Abstract

In this assignment, we implemented a char-based generative LSTM model and trained it to “write” songs. We also explore effects of different hyper-parameters to our model. Specifically, we find that temperature of T=XXX reached a best validation loss(same after) of YYY; 1-layer hidden unit of hid\_dim=XXX reached YYY; dropout rate of dr=XXX reached YYY; optimizer of Adam reached 2.141. We also provide a few pieces of sample music. In our opinion, some of them do sound pretty well.

5 Optimizer Exploration

5.1 Plots

In this section, we explore the performance of different optimizers. Specifically, we used the 1-layer LSTM with 100 hidden units, and applied learning rate of 0.001. For optimizers, we compared Adam, Adagrad and RMSProp. The following figures are train/valid set loss against Epoches.







5.2 Discussion

Note that among the three optimizers we tried, Adam turns out to have the best performance on our model, RMSprop holds the second place, while the Adagrad have the lowest performance.

This can understand that RMSProp resolve the Adagrad’s diminishing learning rates problem. Also, RMSprop divides the learning rate by an exponentially decaying average of squared gradients. For Adam, it not only stores the exponentially decaying average of the past squared gradients like RMSprop, but also keeps an exponentially decaying average of past gradient. Thus, in practice Adam usually works well and is better to other adaptive learning-method algorithm.

6 Feature Evaluation

6.1 Heat Plots

One way to understand the working mechanism of the LSTM generated model, is to visualize the hidden unit output with the generated text and plot them together as a heat map. Using the model with Adam optimizer in Section 5, we pick one of the hidden unit output for the heat plot. Also, we generated the text of 900 characters.



Fig. Heatmap for one Hidden Unit Output with Generated Text

6.2 Discussion

From the plot, we can see that for the header of the .abc text, the neuron tends to have lower activation. On contrast, for the body of the .abc text, it tends to have higher activation. One thing we can know about this neuron is that it can distinguish the header from the body. Thus, our whole LSTM network did learn the structure of the .abc text. The other thing we can tell is this neuron may involve more in the generation of the body of the text. Thus, it “cares” more on the tune instead of the style.

Therefore, we can conclude that the activations of neurons reflect the final output of the LSTM model.

7. Contributions

Fanjin Zeng

Implemented the main structure of the code. Generalized the code for later experiments.

Xinyue Ou

Finished the experiments and writing with part (a) and (b).

Yuhan Chen

Finished the experiments and writing with part (c) and (d).

He Qin

Finished the experiments and writing with part (e) and (f).

8 Reference

[1] The Unreasonable Effectiveness of Recurrent Neural Networks, <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

[2] Understanding LSTM Networks, <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

[3] Sequence Models and Long-Short Term Memory Networks, Pytorch, <http://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html>

[4] Generating Names with a Character-Level RNN, Pytorch, <http://pytorch.org/tutorials/intermediate/char_rnn_generation_tutorial.html>

[3] Heatmap with text in each cell with matplotlib's pyplot, <https://stackoverflow.com/questions/25071968/heatmap-with-text-in-each-cell-with-matplotlibs-pyplot>