Homework Assignment 4:
Generating Music with Recurrent Networks

CSE 253: Neural Networks

Winter 2017

Instructions

Due Friday, March 3^{rd}:

1. Please hand in your assignment via Vocareum. We prefer a report written using LaTeXin NIPS format for each assignment. You are free to choose an alternate method (Word, etc.) if you want, but we still prefer NIPS format.

2. You should submit your code on Vocareum along with your report. For your own purposes, keep your code clean with explanatory comments, as it may be reused in the future.

3. You can use Keras, a Deep Learning library, for the tasks in this assignment, or you may write your own code for basic character level rnn in python or matlab.

4. Please work in teams of size 4-5. In extraordinary circumstances (e.g., you have a highly contagious disease and are afraid of infecting your teammate), we will allow you to do it on your own. Please discuss your circumstances with your TA, who will then present your case to me.

Character level RNN for music generation (40 points)

In this assignment we will explore the power of Recurrent Neural Networks. In the previous asssignment using Convolutional Neural Networks, we had assumed that the data points were IID and temporally independent. However, CNNs are unable to model some tasks where the data have temporal dependencies. For these sequential relationships Recurrent Neural Networks (RNNs) are used. In this problem, we will generate music in abc format. You will train a basic RNN model using characters extracted from a music dataset provided to you and then run the network in generative mode to "compose" music.

Problem

1. **Getting familiar with the data** In this part, we are going to see how we can convert music from ABC notation to a playable format(.midi in this case) online. Go to the website http://mandolintab.net/abcconverter.php. Copy the text from sample-music.txt file (found under Data.zip,which contains music in ABC notation) and hit Submit. Download the tune in midi format and play it on your computer.

   You will be generating the music in a similar format as the sample file, which is ABC format. A sample ABC file is shown in Fig 1.

2. **Read in data**. Read in the data from the input.txt(found under Data.zip) file. This file contains multiple tunes in ABC format, with each tune delineated by <start> and <end> tags.

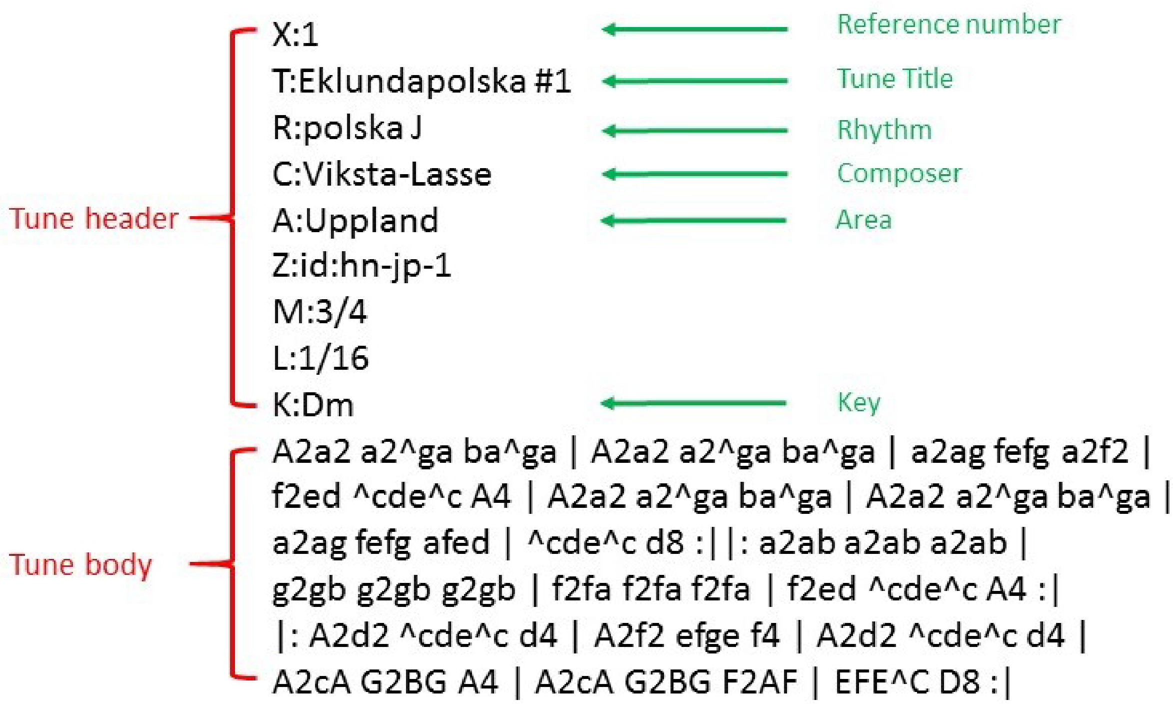

Figure 1: A music file in ABC notation.

3. **Train a network** First, train an RNN to learn the structure of an ABC notation music file through prediction. Your network will take in a slice of the training set - just, say, a batch of 25-30 characters, and given the first character as input, you will train it to predict the second character, etc. So, you are slicing *random* sequences from the training set - not aligned with the beginning of the file. We're not saying this is optimal, it is just how we did it!

   You should use only one hidden layer for this problem. You can either write your own code for the RNN or use the Keras Simple RNN layer to build this network. You should have around 100 neurons in your hidden layer.

   You should use a softmax output and the cross entropy loss. You should divide your data into training and validation set by performing an 80-20 split of data.

   In the training stage, the network takes the ground-truth character of current step as input and predicts the next character. Then the next character is used as input, and it is trained to produce the next character. This is sometimes called "teacher forcing." Once the network has learned well enough, you could try taking the maximum output and feeding that back into the input and train it to produce the next input. Our TAs didn't do that in their version, so take this suggestion with a grain of salt. Another idea you could try is to gradually increase the batch size, once it gets good at a particular length.

4. **Generate music** In the generation stage, the idea is to "prime" the network with a sequence, and then let the network run on its own, predicting the next character, and the using the network's output as the next input. There are at least two ways you could feed back the network output. One is to take the maximum output. We don't recommend this option. A second way is to flip an n-sided coin, assuming n outputs, based on the probability distribution at the softmax layer. You can make the network more or less deterministic by adjusting the T parameter in the softmax, but start with 1.

   Save the sequence to a file, and then use this to convert back to midi format, and play what is generated.

   Note: It might take a lot of time to generate decent music. You can try and generate music after every batch of data, but it took us about 1-2 hours of training. At this point, your network should be able to generate some good tunes in between some not so good tunes.

   For your report, provide:

(a) Generate 6 sample music pieces, two at $T = 1$, two at $T = 2$, and two at $T = 0.5$. They should be of reasonable length. Pick ones that sound the best to you for your report. Provide their ABC notation and music representation generated from http://mandolintab.net/abcconverter.php. Also upload the tunes in midi format on vocareum. Discuss your results. Also report all your hyperparameters. *(15 points)*

A sample output would look like the example shown in Figure 2. Note that this music was one of the best examples generated by our character level rnn in 2 hours; your tune can be shorter than this. Figure 3 shows the music in Figure 2 in standard musical notation.

```
X:44
T:Farscrisue FB cF2 d2Az|B3d fd :|
w: Laka Gan vout B'a
M:3/4
L:1/8
K:Bb
B>G | ABcB | G2cB MBABA | A4c/B/c/B/ cc | BA/G/ BB | G2B2 | cdcA | FABAA | B2z2 | B4z2 | B2A cBA | FAG:|
F2e|B3 GA BB B/G/B | G2B | A F/G/G/A/ B4
M:2/4
L:1/4
K:C
(AGA G2z2 | BBc2 c2A2|B2B2 B2c2 | d2ef (fce)f3g2e| [M:6/8]:
F2A G2B|c2c g2c a4|f2c2|cdc FGB|cBF BAA | B2F2 A2B2|c2z3 ebee|B2c2 bonastot eaut Fupteesafs2 sennc
e,>c/d/2|BA cc | c2G2|d ef/f/ | d2cBcB | B4 |]
```

Figure 2: Generated Music

Figure 3: Music from Figure 2 in standard music notation.

(b) Plot your training loss and validation loss vs number of epochs on data. Discuss your findings. *(5 points)*

(c) Try changing the number of neurons in your hidden layer for at least 3 different numbers, for ex. 50, 75 and 150. Now, again plot your training loss and validation loss vs number of epochs on data. What do you observe? Discuss your findings. *(5 points)*

(d) Use dropout with p=.1, .2, and .3, try generating one sample music for each. Also, plot your training loss and validation loss vs number of epochs for each. Does dropout increase or decrease the training speed? Does it improve the results? (This is a qualitative judgment). Discuss your findings. *(5 points)*

(e) Find out how your model performs with different optimization techniques by using Adagrad and RMSProp. Plot training and validation loss vs number of epochs for both. Compare the performance for each. *(5 points)*

(f) **Feature Evaluation** - For one of your generated music samples, do forward propagation through the network and note the activation of each neuron for each of the characters. Plot each of these activations as a heatmap and report the heatmap for at least 1 neuron whose activation pattern you can interpret as signaling some feature of the music. *(5 points)*.

A example of one of these heatmaps is given in Figure 4.It basically shows some generated text from our trained network and shows how a neuron behaves for each of the character in that. In this case, the neuron had low activation for body of the music and high activation for header, which shows that it is able to recognize header of music in ABC format.

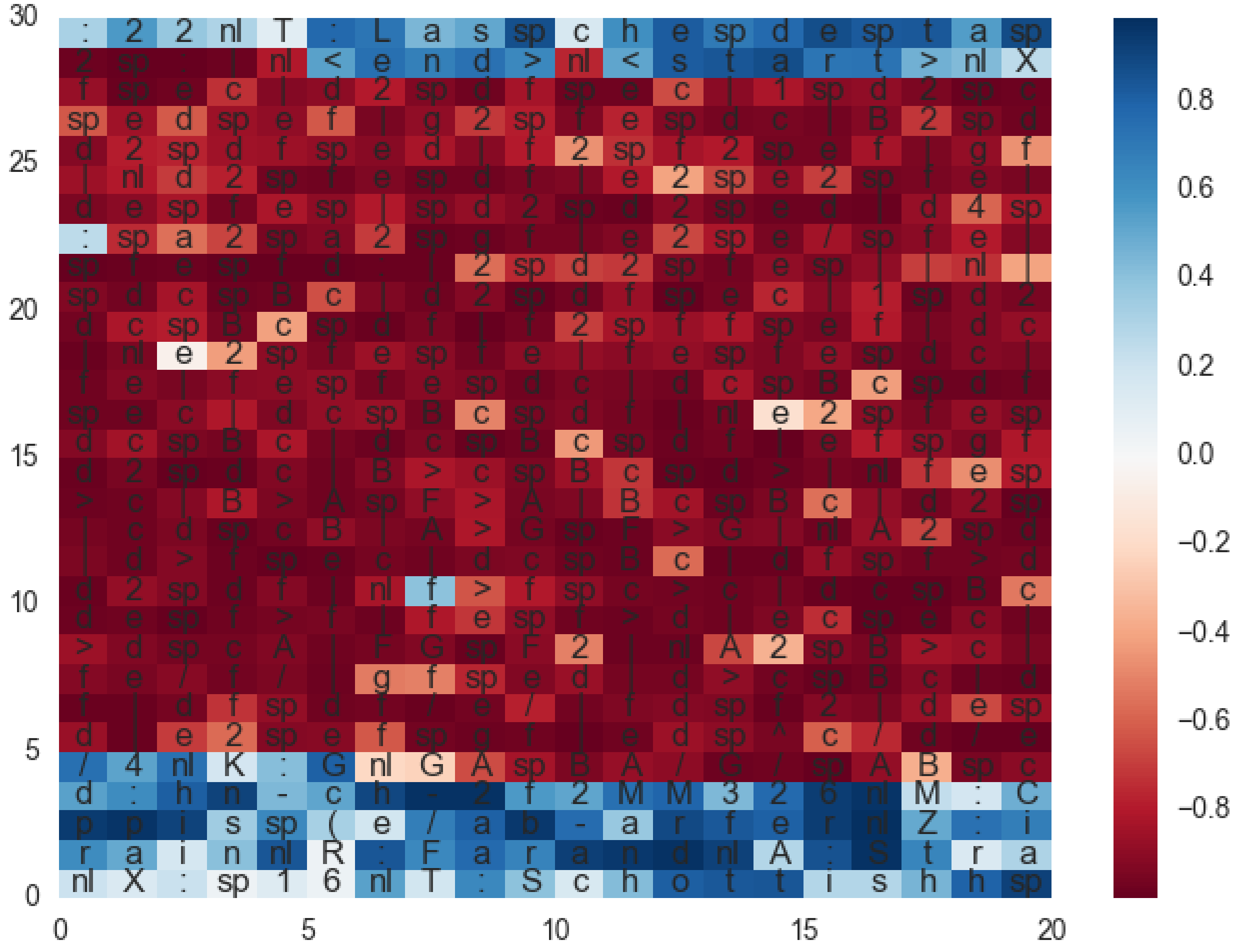

Figure 4: Heatmap showing that this particular neuron fires for the header.

Note- The above heatmap has been generated using an LSTM network that was trained for a whole day. So your heatmap might not be that effective. But a heatmap giving some kind of insight would be good enough for this task.