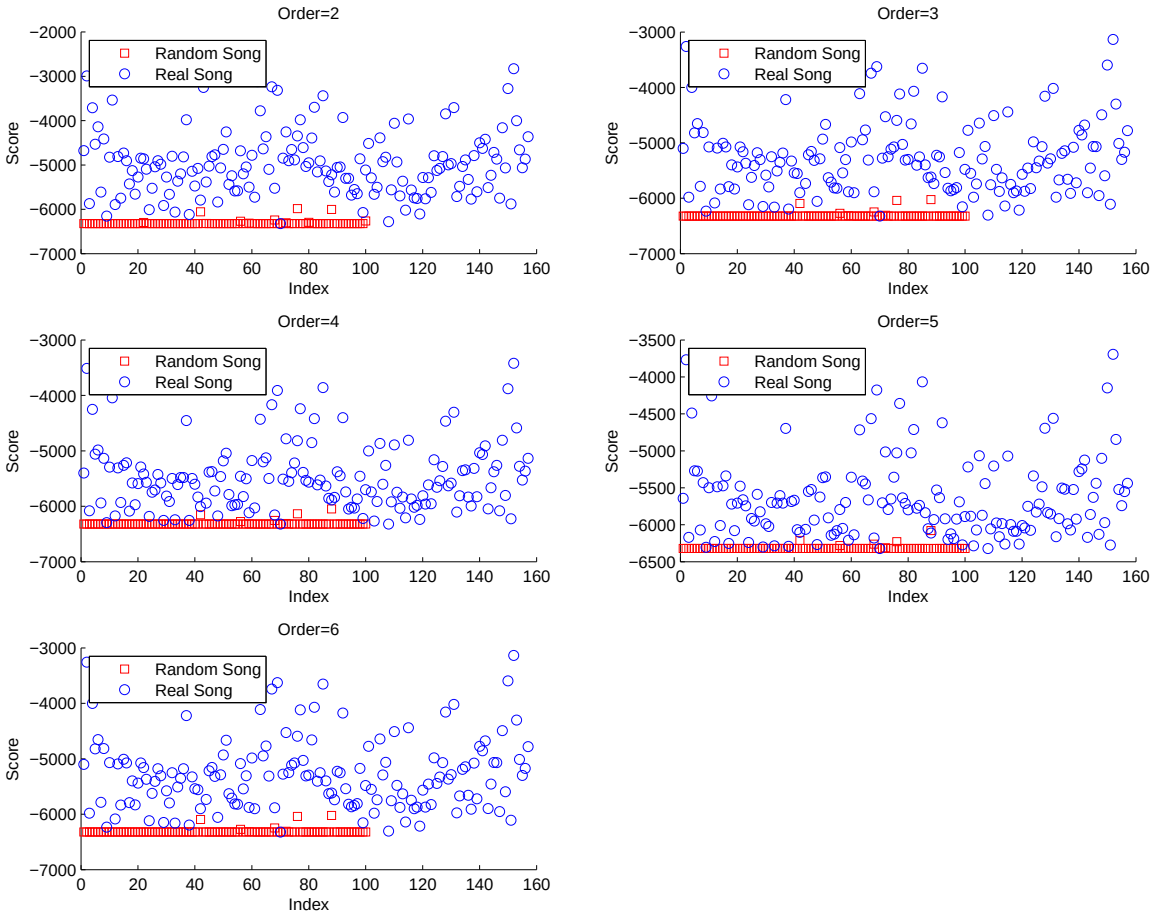
Therefore, if all observations' counts from one songs are zero, then their possibility would be a fixed value $\frac{1}{128^{3^n}}$. The score of this kind of songs would be $Length_{song}\frac{1}{128^{3^n}}$.

From the result, we can also see that because different styles of real songs, the scores of real song distribute widely in the score space. The average scores of random songs and real songs with depth from 2 to 6 are shown in the Table 2:

Table 2: Average scores given by HMM

	Order=2	Order=3	Order=4	Order=5	Order=6
Real	-4368.1	-4603.7	-4787.1	-4939.2	-5068.8
Random	-6309.6	-6312.0	-6314.0	-6316.0	-6317.7

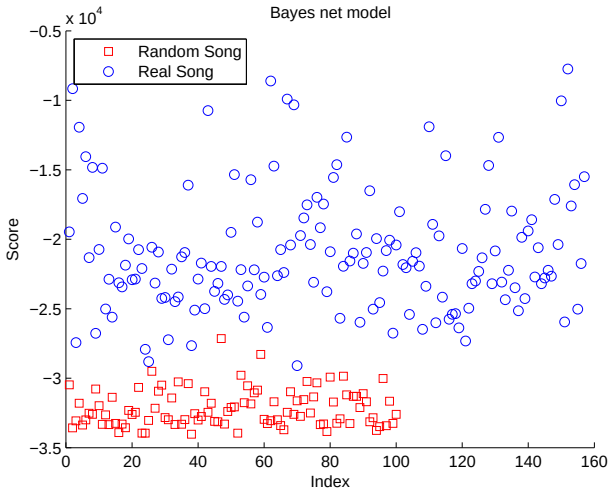
Figure 3: Scores given by HMM



With the increase of depth's value, the distance between real songs and random songs is getting smaller. The reason is that it is more difficult to find instances with higher order. The result also shows that HMM cannot classify these two classes well enough.

Compared with HMM model, Bayes net model gets better results. The scores on the same data set given by Bayes net is shown on Fig.4.

Figure 4: Scores given by Bayes net



In Bayes net model, the average score of real songs is -18441.5, while the average score for random songs is -32180.9. The distance between them is more obvious. Therefore, we can get the conclusion that Bayes net model is better than the HMM model when scoring songs to classify random nose verse actual songs.

4.2 Prediction and Synthesis of New Songs

We also tried using both of these models, HMM and Bayes net, to predict a new song, given a staring point and the probabilities built from the training data.

Our initial strategy was to randomly create a huge number of random songs (millions and millions) and then simply select the song with the highest likelihood according to our models. However, we quickly found that even the most likely songs generated with this method still received extremely low probabilities. This is because the search space of randomly generated, 100 state songs we were randomly choosing from was far too large. Since there are 128 notes, and 3 channels each playing one of those notes at each observation, we were randomly choosing from 128^{3*100} possible songs, the vast, vast majority of which were random garbage. Even the best sounding songs we generated with this method sounded awful.

So we design another way to predict new songs: a simple greedy algorithm that would, at each transition, order the non-zero likelihoods of transitions to next states. Selecting the most likely from this set of transitions would maximize our chance of creating a reasonable sounding song However, it also is deterministic, meaning that our algorithm would generate one-and-only-one song for each starting point. In order to have more possible songs, we altered the algorithm to randomly select a transition among the n^{th} most likely transitions, instead of always selecting the most likely transition. Another trick we use is that when the previous observation is so rare in our data set that the count of it is 0, the function will randomly choose the next observation from all possible observations.

4.3 Evaluation

One of the most difficult problem for this part experiment is how to evaluate the song we predict. We cannot use to the model which creates one song to evaluate it again, while we are unable to get the access to another different model to score our new songs. Therefore, we decide to make survey to let several persons listen our songs and give their scores.

There are 10 songs in our survey: 2 songs are real songs. 2 songs are created by our Bayes net model, 2 songs are created by a 2nd order HMM, 2 songs are created by a 4th HMM model, and the remaining 2 songs were created randomly. Then we asked 30 people to listen these songs and rank them from the most to least favorite. The result of this survey is shown in Table 3.

The value of each cell is the rank given by persons, 1 means the best one, while 10 means the worst. From this result, we can see that 2 random songs get the lowest rank. So our songs are much better that random songs. One the other side, although most persons give the highest rank to those 2 real songs, we still can see that a part of participants give some predicted songs higher rank than real songs. It implies that some people prefer our predicted songs rather than the real ones. Considering some participants are familiar with game music, the average rank would a little higher if we continue this survey in a larger group of persons.

5 Discussion and Conclusion

5.1 The Surprising Success of the HMMs, the Crushing Defeat of the Bayes Net

Despite the relative simplicity of the method, HMMs performed as well as, or better than the Bayes network based models. We attribute this to several factors. First, the nature of HMMs guarantees that, given an X order HMM, any subset of X adjacent sets of notes are guaranteed to have occurred at least once in a 'real song from the training set. This makes the HMM generated songs, in a sense, overlapping medleys of other, training set songs. Our Bayes Network strategy does not have this feature, which, unsurprisingly in retrospect, leads to songs that sound less like songs written by humans.

Table 3: Survey result

Real1	Real2	HMM21	HMM41	Bayes1	HMM22	Bayes2	HMM42	Rand1	Rand2
3	1	5	2	8	7	4	6	9	10
2	3	1	8	6	4	7	5	9	10
2	1	5	8	3	6	7	4	9	10
1	4	5	3	2	6	8	7	10	9
1	3	5	10	2	6	4	7	9	8
1	2	3	5	6	4	7	8	9	10
1	2	4	6	3	7	5	8	9	10
2	1	6	7	5	5	3	8	9	10
3	1	5	8	2	6	4	7	9	10
1	2	4	3	5	8	6	7	10	9
1	2	3	4	7	5	6	8	9	10
1	7	3	4	2	5	6	8	9	10
1	2	4	8	7	3	6	5	10	9
3	6	4	7	1	5	2	8	10	9
2	1	4	3	6	7	10	5	9	8
1	2	4	3	7	6	8	5	9	10
2	1	5	7	3	6	4	9	8	10
2	1	5	3	6	7	8	4	9	10
1	2	3	4	5	7	8	6	10	9
1	2	3	4	5	7	6	8	9	10
1	2	3	6	4	8	5	7	10	9
1	3	4	2	7	5	8	6	10	9
1	2	4	6	5	7	3	8	10	9
1	2	3	6	5	4	7	8	10	9
2	1	3	6	10	4	8	9	5	7
10	9	1	2	7	3	8	4	6	5
2	1	6	3	4	7	8	5	9	10
2	1	3	4	6	7	5	8	9	10
1	3	4	2	7	6	8	5	9	10
2	3	6	1	8	4	7	5	9	10
2	1	6	4	3	10	5	7	9	8

Another reason the Bayes Net model performed less well than we expected is that we may have placed too much weight on the first / melody channel. This lead the model to favor songs that have the most frequently occurring notes in the first channel over all other considerations (such as whether that frequently occurring note harmonizes with the other notes being played at the same time). Even worse, after going back to look closer into why they Bayes Net model performed less well than we expected, we realized that the assumption underpinning the model, that the first channel carried the melody, was often wrong. So even worse than just optimizing for the most frequent note in the most important channel to the detriment of the other channels, we were sometimes optimizing for the most frequent note in a (comparatively) unimportant to the determent of the important channel.

5.2 Improvements and Future Work

We have several ideas for how our models could be improved. First, and least interestingly, we could gather more data to train against. Some of our training songs contain non-melodic sections, which would somewhat pollute our training counts. Assuming that these non-melodic sections are over represented in our training data, more data would diminish the effect of these non-music sections and lead to more useful training data.

We also have several ideas for heuristics that we could build into our model that might improve performance. By expanding our model to build in common features of pop music (ex. that transitions are more likely to occur on even numbered beats than on odd numbered beats, and that transitions are even more likely to occur on every fourth beat) and music theory (ex. that if the the previous to states were on the one and the four chord, its more likely that the next transition should be to the fifth cord), its likely that our model could generate more ′human′ sounding songs.

Finally, a more complicated but more exciting approach to approach algorithmic song writing would be to use a hierarchical HMM approach, first trying to detect transitions between sections of a song, then, within each section, between key or chord changes, and then finally transitions between notes within a given key or chord. Such a model would better capture the structure and constraints of notes in a song, and would even model the writing process that many songwriters go through when creating music.