

Recurrent Generative Stochastic Networks for Sequence Prediction

Abstract

We present a new generative model for unsupervised learning of sequence representations, the recurrent Generative Stochastic Network (RNN-GSN). This approach builds on recent deep recurrent neural network architectures for modeling high-dimensional, multimodal input distributions, and is able to predict complex sequences of these distributions. The RNN-GSN is non-probabilistic and easier to perform inference over compared to other representation models, such as the RNN-RBM or NADE variants. We provide experimental validation for this new model on sequences of MNIST images and standard MIDI music datasets.

1. Introduction

Unsupervised sequence learning is an important problem in machine learning given that most information (ranging from speech to video to music to even consumer behavior) is often unlabeled and has a sequential structure. Most of these sequences consist of high-dimensional, complex objects such as words in text, images in video, or chords in music. Recently, recurrent neural networks (RNN) (Rumelhart et al., 1986) have become state-of-the-art for sequence representation because they have an internal memory that can learn long-term, temporal dependencies. Further advances with Hessian-free optimization and Long Short-term Memory for RNNs (Martens & Sutskever, 2011; Hochreiter & Schmidhuber, 1996) have enabled learning highly complex temporal dependencies.

While traditional RNNs can learn complex sequences through maintaining an internal memory, they don't deal well when modeling complex input distributions, where the conditional distribution at each time step is highly multimodal. In real-world data, we often care more about this conditional distribution rather than the expected value. In music, for example, the existence of a particular note can greatly change the probabilities with which other notes occur at the same time in a chord. In consumer behavior, the

existence of a particular action taken can greatly change the probabilities of other actions made within the same time window. We would prefer to reason and perform inference over these distributions at each time step to form a better representation of the sequences.

Deep RNN architectures have been proposed to alleviate the traditional RNN architecture shortcomings with complex input distributions (Pascanu et al., 2013). One proposed framework involves using deep input-to-hidden or hidden-to-output functions to reduce input complexity, making the RNN's temporal modeling task easier. These functions exploit the ability of deep networks to disentangle the underlying factors of variation of the original input and flatten high-density data manifolds (Goodfellow et al., 2009; Glorot et al., 2011; Bengio et al., 2012).

One such model, the RNN-RBM, replaces the output layer of an RNN with a restricted Boltzmann machine (RBM) to provide a conditional generative model over the input distribution (Boulanger-Lewandowski et al., 2012; Boulanger-Lewandowski, 2014). The RNN-RBM is a generalization of previous recurrent temporal RBM work (Sutskever et al., 2009) that has shown promise for modeling complex sequences such as MIDI representations of polyphonic music. Another possible model is to replace the RNN output layer with a neural autoregressive distribution estimator (NADE), which has been shown to provide a tractable distribution estimator that performs similarly to large, intractable RBMs (Larochelle & Murray, 2011).

Motivated by the promise of deep RNN models for learning sequence representations, we propose a model using a Generative Stochastic Network (GSN) as the hidden-to-output function of an RNN, called the RNN-GSN. GSNs are a recent generalization of denoising autoencoders (Bengio et al., 2013a) that are easier to perform inference over compared to RBMs and sample from compared to NADEs. GSNs are non-probabilistic, generative models that have also been shown to learn the same model as a deep orderless NADE, which alleviates the factorization ordering drawback of a traditional NADE model (Yao et al., 2014).

The rest of this paper explains the RNN-GSN model and provides experimental evidence for its use in modeling complex sequences. In Sections 2, 3, and 4, we introduce the GSN, RNN, and RNN-GSN architectures in more detail. We then provide experimental validation of the RNN-

GSN on sequences of MNIST images and standard MIDI datasets in Section 5.

2. Generative Stochastic Networks

Generative stochastic networks (GSN) are a generalization of the denoising auto-encoder and help solve the problem of mixing between many major modes of the input data distribution.

2.1. Denoising auto-encoder

Denoising auto-encoders use a Markov chain to learn a reconstruction distribution $P(X|\tilde{X})$ given a corruption process $C(\tilde{X}|X)$ for some data X . Denoising auto-encoders have been shown as generative models (Bengio et al., 2013b), where the Markov chain can be iteratively sampled from:

$$X_t \sim P_{\Theta}(X|\tilde{X}_{t-1}) \quad (1)$$

$$\tilde{X}_t \sim C(\tilde{X}|X_t) \quad (2)$$

As long as the learned distribution $P_{\Theta_n}(X|\tilde{X})$ is a consistent estimator of the true conditional distribution $P(X|\tilde{X})$ and the Markov chain is ergodic, then as $n \rightarrow \infty$, the asymptotic distribution $\pi_n(X)$ of the generated samples from the denoising auto-encoder converges to the data-generating distribution $P(X)$ (Bengio et al., 2013b).

2.2. Easing restrictive conditions on the denoising auto-encoder

A few restrictive conditions are necessary to guarantee ergodicity of the Markov chain - requiring $C(\tilde{X}|X) > 0$ everywhere that $P(X) > 0$. Particularly, a large region V containing any possible X is defined such that the probability of moving between any two points in a single jump $C(\tilde{X}|X)$ must be greater than 0. This restriction requires that $P_{\Theta_n}(X|\tilde{X})$ has the ability to model every mode of $P(X)$, which is a problem this model was meant to avoid.

To ease this restriction, Bengio et al. (Bengio et al., 2013a) prove that using a $C(\tilde{X}|X)$ that only makes small jumps allows $P_{\Theta}(X|\tilde{X})$ to model a small part of the space V around each \tilde{X} . This weaker condition means that modeling the reconstruction distribution $P(X|\tilde{X})$ would be easier since it would probably have fewer modes.

However, the jump size σ between points must still be large enough to guarantee that one can jump often enough between the major modes of $P(X)$ to overcome the regions of low probability: σ must be larger than half the largest distance of low probability between two nearby modes, such that V has at least a single connected component between modes. This presents a tradeoff between the difficulty of learning $P_{\Theta}(X|\tilde{X})$ and the ease of mixing between modes

separated by this low probability region.

2.3. Generalizing to GSN

While denoising auto-encoders can rely on X_t alone through a deterministic procedure for the state of the Markov chain, GSNs introduce a latent variable H_t that acts as an additional state variable in the Markov chain along with the visible X_t (Bengio et al., 2013a):

$$H_{t+1} \sim P_{\Theta_1}(H|H_t, X_t) \quad (3)$$

$$X_{t+1} \sim P_{\Theta_2}(X|H_{t+1}) \quad (4)$$

The latent state variable H can be equivalently defined as $H_{t+1} = f_{\Theta_1}(X_t, Z_t, H_t)$, a learned function f with an independent noise source Z_t such that X_t cannot be reconstructed exactly from H_{t+1} . If X_t could be recovered from H_{t+1} , the reconstruction distribution would simply converge to the Dirac at X . Denoising auto-encoders are therefore a special case of GSNs, where f is fixed instead of learned.

GSNs also use the notion of walkbacks to aid training. Walkbacks are the process of generating samples by iteratively sampling from $P_{\Theta_1}(H|H_t, X_t)$ and $P_{\Theta_2}(X|H_{t+1})$ for a given input. By using walkbacks, the model is more likely to seek out spurious modes in the data distribution and correct for them (Bengio et al., 2013b). The resulting Markov chain of a GSN is inspired by Gibbs sampling, but with stochastic units at each layer that can be backpropagated (Rezende et al., 2014).

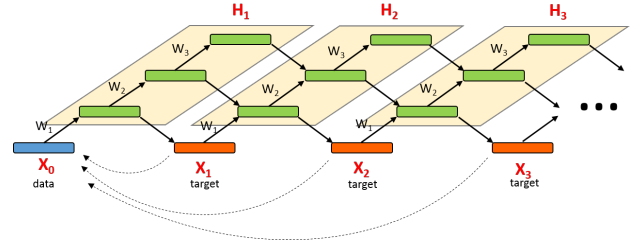


Figure 1. Unrolled GSN Markov chain.

Experimental results with GSNs show that their latent states mix well between the major modes of the data - mixing faster at higher layers in the model (Bengio et al., 2013a). Using this property, we tested a simple temporal GSN model to predict sequences of inputs. The temporal GSN uses a linear transformation $H \rightarrow H$ to predict $P(H_t|H_{t-1}, \dots, H_{t-n})$ with an input history of size n . Using this predicted H_t , an expected input x_t was sampled from the model's learned reconstruction distribution $P_{\Theta_2}(X|H)$.

Qualitatively, this model appears to predict temporal dependencies well within its history window n (Figure 3).

This result provided motivation to use the latent state H as the emitted parameter in a better-suited temporal model, such as an RNN.

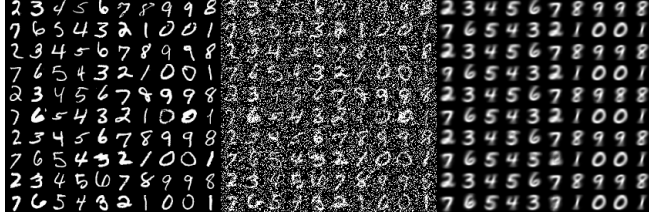


Figure 2. Temporal GSN with history $n = 2$ trained on an arbitrary MNIST sequence. Original input sequence is on the left, a noisy version of the input fed into the model is in the middle, and the predicted input based on the history is on the right.

3. Recurrent Neural Networks

Traditional RNNs simulate a discrete-time system that has an input sequence $\{x_1 \dots x_t\}$ and a hidden state sequence $\{h_1 \dots h_t\}$ that map to an output sequence $\{y_1 \dots y_t\}$. The network is defined for timestep t in the sequence by the hidden and output equations:

$$h_t = \Phi_h(W^T h_{t-1} + U^T x_t + b_h) \quad (5)$$

$$y_t = \Phi_y(V^T h_t + b_y) \quad (6)$$

where Φ_h and Φ_y are element-wise nonlinear functions and W , U , V , b_h , and b_y are the network parameters Θ . The hidden state h contains all information about the sequence for a given timestep t .

RNNs can be trained via backpropagation over a set number of time steps, which is known as backpropagation through time. However, due to the exploding gradient problem, backpropagation through time has difficulty training the parameters for long-term temporal dependencies. To fix the exploding gradient problem, long short-term memory units (LSTM) (Hochreiter & Schmidhuber, 1996) can be introduced as the hidden units that contain extra parameters to learn a gated memory for storing the unit's activation value for a period of time steps. With this architecture, normal backpropagation through time can train the model to learn much longer temporal dependencies. Alternatively, Hessian-free optimization (Martens & Sutskever, 2011) can learn the model parameters for long-term temporal dependencies without needing to introduce specialized hidden units.

3.1. Extension to deep RNN

While traditional RNNs can be seen as deep networks over multiple timesteps due to the hidden connections $h_{t-1} \rightarrow h_t$, they suffer from being shallow, where they only containing a single nonlinear layer, at each individual timestep,

which limits the complexity of inputs they can realistically represent. When analyzing RNNs in a single timestep, there are four areas where depth can be added: input-to-hidden function $x_t \rightarrow h_t$, hidden-to-output function $h_t \rightarrow y_t$, hidden-to-hidden function $h_{t-1} \rightarrow h_t$, and stacking hidden layers h_t (Pascanu et al., 2013).

4. The RNN-GSN

The recurrent GSN (RNN-GSN)

$$\hat{H}_t = \Phi_h(A^T U_t + b_h) \quad (7)$$

$$\hat{X}_t \sim GSN(X, \hat{H}) \quad (8)$$

$$U_{t+1} = \Phi_u(B^T X_t + C^T U_t + b_u) \quad (9)$$

where Φ_h and Φ_u are element-wise nonlinear functions, A , B , C , b_h , and b_u are the recurrent network parameters, and $GSN(X, \hat{H})$ is a GSN from Section 2 trained with parameters Θ_1 and Θ_2 .

This architecture retains separation between the recurrent part and the deep GSN part.

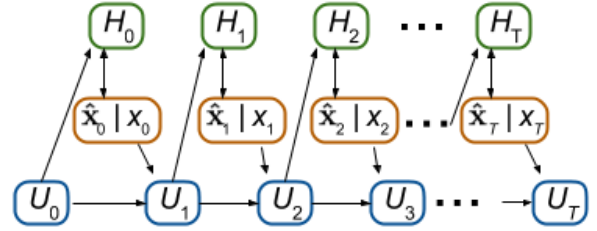


Figure 3. RNN-GSN architecture. Input distribution is X , GSN hidden layers are H , and recurrent hidden units are U .

$$\hat{H}_t = \Phi_h(A^T U_t + b_h) \quad (10)$$

$$\hat{X}_t \sim GSN(X, H) \quad (11)$$

$$H_t \sim GSN(X, H) \quad (12)$$

$$U_{t+1} = \Phi_u(B^T H_t + C^T U_t + b_u) \quad (13)$$

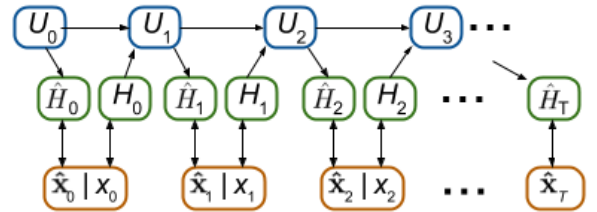


Figure 4. Extension of the RNN-GSN to use both deep input-to-hidden and hidden-to-output functions with a GSN.

4.1. Initialization strategies

5. Experiments

This is the experiments section. Each RNN-GSN was first initialized with GSN parameters and trained with noise scheduling, which has been shown to help the network learn appropriate features during stochastic gradient descent (Geras & Sutton, 2014).

5.1. Sequences of MNIST digits

arbitrary sequences of images.

- Sequence1 is a simple linear sequence of digits 0-9 repeating.
- Sequence2 introduces one bit of parity by alternating sequences 0-9 and 9-0 repeating, where the next value depends on whether the sequence is ascending or descending.
- Sequence3 creates a longer, more complex sequence by introducing a second bit of parity. It is formed by:

Log-likelihoods are estimated by a Parzen density estimator, which is biased. Further validation can be seen qualitatively by the predicted samples produced from the model.

5.2. Sequences of polyphonic music

midi stuff.

- Piano-midi.de
- Nottingham
- MuseData
- JSB chorales

Log-likelihoods estimated by the Parzen density estimator are biased and cannot be compared to the AIS estimation used by Boulanger-Lewandowski et al. However, accuracies as computed by Bay et al. are provided for comparison (Bay et al., 2009).

6. Conclusion

We presented the RNN-GSN, a generative, non-probabilistic model for learning deep sequence representations. We validated its efficacy as a deep recurrent model on complex inputs with sequences of MNIST images and MIDI representations of polyphonic music. The RNN-GSN works as well as other deep recurrent models, such as the RNN-RBM, but is an easier framework

to perform inference over and sample from due to its non-probabilistic nature.

Further improvements and areas of study could be using LSTM units in the RNN or Hessian-free optimization, and combining depth at multiple stages of the RNN. We would like to look into using GSNs as both the input-to-hidden and hidden-to-output functions of the RNN, further reducing the complexity for representing the sequence. Finally, multimodal GSNs could provide benefits when the hidden representations of the input data distribution cannot be assumed as unimodal (Ozair et al., 2013).

References

- Bay, Mert, Ehmann, Andreas F, and Downie, J Stephen. Evaluation of multiple-f0 estimation and tracking systems. In *10th International Society for Music Information Retrieval Conference*, 2009.
- Bengio, Yoshua, Mesnil, Grégoire, Dauphin, Yann, and Rifai, Salah. Better mixing via deep representations. *CoRR*, abs/1207.4404, 2012.
- Bengio, Yoshua, Thibodeau-Laufer, Eric, and Yosinski, Jason. Deep generative stochastic networks trainable by backprop. *CoRR*, abs/1306.1091, 2013a.
- Bengio, Yoshua, Yao, Li, Alain, Guillaume, and Vincent, Pascal. Generalized denoising auto-encoders as generative models. *CoRR*, abs/1305.6663, 2013b.
- Boulanger-Lewandowski, Nicolas. *Modeling High-Dimensional Audio Sequences with Recurrent Neural Networks*. PhD thesis, Université de Montréal, april 2014.
- Boulanger-Lewandowski, Nicolas, Bengio, Yoshua, and Vincent, Pascal. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.
- Geras, Krzysztof J. and Sutton, Charles A. Scheduled denoising autoencoders. *CoRR*, abs/1406.3269, 2014.
- Glorot, Xavier, Bordes, Antoine, and Bengio, Yoshua. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *In Proceedings of the Twenty-eight International Conference on Machine Learning, ICML*, 2011.
- Goodfellow, Ian, Lee, Honglak, Le, Quoc V., Saxe, Andrew, and Ng, Andrew Y. Measuring invariances in deep networks. In Bengio, Y., Schuurmans, D., Lafferty, J.D., Williams, C.K.I., and Culotta, A. (eds.), *Advances*

440	in <i>Neural Information Processing Systems 22</i> , pp. 646–	495
441	654. Curran Associates, Inc., 2009.	496
442		497
443	Hochreiter, Sepp and Schmidhuber, Jürgen. Bridging long	498
444	time lags by weight guessing and "long short term mem-	499
445	ory". In <i>Spatiotemporal models in biological and artifi-</i>	500
446	<i>cial systems</i> , pp. 65–72. IOS Press, 1996.	501
447		502
448	Larochelle, Hugo and Murray, Iain. The neural autoregres-	503
449	sive distribution estimator. <i>Journal of Machine Learning</i>	504
450	<i>Research</i> , 15:29–37, 2011.	505
451	Martens, James and Sutskever, Ilya. Learning recurrent	506
452	neural networks with hessian-free optimization. In <i>Pro-</i>	507
453	<i>ceedings of the 28th International Conference on Ma-</i>	508
454	<i>chine Learning (ICML-11)</i> , pp. 1033–1040, 2011.	509
455		510
456	Ozair, Sherjil, Yao, Li, and Bengio, Yoshua. Multimodal	511
457	transitions for generative stochastic networks. <i>CoRR</i> ,	512
458	abs/1312.5578, 2013.	513
459		514
460	Pascanu, Razvan, Gülçehre, Çağlar, Cho, Kyunghyun, and	515
461	Bengio, Yoshua. How to construct deep recurrent neural	516
462	networks. <i>CoRR</i> , abs/1312.6026, 2013.	517
463	Rezende, Danilo J., Mohamed, Shakir, and Wierstra, Daan.	518
464	Stochastic backpropagation and approximate inference	519
465	in deep generative models. In <i>Proceedings of the 31st</i>	520
466	<i>International Conference on Machine Learning</i> , 2014.	521
467		522
468	Rumelhart, David E., Hinton, Geoffrey E., and Williams,	523
469	Ronald J. Learning representations by back-propagating	524
470	errors. In <i>Parallel Dist. Proc.</i> , pp. 318–362. MIT Press,	525
471	1986.	526
472		527
473	Sutskever, Ilya, Hinton, Geoffrey E., and Taylor, Gra-	528
474	ham W. The recurrent temporal restricted boltzmann	529
475	machine. In Koller, D., Schuurmans, D., Bengio, Y., and	530
476	Bottou, L. (eds.), <i>Advances in Neural Information Pro-</i>	531
477	<i>cessing Systems 21</i> , pp. 1601–1608. Curran Associates,	532
478	Inc., 2009.	533
479	Yao, Li, Ozair, Sherjil, Cho, Kyunghyun, and Bengio,	534
480	Yoshua. On the equivalence between deep nade and	535
481	generative stochastic networks. <i>CoRR</i> , abs/1409.0585,	536
482	2014.	537
483		538
484		539
485		540
486		541
487		542
488		543
489		544
490		545
491		546
492		547
493		548
494		549