Learning Stochastic Recurrent Networks

Justin Bayer
Lehrstuhl für Echtzeitsysteme und Robotik
Fakultät für Informatik
Technische Universität München
bayer.justin@googlemail.com

Christian Osendorfer
Lehrstuhl für Echtzeitsysteme und Robotik
Fakultät für Informatik
Technische Universität München
osendorf@in.tum.de

Abstract

Leveraging advances in variational inference, we propose to enhance recurrent neural networks with latent variables, resulting in Stochastic Recurrent Networks (STORNs). The model i) can be trained with stochastic gradient methods, ii) allows structured and multi-modal conditionals at each time step, iii) features a reliable estimator of the marginal likelihood and iv) is a generalisation of deterministic recurrent neural networks. We evaluate the method on four polyphonic musical data sets and motion capture data.

1 Introduction

Recurrent Neural Networks (RNNs) are flexible and powerful tools for modeling sequences. While only bearing marginal existence in the 1990’s, recent successes in real world applications [1, 2, 3, 4, 5] have resurged interest. This is partially due to architectural enhancements [6], new optimisation findings [7, 8, 9] and the increased computational power available to researchers. RNNs can be employed for a wide range of tasks as they inherit their flexibility from plain neural networks. This includes universal approximation capabilities, since RNNs are capable of approximating any measurable sequence to sequence mapping and have been shown to be Turing complete [10, 11].

One typical application is to let an RNN model a probability distribution over sequences, i.e. \( p(x_{1:T}) \). This is done by writing the distribution in cascade form,

\[
p(x_{1:T}) = \prod_{t=1}^{T} p(x_t|x_{1:t-1}),
\]

where \( x_{1:0} = \emptyset \). Each \( p(x_t|x_{1:t-1}) \) is then represented by the output at a single time step of an RNN, identifying each of its components with the statistics of the distribution. A simple example is that of a Bernoulli, i.e.

\[
p(x_{t,k} = 1|x_{1:t-1}) = \eta_k(x_{1:t-1}) \tag{1}
\]

where \( x_{t,k} \) corresponds to the \( k \)’th component of the \( t \)’th time step of \( x \) with \( k = 1, \ldots, \omega \) and \( t = 1, \ldots, T \). Each \( \eta_k(x_{1:t-1}) \) is the \( k \)’th output of some RNN at time step \( t-1 \), constrained to lie in the interval \((0,1)\). Learning such an RNN then boils down to minimising the negative log-likelihood of the data with respect to the parameters of the network.

This framework gives practitioners a powerful tool to model rich probability distributions over sequences. A common simplification is a naive Bayes assumption that the individual components factorise:

\[
p(x_t|x_{1:t-1}) = \prod_k p(x_{t,k}|x_{1:t-1}).
\]

While sufficient for many applications, reintroduction of dependency among the components of \( x_t \) leaves room for improvement. This is especially true for sequences over spaces which are high
dimensional and tightly coupled. The approach taken by [1] is to use a mixture distribution for 
\[ p(x_t|x_{1:t-1}) \]. Arguably powerful enough to model any dependency between the components of 
\[ x_t \], a drawback is that the number of parameters scales at least linearly with the number of chosen 
mixture components.

Models based on restricted Boltzmann machines and variations [12, 13, 14] provide a solution to 
this as well, yet come with tighter restrictions on the assumptions that can be made. E.g. RBMs are 
restricted to model data from the exponential family [15].

In this work, we propose to consider adding latent variables similar to [16] to the network. Using 
stochastic gradient variational Bayes (SGVB) [17, 18] as an estimator, we train RNNs to model high 
dimensional sequences.

2 Preliminaries

In this section we will recap the basis of our method. We will first describe the used model fam-
ily, that of recurrent neural networks and then the estimator, stochastic gradient variational Bayes 
(SGVB).

2.1 Recurrent Neural Networks

Given an input sequence \( x = (x_1, \ldots, x_T), x_t \in \mathbb{R}^n \) we compute the output sequence of a simple 
Recurrent Neural Network (sRNN) \( y = (y_1, \ldots, y_T), y_t \in \mathbb{R}^n \) via an intermediary hidden state layer 
\( h = (h_1, \ldots, h_T), h_t \in \mathbb{R}^n \) by recursive evaluation of the following equations:

\[
\begin{align*}
    h_t &= f_h(x_t W_{in} + h_{t-1} W_{rec} + b_{hidden}), \\
    y_t &= f_y(h_t W_{out} + b_{out}).
\end{align*}
\]

The set of adaptable parameters is given by \( \theta = \{ W_{in}, W_{rec}, W_{out}, b_{hidden}, b_{out} \} \). \( f_h \) and \( f_y \) are 
transfer functions introducing nonlinearity into the computation.

Adaptation of the network’s behaviour can be done by optimising a loss function with respect to 
the network’s parameters with gradient-based schemes. Consider a data set of finite size, i.e. \( D = 
\{(x_{1:t})\}_{t=1}^T \) on which the loss operates. In a setting as in Equation [1], a reasonable choice is the 
negative log-likelihood given by \( \mathcal{L}_{NLL}(\theta) = - \sum_{t=1}^T \sum_{i=1}^T \log p(x_t|x_{1:t-1}) \).

2.2 Stochastic Gradient Variational Bayes

SGVB was introduced independently by [17] and [18]. For this paper, we will review the method 
briefly in order to introduce notation. We are interested in modelling the data distribution \( p(x) \) with 
the help of unobserved latent variable \( z \) represented as a directed graphical model, i.e. \( p(x) = 
\int p(x|z)p(z)dz \). The integral is in general intractable, which is why we will use a variational upper 
bound on the negative log-likelihood for learning.

\[
- \log p(x) = - \log \int p(x|z)p(z)dz = - \log \int \frac{q(z|x)}{q(z|x)}p(x|z)p(z)dz \leq K L(q(z|x)||p(z)) - E_{z \sim q}[\log p(x|z)] =: \mathcal{L}.
\]

where \( K L(q||p) \) denotes the Kullback-Leibler divergence between two distributions \( q \) and \( p \). In this 
case, we call \( q \) the recognition model since it allows for fast approximate inference of the latent 
variables \( z \) given the observed variables \( x \). Note that \( q \) is a variational approximation of \( p(x|z) \), 
which is the inverse of the generating model\(^1\) \( p(x|z) \) that cannot be found in general.

Both the recognition and the generating model can be chosen arbitrarily in their computational form 
with the possibility to represent probability distributions as outputs and stochastic training being

\(^1\)We use the non standard term “generating model” for \( p(x|z) \) to distinguish it more clearly from the generative model \( p(x) \).
In this work, we restrict ourselves to a standard Normal prior $p(z|\theta^\rho)$ and $q(z|x, \theta^\rho)$ to make the dependency on the respective parameter sets explicit. Learning good parameters can then be done by performing stochastic optimization of $\mathcal{L}$ with respect to both $\theta^\rho$ and $\theta^\rho$, where the expectation term is approximated by single draws from $q$ in each training step.

Designing a model is then done by the following steps: (1) Choice of a prior $p(z)$ over the latent variables. (2) Choice of a recognition model $q(z|x, \theta^\rho)$. The Kullback-Leibler divergence between the prior and the recognition model has to be tractable and efficient to compute. (3) Choice of a generating model $p(x|z, \theta^\rho)$, which is often given by the type of data under investigation.

We want to stress that even if $p(z)$ follows a distribution with a single maximum (e.g. a Gaussian) and factorises, the induced $p(x)$ will generally be multi-modal. First note that a simple rotation for $f: z \rightarrow x$ suffices to introduce dependencies among the components. For introducing new modes, an informal argument is as follows. Let $x \in \mathcal{X}$, $z \in \mathcal{Z}$ and $f$ be a parametrised universal approximator (such as an MLP or an RNN). If $f$ is smooth, arbitrary connected subsets $X_k \subset \mathcal{X}$ can be mapped arbitrarily to connected subsets $Z_k \subset \mathcal{Z}$. This allows changes in volume and hence probability density, possibly giving rise to new maxima of the probability landscape.

3 Methods

We propose to combine SGVBs and RNNs by making use of an sRNN for both the recognition model $q(z_{1:t-1}|x_{1:t-1})$ and the generating model $p(x_t|z_{1:t})$. We will also treat the hidden states of the generating models, $h_{1:T}$ as random variables following a Dirac distribution with infinite mass at a single point.

The derivation of the training criterion is as follows.

$$-\log p(x_{1:T}) = -\log \prod_{t=1}^{T} p(x_t|x_{1:t-1})$$

$$= -\log \prod_{t=1}^{T} \int h_{t-1} p(x_t|h_{t-1}, x_{1:t-1}) p(h_{t-1}) dh_{t-1}$$

$$= -\log \prod_{t=1}^{T} p(x_t|h_{t-1}),$$

where we have used the fact that $x_t$ is conditionally independent of $x_{1:t-1}$ given $h_{t-1}$ which follows a Dirac distribution. We then introduce a set of latent variables $z_{1:T}$ which we marginalise over. Assuming that each $x_t$ is conditionally independent of $z_{t+1:T}$ lets us arrive at

$$-\log p(x_{1:T}) = -\log \int_{z_{1:T}} p(z_{1:T}) \prod_{t=1}^{T} p(x_t|h_{t-1}, z_{1:t}, z_{t+1:T}) dz_{1:T}.$$ 

We then obtain a variational upper bound by use of Jensen’s inequality:

$$-\log p(x_{1:T}) = -\log \int_{z_{1:T}} \frac{q(z_{1:T}|x_{1:T})}{q(z_{1:T}|x_{1:T})} p(z_{1:T}) \prod_{t=1}^{T} p(x_t|h_{t-1}, z_{1:t}) dz_{1:T}$$

$$\leq KL(q(z_{1:T}|x_{1:T})||p(z_{1:T})) - \mathbb{E}_{z_{1:T} \sim q} \sum_{t=1}^{T} \log p(x_t|h_{t-1}, z_{1:t}) \quad (2)$$

$$: = \mathcal{L}_{\text{STORN}}$$

In this work, we restrict ourselves to a standard Normal prior\footnote{In a preliminary report, we proposed the use of a Wiener process for a prior. However, the presented results were invalid due to implementation errors and the paper has been withdrawn.} of the form

$$p(z_{1:T}) = \prod_{t,k} \mathcal{N}(z_{t,k}|0, 1),$$

where $z_{t,k}$ is the value of the $k$th latent sequence at time step $t$. 

The only requirements. In order to minimise the upper bound of the negative log-likelihood $\mathcal{L}$ with numerical means, it is convenient to choose parametric models. In that case we write $p(x|z, \theta^\rho)$ and $q(z|x, \theta^\rho)$.
Table 1: Results on the midi data sets. All numbers are average negative log-likelihoods on the test set, where “FD-RNN” represents the work from [20]; “sRNN” and “RNN-NADE” results are from [9] while “Deep RNN” shows the best results from [19]. The results of our work are shown as “STORN” and have been obtained by means of the importance sampler described in [17].

| Data set     | STORN | FD-RNN | sRNN | RNN-NADE | Deep RNN |
|--------------|-------|--------|------|----------|----------|
| Piano-midi.de | 7.13  | 7.39   | 7.58 | 7.05     | –        |
| Nottingham   | 2.85  | 3.09   | 3.43 | 2.31     | 2.95     |
| MuseData     | 6.16  | 6.75   | 6.99 | 5.60     | 6.59     |
| JSBChorales  | 6.91  | 8.01   | 8.58 | 5.19     | 7.92     |

Remark The model reduces to an sRNN as soon as we remove any latent variables, e.g. by setting $W^g_w = 0$. This implies that STORNs are a generalisation and hence superior to sRNNs, given that both models are of the same architecture and find their respective optimal parameters.

Remark While we started out with each $h_t$ following a Dirac distribution, this is no longer the case as we introduced latent variables. In particular, each $h_t$ is a deterministic function of the latent variable at the current time step and the hidden layer at the previous time step, i.e. $h_t(h_{t-1}, z_t)$.

The recognition model $q$ will in this case be parametrised by a single mean $\mu_{t,k}$ and variance $\sigma^2_{t,k}$ for each time step and latent sequence. Both will be represented by the output of a recurrent net, which thus has $2\omega$ outputs of which the first $\omega$ (representing the mean) will be unconstrained, while the second $\omega$ (representing the variance) need to be strictly positive. Given the output $y_{1:T} = f^r(x_{1:T})$ of the recognition RNN $f^r$, we set

$$\mu_{t,k} = y_{t,k},$$
$$\sigma^2_{t,k} = y_{t,k+\omega}^2.$$

Note that the square ensures positiveness.

Going along with the reparametrisation trick of [18], we will sample from a standard Normal at each time step, i.e. $\epsilon_{t,k} \sim N(0, 1)$ and use it to sample from $q$ via $z_{t,k} = \mu_{t,k} + \sigma_{t,k}\epsilon_{t,k}$. Given the complete sample sequence $z_{1:T}$ we calculate the two terms of Equation (2). The KL-divergence can be readily computed, while we need to pass $z_{1:T}$ through the generating model $f^g$ which gives $-\log p(x_{1:T}|z_{1:T})$.

4 Experiments

For evaluation we trained the proposed model on a set of midi music, which was previously used to evaluate RNNs by [9] [19] [20] [12]. We also investigated modelling human motion in the form of motion capture data, previously used by [12] [14] [21]. We employ Fast Dropout Recurrent Networks (FD-RNNs) [20] for both the recognition and the generating model. While we determine the dropout rates for the generating model via model selection on a validation set, we include them into the parameter set for the recognition model. In a manner similar to [22], we exploit fast dropout’s natural inclusion of variance as the variance for the recognition model, i.e. $\sigma^2_{t,k}$. We used Adadelta [23] enhanced with Nesterov momentum [8] for optimisation.

4.1 Polyphonic Music Generation

All experiments were done by performing a random search [24] over the hyper parameters, where 128 runs were performed for each data set. Both the recognition and the generating model used 300 hidden units with the logistic sigmoid as the transfer function. We report the estimated negative log-likelihood on the test set of the parameters which yielded the best bound on the validation set.
Figure 1: Illustration of missing value imputation on the motion capture data set. We show the first 48 of the 49 channels of a random sample, where time steps 30 to 40 were initialised with random noise. Subsequently, a maximum a posteriori point estimate of the latent variables was used to reconstruct the missing parts of the signals.

4.2 Motion Capture Data

The motion capture data set \cite{25,21} is a sequence of kinematic quantities obtained from a human body during walking. It consists of 3128 time steps of 49 angular quantities each.\footnote{We want to stress that to comply with previous results, no training/validation/test split was used.}

For motion capture data, we chose a Gaussian likelihood with a fixed standard deviation for the generating model. The recognition model was chosen to be a bidirectional RNN. While the standard deviation was fixed to 1 during training, we performed a binary search for a better value after training; the resulting estimate of the negative log-likelihood on the validation set was then used for model selection.

The estimated negative log-likelihood of the data was 15.99. Other models trained on this data set, namely the RNN-RBM, RTRBM and eRBM do not offer a tractable way of estimating the log-likelihood of the data, which is why there is no direct mean of comparison respecting the probabilistic nature of the models. In the case of the former two, the mean squared prediction error is reported instead, which is 20.1 and 16.2 respectively. Our method achieved an average MSE of 4.94, which is substantially less than previously reported results.

For additional means of comparison, we performed missing value imputation of motion capture data. We picked random sequences of length 60 and replaced all of the 49 channels from time steps 30 to 40 with standard normal noise. We then performed a maximum a posteriori point selection of the recognition model, i.e. \( \arg\max_{\hat{z}_{1:T}} q(\hat{z}_{1:T}|x_{1:T}) \), from which we reconstructed the output via \( \arg\max_{\hat{x}_{30:40}} \log p(x_{1:T}|\hat{z}_{1:T}) \). We show the results of the imputations in Figure 1.

To demonstrate the generative capabilities of the method, we drew 50 samples from the model after initialising it with a stimulus prefix. The stimulus had a length of 20, after which we ran the model in “generating mode” for another 80 time steps. This was done by feeding the mean of the model’s output at time step \( t \) into the generating model at time step \( t + 1 \). Additionally, we drew \( z_{20:80} \) from the prior. The results are visualised in Figure 2.
Figure 2: Samples from the model trained on motion capture data after providing a stimulus prefix sequence of 20 time steps. The uncertainty of the learned distribution is visible by the diversity of the samples; nevertheless, the distribution is rather unimodal.

5 Conclusion and Future Work

We have presented a model class of stochastic RNNs that can be trained with a recently proposed estimator, SGVB. The resulting model fulfills the expectation to greatly improve over the performance of RNNs erroneously assuming a factorisation of output variables. Still, RNN-NADE has a competitive edge; nevertheless use of deeper architectures, better training procedures or a more adequate prior might close the gap, which needs to be verified experimentally. We will also evaluate the use of STORN for supervised learning, where the generating model is additionally conditioned on some external input.

References

[1] Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

[2] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. arXiv preprint arXiv:1303.5778, 2013.

[3] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. arXiv preprint arXiv:1409.3215, 2014.

[4] Alex Graves, Santiago Fernández, Marcus Liwicki, Horst Bunke, and Jurgen Schmidhuber. Unconstrained online handwriting recognition with recurrent neural networks. Advances in Neural Information Processing Systems, 20:1–8, 2008.

[5] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[7] J. Martens and I. Sutskever. Learning recurrent neural networks with hessian-free optimization. Proc. 28th Int. Conf. on Machine Learning, 2011.

[8] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. 2013.
[9] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu. Advances in optimizing recurrent networks. arXiv preprint arXiv:1212.0901, 2012.

[10] Barbara Hammer. On the approximation capability of recurrent neural networks. Neurocomputing, 31(1):107–123, 2000.

[11] Hava T Siegelmann and Eduardo D Sontag. Turing computability with neural nets. Applied Mathematics Letters, 4(6):77–80, 1991.

[12] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. arXiv preprint arXiv:1206.6392, 2012.

[13] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. High-dimensional sequence transduction. In ICASSP, 2013.

[14] I. Sutskever, G. Hinton, and G. Taylor. The recurrent temporal restricted boltzmann machine. Advances in Neural Information Processing Systems, 21, 2008.

[15] Max Welling, Michal Rosen-Zvi, and Geoffrey E Hinton. Exponential family harmoniums with an application to information retrieval. In Advances in neural information processing systems, pages 1481–1488, 2004.

[16] Yichuan Tang and Ruslan Salakhutdinov. A new learning algorithm for stochastic feedforward neural nets. 2013.

[17] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic back-propagation and variational inference in deep latent gaussian models. arXiv preprint arXiv:1401.4082, 2014.

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[19] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. arXiv preprint arXiv:1312.6026, 2013.

[20] Justin Bayer, Christian Osendorfer, Daniela Korhammer, Nutan Chen, Sebastian Urban, and Patrick van der Smagt. On fast dropout and its applicability to recurrent networks. arXiv preprint arXiv:1311.0701, 2013.

[21] Graham W Taylor, Geoffrey E Hinton, and Sam T Roweis. Modeling human motion using binary latent variables. In Advances in neural information processing systems, pages 1345–1352, 2006.

[22] Justin Bayer, Christian Osendorfer, Sebastian Urban, et al. Training neural networks with implicit variance. In Proceedings of the 20th International Conference on Neural Information Processing, ICONIP-2013, 2013.

[23] Matthew D Zeiler. Adadelta: An adaptive learning rate method. arXiv preprint arXiv:1212.5701, 2012.

[24] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. The Journal of Machine Learning Research, 13:281–305, 2012.

[25] Eugene Hsu, Kari Pulli, and Jovan Popović. Style translation for human motion. ACM Transactions on Graphics (TOG), 24(3):1082–1089, 2005.