A Tractable Model for Generating High-Dimensional Sequences

Anonymous Author(s)
Affiliation
Address
email

Abstract

We propose a tractable probabilistic model for generating complex high dimensional sequences. This model combines a powerful distribution estimator, the Real-Valued Neural Autoregressive Distribution Estimator (RNADE) with a Recurrent Neural Network (RNN) for capturing temporal dependencies in high-dimensional data. Maximum likelihood learning can be applied to efficiently train the model using a gradient based optimiser. Unlike other models in this family, log-likelihoods for sequences can be computed exactly and efficiently. We evaluate the model's performance on two standard datasets and compare its performance to other models.

1 Introduction

Modeling sequences is a fundamental problem in machine learning. Many popular models like Hidden Markov Models (HMMs) and Linear Dynamical Systems use latent variables to capture temporal properties of sequential model. However HMMs employ a 1-of-K state representation for the hidden state and trying to model larger histories can lead to exponentially large hidden states. RNNs are an interesting alternative for modeling sequential data and have recently been employed in speech recognition, MIR and various other tasks. RNNs employ a deterministic distributed hidden representation that can in theory capture temporal properties over very long time scales. However RNNs are limited by the effectiveness of gradient based optimisers and the size of the output space that they can model. To rectify this issue, a family of models that condition complex distribution estimators on the hidden state of an RNN have been proposed. The first such model was the RTRBM and it was later extended by the RNN-RBM and the RNN-NADE [2].

The main idea behind this family of models is to use complex distribution estimators like the RBM and the NADE and incorporate a temporal element by conditioning the parameters of the model at each time step on previously observed parts of the sequence. As mentioned before an RNN can in theory learn dependencies over large time-scales within the data and is used in these models for conditioning the distributions at each step. RNN based models are also attractive because the recurrent parts of the model can be easily trained by using BPTT or Hessian Free (HF) optimisation. The RTRBM and the RNN-NADE models cannot be trained exactly by maximum likelihood learning because the gradients of the RBMs cannot be computed exactly. This problem was rectified by using a NADE instead of an RBM. However, this family of temporal models is not very well suited to modeling real-valued data. Both the RBM and the NADE estimate distributions over binary spaces. Although the RBM can be used on real data by either scaling the data between $[0, 1]$ or by using the Gaussian-Bernouilli RBM (gRBM), we feel that using an estimator meant for real data could provide further gains.

In this paper, we propose a model called the RNN-RNADE that combines the temporal modeling properties of the RNN with the recently proposed RNADE model. The RNADE is a generalisation of the NADE to real-valued data. It has been shown to outperform gaussian mixture models (GMMS) and many other distribution estimators on a variety of datasets of varying dimensionality and complexity. In this paper we explore how the RNADE can be combined with the RNN to yield a temporal generative model. We show how the model can be trained efficient using maximum likelihood learning using gradient based optimisers. Finally we evaluate the performance

2 The RNADE

The RNADE is a generalisation of the NADE to real-valued data. Like the NADE, the RNADE expresses the joint probability of the data, as a product of one-dimensional conditional distributions as follows:

$$p(x) = \prod_{d=1}^{D} p(x_d|\mathbf{x}_{<\mathbf{d}}) \text{ with } p(x_d|\mathbf{x}_{<\mathbf{d}}) = p_{\mathcal{M}(x_d|\theta_d)}$$

where $p_{\mathcal{M}}$ is a mixture of Gaussians and $\boldsymbol{x}_{<d}$ is a vector of all the dimensions of the data point $< d$. The RNADE models each of the conditional distributions by means of a feed-forward neural network with tied weights, with one neural network for each dimension. The activations of the hidden units are calculated as follows:

$$\mathbf{a}_d = \boldsymbol{W}_{.,<d}\boldsymbol{x}_d + \mathbf{c}$$
$$\boldsymbol{h}_d = \sigma(\rho_d \mathbf{a}_d)$$

where $\mathbf{c} \in \mathbb{R}^H$ and $\boldsymbol{W} \in \mathbb{R}^{D\times(H-1)}$ are neural network parameters that are shared across all the neural networks and $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid function. $\boldsymbol{W}_{.,<d}$ represents the first $D-1$ columns of the shared weight matrix. The term ρ_d is a scaling factor which is also learnt from the data. The scaling factor was introduced in [1] in order to prevent the sigmoid hidden units from saturating. The computation of the activations of the hidden units can be made more efficient by performing the computation as:

$$\mathbf{a}_1 = \mathbf{c}, \quad \mathbf{a}_{d+1} = \mathbf{a}_d + x_d \mathbf{W}_{.,d}$$

Unlike the NADE which models each output as a bernouilli distribution, the outputs of each of the feed forward neural networks of the RNADE are mixtures of Gaussians. Therefore the RNADE comprises of D mixture density networks with tied input-to-hidden weights. Once the hidden units of the RNADE have been computed, they are used to compute the parameters of the GMMs $\boldsymbol{\theta}_d = \{\boldsymbol{\alpha}_d, \boldsymbol{\mu}_d, \boldsymbol{\sigma}_d\}$ at each output, where $\boldsymbol{\alpha}_d$ are the mixing coefficients, $\boldsymbol{\mu}_d$ are the means and $\boldsymbol{\sigma}_d$ are the variances. These parameters are computed as follows:

$$\boldsymbol{\alpha}_d = \text{softmax}({\mathbf{V}_d^{\alpha}}^T \mathbf{h}_d + \mathbf{b}_d^{\alpha})$$
$$\boldsymbol{\mu}_d = {\mathbf{V}_d^{\mu}}^T \mathbf{h}_d + \mathbf{b}_d^{\mu}$$
$$\boldsymbol{\sigma}_d = \exp({\mathbf{V}_d^{\sigma}}^T \mathbf{h}_d + \mathbf{b}_d^{\sigma})$$

where $\mathbf{V}_d^{\alpha}, \mathbf{V}_d^{\mu}, \mathbf{V}_d^{\sigma}$ are $H \times K$ matrices , $\mathbf{b}_d^{\alpha}, \mathbf{b}_d^{\mu}, \mathbf{b}_d^{\sigma}$ are vectors of size K and K is the number of components in the GMM. The parameters of the RNADE can be learnt by performing gradient ascent on the log-likelihood of the training set.

3 RNN-based Models for Sequences

An RNN can define a distribution over sequences $\mathbf{x}_1^T$ if the loss function is of the form $L = \sum_t -\log p(\mathbf{x}^{t+1}; \boldsymbol{\theta}^t)$ where $p(.;\boldsymbol{\theta})$ is a distribution with parameters $\boldsymbol{\theta}$, and $\boldsymbol{\theta}^t \equiv f(\mathbf{x}_1^t)$. In more detail, according to figure 1, let the hidden state of the RNN at time t be give by:

$$\mathbf{h}_{RNN}^t = \sigma(\mathbf{W}_{in}\mathbf{x}^t + \mathbf{W}_{rec}\mathbf{h}_{RNN}^{t-1} + \mathbf{b}_{RNN})$$

Let the hidden state of the RNN at time $t-1$ be used to output $\mathbf{z}^t = \sigma(\mathbf{W}_{out}\mathbf{h}_{RNN}^{t-1} + \mathbf{b}_{out})$ and let the parameters of the distribution at time t, $\boldsymbol{\theta}^t = f(\mathbf{z}^t, \boldsymbol{\theta}_{model})$ where $\boldsymbol{\theta}_{model}$ are the independent parameters of the distribution estimator and f is some differentiable function with respect to the model parameters. If we can obtain the derivates of the cost function with respect to the parameters of the distribution $p(;|\boldsymbol{\theta})$, then it can be shown that the parameters of the RNN can be trained by using the chain rule and Back Propagation Through Time (BPTT). Therefore:

$$\frac{\partial L}{\partial \theta^t} = -\frac{\partial p(\mathbf{x}^t)}{\partial \theta^t}$$
$$\frac{\partial L}{\partial \theta_{model}} = \sum_t -\frac{\partial p(\mathbf{x}^t)}{\partial \theta_{model}}$$

$$\frac{\partial L}{\partial \mathbf{z}^t} = \sum_T \sum_i -\frac{\partial p(\mathbf{x}^t)}{\partial \theta^i} \frac{\partial f(\mathbf{z}^t, \boldsymbol{\theta}_{model})^i}{\partial \mathbf{z}^t}$$

Once we obtain all the errors with respect to the outputs of the RNN, the gradients with respect to the RNN parameters can be easily found by applying the chain rule further and backpropagating the gradients back in time. We show the calculations for the proposed model in section 3. Further derivations of the results can be found in the supplementary material.

The RTRBM [4] was the first model that uses the above idea to condition RBMs on the outputs of an RNN. However, the model is constrained by the fact that the outputs of RNN are responsible for both the density estimation and for propagating temporal information to the following time step. The RNN-RBM [2] generalises the RTRBM model by introducing a separate hidden layer for the RNN. Decoupling the states of the RNN and conditional RBMs led to an improvement in performance.

The RNN-RBM model cannot be trained exactly because the error derivatives with respect to the parameters of the conditional RBMs are approximated using Contrastive Divergence (CD). Another drawback of the RNN-RBM model is that obtaining the log-likelihood of the sequence is intractable. This is because the RBM defines a joint distribution over the observed and hidden variables. Obtaining probabilities over the observed variables implies a summation over an exponential number of hidden states. These shortcomings can be addressed by replacing the RBM with the NADE, which is a tractable distribution estimator with performance comparable to RBMs. The log-likelihood can be easily calculated and more powerful gradient based optimisers like HF [3] can be applied because the gradients are exactly calculable.

References

[1] Yoshua Bengio. Discussion of ”the neural autoregressive distribution estimator”. In Geoffrey J. Gordon and David B. Dunson, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, pages 38–39. Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011.

[2] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012.

[3] James Martens and Ilya Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1033–1040, 2011.

[4] Ilya Sutskever, Geoffrey E Hinton, and Graham W Taylor. The recurrent temporal restricted boltzmann machine. In *Advances in Neural Information Processing Systems*, pages 1601–1608, 2008.