Music Composition with RNN

Jason Wang (zwang01@stanford.edu)

Stanford University - Statistics

PREDICTING

The goal is to train a machine to generate music in a freestyle manner. We use the Recursive Neural Net (RNN) with its power to model sequential data such as generating text, predicting trends in time series, and classifying sentiment of texts. In our problem, we feed our RNN segments of music data to train it to produce music on its own. In doing so, we hope the RNN will learn dependencies between notes and the conditional probability of notes in sequence so that we can generate new and original sequences much like a HMM or Markov/n-gram models without restrictions.

DATA

We process the entire Nottingham files of songs which contains 762 songs, each over a minute long. We do a 70-30 split between training and test data for sake of comparison with the other methods we use: random and n-gram (In fact, test data is not necessary at all for our generative model because we're not looking to reproduce music exactly).

We extract the melody from each song and divide each measure into 8 segments where each segment $\boldsymbol{v_i}$ is equal to the value of the note played at time $\boldsymbol{i}$. For each song, we extract the sequence $\boldsymbol{v_i}, \boldsymbol{v_{i+1}} \ldots, \boldsymbol{v_{i+31}}$ for all possible timestamp $\boldsymbol{i}$ to construct the features and train the RNN to classify $\boldsymbol{v_{i+32}}$. This gives us a total of 154992 training sequences.

FEATURES

The only explicit feature is the **pitch quality** of the note. This is the 88 encoding of piano keys to numbers where 21 is the lowest A on the piano and 109 is highest C. Implicitly, we take advantage of sequential information so this includes information such as

- **Duration of the notes**

  [C, C, C, C, G] indicates that C is held for 4 beats while G is held for 1 beat.

- **Transition probability of notes**
- [C, E, G] is a more likely musical transition than [C, F#, D].

MODELS

N-GRAM

From the training set, we examine all sequences of notes for n beats. This gives us a massive dictionary of all the short sequences of musical expressions and the probability each phrase is used in music. Then we can compute

$$p(x_t | x_{t-1}, \ldots, x_{t-n}) = \frac{\#\, times\ [x_{t-n} \ldots, x_t]\ observed + c}{\#\, times\ [x_{t-n} \ldots, x_{t-1}]\ observed + kc}$$

where c is the smoothing constant and k is the total possible values the notes can take.

We generate music by sampling the first n beats of a random song from the test data and iteratively sample the new note based from the learned probabilities above.

RNN with LSTM

A Many-to-One RNN is define as:

Parameters: $W_{hh}, W_{xh}, W_{hy}, T$

Input Layer: $X_i\ ,\ i\ in\ 1{:}T$

Hidden Layer: $h_i \in [-1, 1],\ i\ in\ 1{:}T$

Update: $h_i = \tanh(W_{hh} h_{t-1} + W_{xh} x_i), h_0 = 0$

Output: $y = W_{hy} h_T$

The LSTM cell adds more parameters and has a more complex update step (omitted here), but the mechanisms are still the same. Training is done by back-propagating from the output sampled in batches to increase the speed.

We seek to minimize the multiclass log-loss defined as

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \log p_{ij}$$

M is number of labels
N is number of samples
y_{ij} is binary indicator of instance i is labelled correctly
p_{ij} is the model probability of assigning label j to instance i

RESULTS

Model	Log-loss	Training Accuracy	Test Accuracy
Random	--	0.024	0.024
3-gram* (*best with test set)	--	0.369	0.315
RNN-LSTM (50 epochs)	0.7460	0.766	0.588

Accuracy is defined as the percentage of the next 32 beats where the generated music and the actual music sequence match. Mathematically, it is defined as the term below.

$$Acc(y_{t,} \ldots, y_{t-n+1} | x_{t,} \ldots, x_{t-n+1}) = \frac{1}{m}\sum_{i=1}^{m} \frac{\sum_{j=t-n+1}^{t} I(y_j = x_j)}{n}$$

However, since we are mainly concerned about the ability for the RNN to generate novel music, we instead minimize the training multiclass log loss and are only concerned with this quantity. Accuracy is simply just a benchmark to measure some related metric. Qualitative analysis of generated samples is still required.

DISCUSSION

Despite the free range of music, music generated had musical qualities.

Melody was very distinct and lyrical (small steps and alternating between ascending and descending in pitch). Chord progression/cadences was also present as the I-IV-V-I progression was widely used in all generated samples. Finally, the algorithm seemed to generate plausible music after 100 beats even though we only fed a seed of 32 beats.

Music from the 3-gram was also rather lyrical but music composed lacked a feel that something more intelligent wrote them since the machine is unable to store data on more than 3 beats previously whereas RNN repeated patterns over 32 beats.

FUTURE WORK

Collection of more data especially complicated melodies would be nice. We can also learn rhythm explicitly (How do we encode?). Finally, can we find a way to generate harmony? Is there a loss function that we can train for learning musical qualities of two voices instead of one such as harmony and counterpoint.

REFERENCES

[1] N Boulanger-Lewandowski, Y. Bengio, P. Vincent, Modeling Temporal Dependencies in High-Dimensional Sequences: Applications to Polyphonic Music Generation and Transcription, in Proceedings of the 29th International Conference on Machine Learning (ICML), 2012

[2] K Goel, R Vohra, and J.K. Sahoo, Learning Temporal Dependencies in Data Using a DBN-BLSTM

[3] http://deeplearning.net/tutorial/rnnrbm.html

[4] http://deeplearning.net/tutorial/rbm.html

[5] http://christianherta.de/lehre/dataScience/machineLearning/neuralNetworks/LSTM.[6] Eck, D. and Schmidhuber, J. Finding temporal structure in music:

Blues improvisation with LSTM recurrent networks. In NNSP, pp. 747756, 2002.