Regularization Techniques for RNN-Generated Melodies

Nalini Singh

Harvard-MIT Division of Health Sciences and Technology, MIT Computer Science and Artificial Intelligence Laboratory

Objectives

This project aims to characterize and compare three "regularization" strategies for recurrent neural networks used for token sequence generation:

- Dropout Regularization [1]
- Scheduled Sampling [2]
- Professor Forcing [3]

In particular, this project aims to implement each of these four strategies in Tensorflow. Then, this project aims to apply and evaluate these strategies using a MelodyRNN (an RNN for generation of single-note melodies) [4] trained on the Yamaha Piano E-Competition Dataset.

Motivation

- *Overfitting:* Learned model fits training data too well
- Two sources of overfitting in generative RNNs:
  - Neurons adapt *too well* to patterns in input data
  - Training and inference mismatch: in training, token likelihood conditioned on previous *actual* token, but during inference, token likelihood condition on previous *generated* token
- Goal: **Determine which regularization strategies best prevent overfitting** in order to generate better sequences
- One token sequence generation problem of interest: melody composition

Background: Regularization Methods

Dropout Regularization [1]

- Idea: remove units from neural network with probability p at training time; scale weights by p during generation
- Mathematical characterization for hidden unit i in layer l:

$$r_j^l \sim \text{Bernoulli}(p)$$
$$\tilde{\mathbf{y}}^l = \mathbf{y}^l * \mathbf{r}^l$$
$$z_i^{l+1} = \mathbf{w}_i^{l+1}\tilde{\mathbf{y}}^l + b_i^{l+1}$$
$$y_i^{l+1} = f\left(z_i^{l+1}\right)$$

Scheduled Sampling [2]

- Idea: to estimate previous token at iteration i of training time, use true token with probability ϵ_i and use generated token with probility $1 - \epsilon_i$
- Inverse sigmoidal ϵ_i decay schedule:

$$\epsilon_i = \frac{k}{(k + \exp(i/k))}, k > 1$$

Professor Forcing [3]

- [5]: [2] yields biased estimator of previous token
- Idea: train discriminator to differentiate between true and generated samples; train generator to maximize data likelihood and fool generator

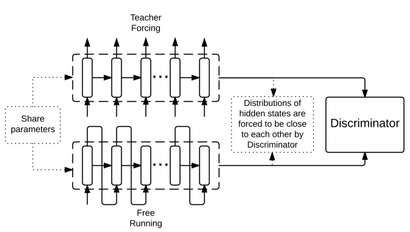
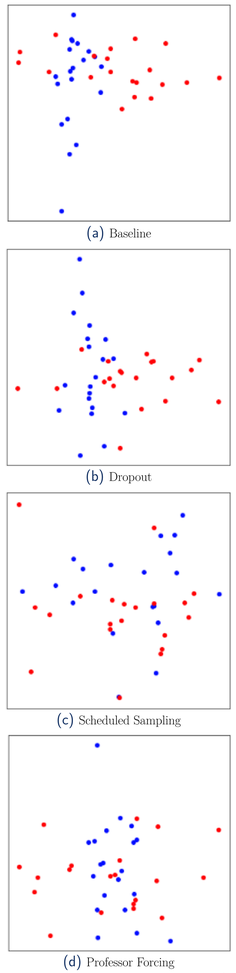


Figure 1: Network architecture used for professor forcing [3]

Experiments

- Model: MelodyRNN [4]
- Dataset: Yamaha Piano E-Competition Dataset
  - 1,400 performances
  - Randomly selected 90:10 training-evaluation split
- Implemented scheduled sampling and professor forcing; dropout implicitly available within MelodyRNN
- Architecture: 2-layers [64,64]
- Batch Size: 32
- Number of Iterations: 5,000
- After training: generated 10 128-step melodies primed at middle C
- Evaluation:
  - Hidden state discrepancy between training and generated sequences
  - Training and evaluation loss
  - Log likelihood

Results: Hidden State Analysis



(a) Baseline

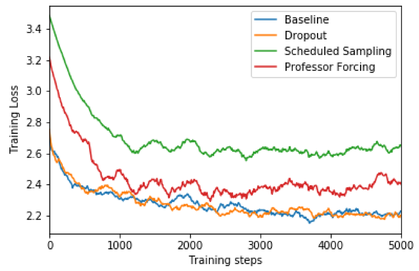
(b) Dropout

(c) Scheduled Sampling

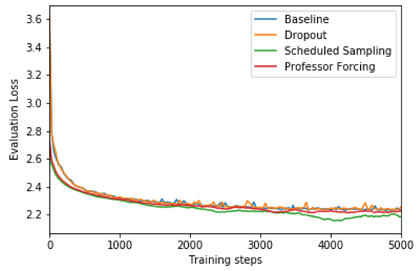
(d) Professor Forcing

Figure 2: T-SNE plots of RNN hidden states during training (red) and generation (blue)

Results: Training and Evaluation Loss



(a) Training loss for different regularization techniques



(b) Evaluation loss for different regularization techniques

Figure 3: Computed training and evaluation loss

Results: Log Likelihood

Strategy	Log Likelihood
Baseline	-280.3
Dropout	-235.5
Scheduled Sampling	-205.4
Professor Forcing	-226.2

Discussion

- Hidden State Analysis:
  - Clear disparity between training and generation states for baseline and dropout
  - Scheduled sampling and professor forcing show closer similarity in hidden states between training and generation
- Training and Evaluation Loss:
  - Minimal difference between baseline and dropout performance
  - Scheduled sampling and professor forcing do *worse* in training, but scheduled sampling performs slightly better in evaluation, indicating a regularizing effect
- Log Likelihood:
  - Best performing strategy: scheduled sampling
- Comparing Strategies:
  - Order of magnitude of training/generation mismatch in RNNs seems larger than that of units directly memorizing training data
  - Scheduled sampling and professor forcing outperform baseline/dropout
  - Bias in estimator shown by [5] is not an issue here (potential confounder: inadequate discriminator training)
  - Overall: scheduled sampling appears to be the best regularization strategy; also, faster runtime than professor forcing

Limitations

- All evaluations performed with the same training network architecture as the originally chosen baseline network
- Results show effects of different regularization techniques when network structure is held constant
- Results do not explain behavior when architecture for each strategy is tuned separately

Future Work

- *Evaluation of SeqGAN [6]*:
  Generative model: RL agent trained on a reward signal from discriminator; discriminator trained to distinguish between real and generated sequences
- *Application to long-term structure models*:
  Extend regularization methods described here to 'lookback' and attention mechanisms used to induce long-term structure in melodies

Conclusion

- Scheduled sampling generated melodies with the highest log likelihood
- Scheduled sampling demonstrated highest similarity between the hidden states observed during training and generation
- **Scheduled sampling is the most effective regularization method for this MelodyRNN architecture**

References

[1] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov.
Dropout: a simple way to prevent neural networks from overfitting.
Journal of machine learning research, 15(1):1929–1958, 2014.

[2] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer.
Scheduled sampling for sequence prediction with recurrent neural networks.
In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.

[3] Alex M Lamb, Anirudh GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio.
Professor forcing: A new algorithm for training recurrent networks.
In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.

[4] Magenta: Music and art generation with machine intelligence.
`https://github.com/tensorflow/magenta`, 2017.

[5] Ferenc Huszár.
How (not) to train your generative model: Scheduled sampling, likelihood, adversary?
arXiv preprint arXiv:1511.05101, 2015.

[6] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu.
Seqgan: Sequence generative adversarial nets with policy gradient.
In *AAAI*, pages 2852–2858, 2017.

Acknowledgements

Many thanks to the 9.520 course staff for their instruction over the course of this semester. Special thanks to the Georgios Evangelopolous for his guidance in project selection.

Additional Information

- Code: www.github.com/nalinimsingh/9.520-MelodyRnn-Regularization
- Contact Email: nmsingh@mit.edu