Barebones Music Generation

Ben Rocklin (brocklin@stanford.edu), Jacob Meisel (jameisel@stanford.edu)

Motivation and Overview

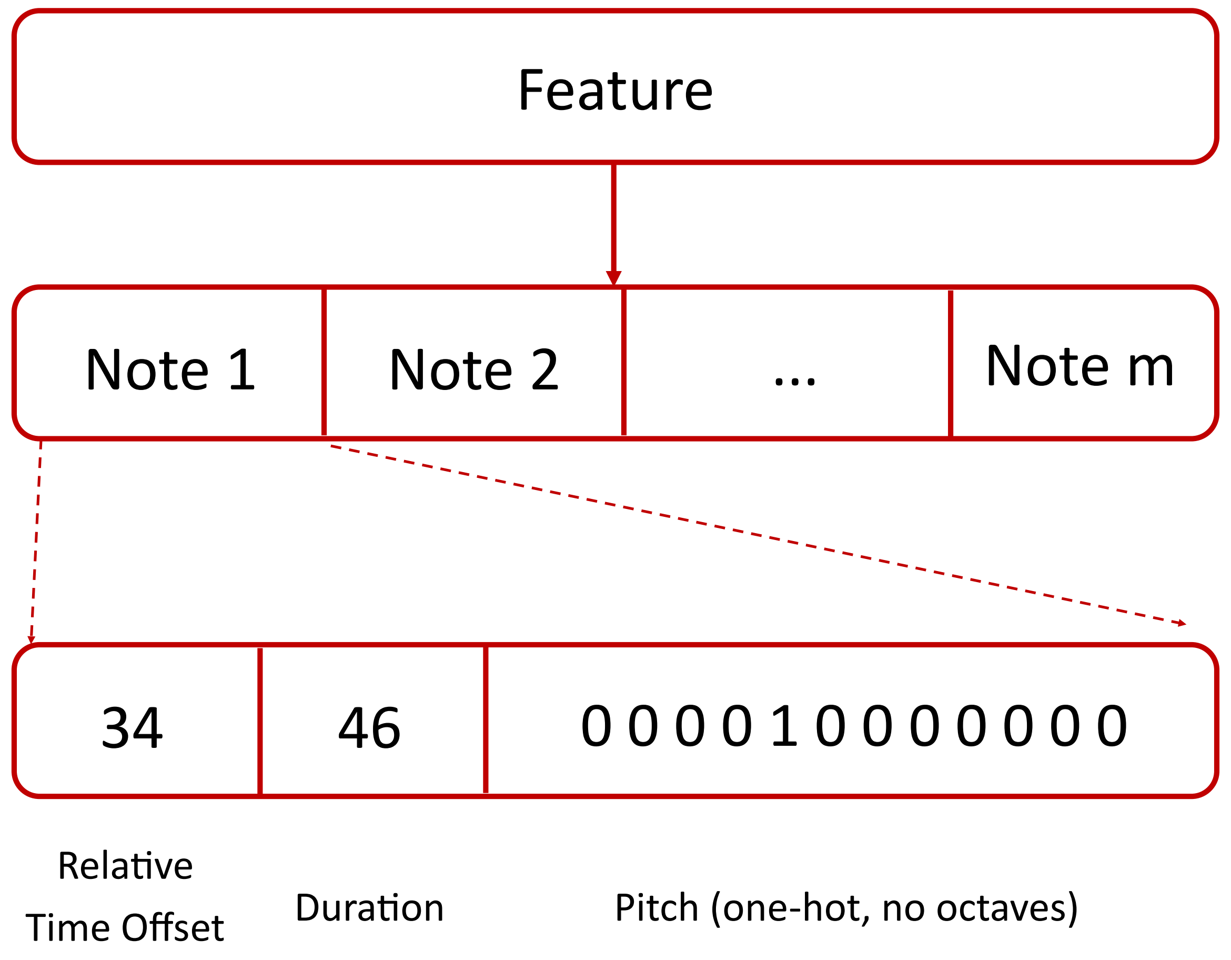
Given the extreme complexity of existing music generation models, we sought to answer whether or not complicated music pieces can be replicated — or if new ones can be generated — using low-complexity models (such as neural nets and linear models) given nothing but the prior notes. This did not turn out completely as expected, but there was limited success in some of the songs. More details on our models used and exact prior notes given as input are further below.

Data

Our data source was 327 MIDI files found online [1]. Since our task is notewise prediction, we had three labels we wanted to predict: pitch (how high a note is), duration (how long the note is played for), and delay (how long until the next note in the song is played), and so we trained a model to predict each, forming three y vectors. Each song contained many notes on the order of thousands, and these were then modified to become a feature vector for each of the label tuples as detailed in the section below, where each feature vector was a row in our X matrix. All time measurements are in MIDI ticks to become time invariant.

Features

Our features are the notes that are played before the note we are trying to predict, so the features take on a similar form to the labels. They can be thought of as vectors of notes, where each note is three elements. The first element of each note is the “relative time offset,” or, more simply, this is how long the prior note was played before the note that is being predicted. The second element is the duration, or how long the prior note was played for. The last element is the pitch, which is represented as a one-hot vector; in our models so far, we haven’t factored in octaves, so this is of length 12 (each element is between 0—C and 11— B flat). See below:



Feature

Note 1 | Note 2 | ... | Note m

34 | 46 | 0 0 0 0 1 0 0 0 0 0 0 0

Relative Time Offset | Duration | Pitch (one-hot, no octaves)

Results

		Training Set Performance	Test Set Performance
Delay Models (RMSE):	Linear Regression (m=3)	152.3	151.4
	Linear Regression (m=5)	151.0	149.4
	Neural Network (m=3, 4 layers, 100	128.9	131.7
	Neural Network (m=5, 4 layers, 100	116.4	122.4
Duration Models (RMSE):	Linear Regression (m=3)	271.8	269.2
	Linear Regression (m=5)	269.1	272.5
	Neural Network (m=3, 4 layers, 100	250.5	251.7
	Neural Network (m=5, 4 layers, 100	232.7	248.7
Pitch Models (% accuracy):	Linear Regression (m=3)	19.86	19.87
	Linear Regression (m=5)	21.91	21.87
	Neural Network (m=3, 4 layers, 100	27.19	25.05
	Neural Network (m=5, 4 layers, 100	28.62	26.65

Train/Test Set Sizes:

	Train Set Size	Test Set Size
m=3	643598	113577
m=5	642412	113367

To the left are our statistics for our models’ accuracy on predicting notes given different numbers m of prior notes for each of the three main labels and above are our train/test split sizes. Obviously, the generated songs cannot be put onto this poster, but they are available to be listened to in person on demand (note: a link to them online, along with additional data for the pitch models’ performance, will be available in the final report and on demand in person)

Models

Our models were a combination of varying our prior note factor parameter denoted by m (the number of prior notes before each note we want to predict, or the number of priors in our feature vector) and changing between using a linear/logistic regression model baseline and a neural network. Additionally, we trained neural networks with a varied number of hidden layers and neurons in each layer, using ReLU for all our activation functions (except for predicting pitch, which is a multiclass classification problem, so we used softmax there).

Softmax:

$$g(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

ReLU:

$$g(z) = \max(z, 0)$$

Discussion

To be frank, while we considered that our results could be disappointing, we did not expect them to be this bad or struggle in the areas that it did. Namely, we expected our pitch model to struggle more than it did, while we expected our other models to perform well, while the opposite was true. One important thing to note is that when several different notes were played at the same time, during prediction, each note had the same set of priors. Thus, every time two or more notes were played at once, the same note would be predicted for both notes, which artificially drove our pitch accuracy down, meaning that the results on the left seem deceptively low for pitch when they’re actually okay. This manifested in our generated songs, where the note progressions seemed acceptable, but the notes were seemingly played at strange, random intervals. In hindsight, it may have been worth trying other models such as Naïve Bayes for prediction and anticipate multiple notes earlier. For the other two labels, we believe that barebones music generation as we did it was not enough, and more information is required to predict this, or it may not be possible to predict accurately at all.

Future

It might be worth trying to train a left hand and right hand model separately for our predictions and generators in the future, as it’s possible that merging the left and right hands’ notes caused issues in our model. Additionally, we would spend time adding different features and trying larger values of m in order to try and improve performance on this model, but we unfortunately ran out of time to try additional options.

References

[1] B. Krueger. (1996). “Classical Piano Midi Page” [Online Midi repository]. http://www.piano-midi.de/