Probabilistic Interpretations of Recurrent Neural Networks

Yo Joong Choe [*1] **Jaehyeok Shin** [*1] **Neil Spencer** [*1]

1. Introduction

Recurrent neural networks (RNNs) are the primary deep learning method for sequential data. In recent years, sophisticated versions of RNNs have shown success in various applications, including natural and syntactic languages (Graves, 2013; Sutskever et al., 2014; Vinyals et al., 2015; Kaparthy, 2015), music (Boulanger-Lewandowski et al., 2012), and videos (Srivastava et al., 2015).

In the presence of unsupervised sequential data, such as long passages of text with no labels, RNNs can model the generative process of the sequential data by learning to estimate the distribution of the next item in the sequence given all the previous ones. This is the key aspect of RNNs that allows them to "generalize" by generating sequences that come from the conditional distribution. This aspect also opens up connections to existing statistical models such as state space models and hidden Markov models. These existing models also perform the same task within the probabilistic graphical model framework. Recently, there has been work that attempts to formalize such connections (Mohamed, 2015; Andreas, 2016).

Our paper first formalizes the notion of RNNs as generative models in a fully probabilistic framework. This involves re-casting the standard computational model for RNNs as a graphical model. We show that the connection can be made precise, even when sophisticated mechanisms such as long short-term memory (Hochreiter & Schmidhuber, 1997) and deep architectures (Pascanu et al., 2014) are used.

We then provide a formal description of RNNs as nonlinear approximations of inference algorithms in probabilistic models. We first formalize the idea that RNNs are nonlinear approximations of the forward algorithm in hidden Markov models that have many hidden states.[1] We then establish the connection between RNNs and particle filters for state space models. In particular, we show that a class of RNNs can be viewed as approximations of the inference procedure via particle filters. This development is in line with other views of RNNs as inferential procedures (Andreas, 2016).

[*]Equal contribution [1]Carnegie Mellon University, Pittsburgh, PA, USA. Correspondence to: YJ Choe <yjchoe@cmu.edu>, Jaehyeok Shin <jaehyeos@andrew.cmu.edu>, Neil Spencer <nspencer@andrew.cmu.edu>.

Final Report, *10-708 Probabilistic Graphical Models*, 2017.

[1]We first learned about this idea from Prof. Jason Eisner's talk at the LTI Colloquium at CMU.

2. Background and Related Work

Throughout the paper, we consider the problem of modeling sequential data of the form $(x_1, \dots, x_T) \in \mathcal{X}^T$ for some domain $\mathcal{X}$ and time length T. Here, we assume that we have n independent observations, each having potentially different lengths T_i. When necessary, we denote each sequence as $(x_1^{(i)}, \dots, x_{T_i}^{(i)}) \in \mathcal{X}^T$ for $i = 1, \dots, n$.

2.1. Recurrent Neural Networks

Recurrent neural networks (RNNs) (Rumelhart et al., 1985; Werbos, 1988) are a class of neural networks that can process sequential data. In the context of our problem, RNNs can be used to predict the next item x_{t+1} in the sequence given $x_1, \dots, x_t$, for every $t = 1, \dots, T-1$. Assuming a single layer, an RNN that predicts the next item in sequences can be defined as

$$h_s = f_\alpha(h_{s-1}, x_s) \;\; \forall s = 1, \dots t$$
$$x_{t+1} = g_\beta(h_t)$$

for some nonlinear functions f_α and g_β that are parametrized by vector-valued parameters α and β respectively. Note that the functions are independent of time. The corresponding computational graph is drawn in Figure 1.

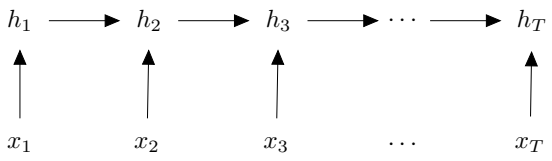

Figure 1. The computational graph of recurrent neural networks. Note that this is not a graphical model representation and edges represent deterministic computations.

A simple RNN defines these functions as

$$f_\alpha(h_{s-1}, x_s) = \mathcal{H}_f(Wx_s + Uh_{s-1} + b) \;\; \forall s = 1, \dots t$$
$$g_\beta(h_t) = \mathcal{H}_g(Vh_t + c)$$

where $\alpha = (W, U, b)$ and $\beta = (V, c)$ for some weight matrices W, U, and V as well as bias vectors b and c. $\mathcal{H}_f$ and $\mathcal{H}_g$ are nonlinear activation functions such as the sigmoid function, the hyperbolic tangent (tanh), or the rectified linear unit (ReLU).

A simple RNN can be trained using backpropagation, but when T is large it is not effective in modeling long-term dependencies due to the problem of vanishing gradients. Long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) and gated recurrent units (GRUs) (Cho et al., 2014) address this issue by defining f_α to be an over-parametrized function that preserves "memory" from the distant past. LSTMs, GRUs and their variants are a large part of RNN's recent success in natural language processing, speech recognition, and other real-world applications.

Stacked RNNs (Schmidhuber, 1992; El Hihi & Bengio, 1995; Graves, 2013), the most common variant of deep RNNs, build up extra layers of hidden units that accept the hidden units of the previous layer as their input sequences. Other variants of deep RNNs (Pascanu et al., 2014) include replacing the functions f_α and g_β with deep feedforward neural networks themselves.

2.2. State Space Models and Hidden Markov Models

State space models (SSMs) or dynamic linear models (DLMs) are a class of probabilistic graphical models that encode the dependency structure of sequential data by latent variables that follow Markovian dynamics. Examples of SSMs include hidden Markov models (HMMs) (Baum & Petrie, 1966), in which observations and latent states are assumed to be discrete, and linear Gaussian SSMs or the Kalman filtering model (Kalman et al., 1960), in which observations and latent states are assumed to be Gaussian.

In order to probabilistically model sequential data, we first introduce random variables $X_1, \dots, X_T$ defined on $\mathcal{X}$ to model observations $x_1, \dots, x_T$. We also introduce latent variables $H_1, \dots, H_T$ to model the hidden states corresponding to each item in the sequence. Then, the graphical model representation of SSMs can be drawn as in Figure 2.

2.3. Particle Filters (Sequential Monte Carlo)

The particle filter (or sequential Monte Carlo) is a stochastic inference strategy that can be applied to both HMMs and SSMs. It is essentially a sequential importance sampling algorithm with intermediate resampling steps. It approximates the target distribution $P(H_1, \dots, H_T | x_1, \dots, x_T)$ in an online fashion using a set of M weighted samples

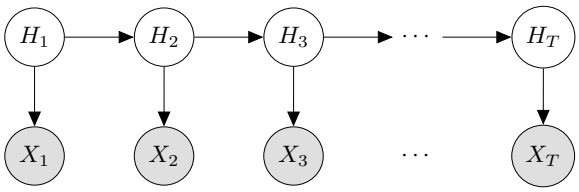

Figure 2. The graphical model for state space models, including hidden Markov models.

$$\left(H_1^{(1)}, \dots, H_T^{(1)}, W^{(1)}\right) \dots \left(H_1^{(M)}, \dots, H_T^{(M)}, W^{(M)}\right).$$

Let $q\left(H_t | H_{t-1}\right)$ denote a proposal distribution which is equal to, or is similar to, $P\left(H_t | H_{t-1}\right)$. Let H_0 be arbitrary. For a state space model, the particle filter algorithm is defined as follows.

- Propagate the particles $H_t^{(i)} \sim q(H_t | H_{t-1})$.
- Compute the weights

$$W_t^{(i)} = P(H_t | x_t, H_{t-1}) / q(H_t | H_{t-1}).$$

- Normalize the weights such that $\sum_{i=1}^M W_t^{(i)} = 1$.
- Resample to obtain M equally weighted particles $\left(H_1^{(1)}, \dots, H_t^{(1)}, 1/M\right) \dots \left(H_1^{(M)}, \dots, H_t^{(M)}, 1/M\right)$.

Iterating the above steps for $t = 1, \dots, T$ results in the approximation $P(H_1, \dots, H_t | x_1, \dots, x_t) \approx \sum_{i=1}^M W_t^{(i)} \delta_{H_{1:t}^{(i)}}$.

2.4. Related Work

While there is a lot of work on viewing recurrent neural networks as generative models, in the sense that the model can read in certain sequential data and predict the most likely future entries (Bengio et al., 2003; Graves, 2013), a formal treatment of RNNs as fully probabilistic graphical models has not been comprehensively studied.

In (Mohamed, 2015), Mohamed describes an interpretation of a simple RNN as the maximum likelihood estimate of a state space model that follows deterministic dynamics. In this sense, RNNs can be viewed as a sophisticated version of a state space model with potentially highly nonlinear transition functions. However, the precise graphical model representation of RNNs and their implicit conditional independence assumptions are not studied, and we explore these relationships formally in Section 3.1.

In (Andreas, 2016), Andreas illustrates the inevitable and often misleading entanglement of models and inference procedures in neural networks, which he thus refers to as *monferences*. He claims that, unlike standard probabilistic

models in which the inference procedure is separated from the model architecture, neural networks including RNNs implicitly build a model-inference pair that cannot easily be separately identified.

Andreas notes that, in recurrent neural networks, a hidden unit cannot simply be viewed as encoding a single hidden state as in a hidden Markov model, but rather encoding "the whole table of probabilities" of the HMM forward algorithm. In his experiments, an RNN trained on the hidden states and observations generated from a pre-defined HMM can learn to mimic the same predictions that an HMM would make using its forward probabilities. However, no mathematical formalization of the connection is made precise in the article, and we present formal characterizations of this connection from a functional approximation point-of-view (Section 3.2) and from an approximate inference point-of-view (Section 3.3).

3. Probabilistic Interpretations of RNNs

3.1. RNNs as Generative Models

In this section, we formally develop a probabilistic graphical model that is induced by a generative model formulation of an RNN and discuss its properties. For simplicity, we only consider the case of binary random sequences $(X_1, \dots, X_T) \in \{0,1\}^T$. This argument can be generalized for the general discrete random sequence case where $(X_1, \dots, X_T) \in \{1, \dots, K\}^T$ for some K.

For given parametrized deterministic functions f_α and g_β, such as sigmoidal activation units, LSTMs, and their deep extensions, an RNN induces a set of procedures to model the conditional distribution of X_{t+1} given $X_{\leq t}$ for all $t = 1 \dots T$. For instance, to model $P(X_{t+1} = 1 | X_{\leq t} = x_{\leq t})$, an RNN first calculates a set of inner states $\{h_s\}_{s=0}^t$ which is recursively defined by

$$h_0 := 0, \;\; h_s = f_\alpha(h_{s-1}, x_s) \;\; \forall s = 1, \dots t$$

Then, the conditional probability is modeled by

$$P(X_{t+1} = 1 | X_{\leq t} = x_{\leq t}; \alpha, \beta) = g_\beta(h_t)$$

Note that h_t depends on $x_1, \dots x_t$ and α. By the following equation, we can model the conditional distribution of $(X_2, \dots, X_T)$ given X_1 by using the RNN.

$$P(X_2, \dots X_T | X_1) = \prod_{t=1}^{T-1} P(X_{t+1} | X_{\leq t})$$

For given n training sequences $\{x_1^{(i)}, \dots x_{T_i}^{(i)}\}_{i=1}^n$, we can estimate parameters (α, β) by using the maximum conditional likelihood estimator

$$\begin{aligned}(\hat{\alpha}, \hat{\beta}) &= \underset{\alpha,\beta}{\operatorname{argmax}} \sum_{i=1}^{n} \log P\left(x_2^{(i)}, \dots x_{T_i}^{(i)} | x_1^{(i)}; \alpha, \beta\right) \\ &= \underset{\alpha,\beta}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{t=1}^{T_i - 1} \log P\left(x_{t+1}^{(i)} | x_{\leq t}^{(i)}; \alpha, \beta\right)\end{aligned}$$

Remark 1. *The RNN does not model the marginal distribution of X_1. To model the full joint distribution of $(X_1, \dots, X_T)$, we need to introduce an additional model for $P(X_1)$.*

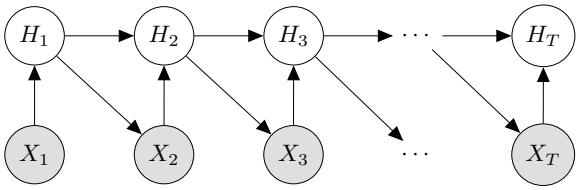

Figure 3. A fully probabilistic graphical model representation of recurrent neural networks.

Given the generative formulation of an RNN, we can derive a graphical model interpretation based on the graph G described in the figure 3.

Remark 2. *The graphical model in Figure 3 is also generative, in the sense that we can sample observations and hidden states from the joint model (given X_1). However, we distinguish it from the standard generative formulation because it is also a probabilistic graphical model that further encodes conditional independence relationships.*

The structure of the graph G induces the following conditional independence relationships.

$$\begin{aligned}X_{t+1} \perp\!\!\!\perp X_{\leq t} \;\mid\; H_t \;\; \forall t = 1, \dots, T-1 \\ H_t \perp\!\!\!\perp X_{\leq t-1} \;\mid\; H_{t-1}, X_t \;\; \forall t = 2, \dots, T-1\end{aligned}$$

That is, the future observation is independent to current and past observations given the current hidden state, and the current hidden state is independent to past observations given the past hidden state and current observation.

For given RNN parameters (α, β), we can define a graphical model $M(G; \alpha, \beta)$ which has the graph structure G and the following parameterizations

$$\begin{aligned}P(X_{t+1} = 1 | H_t = h_t) &= g_\beta(h_t) \\ P(H_t | H_{t-1} = h_{t-1}, X_t = x_t) &= \delta\left(H_t = f_\alpha(h_{t-1}, x_t)\right) \\ P(H_1 | X_1 = x_1) &= \delta\left(H_1 = f_\alpha(0, x_1)\right)\end{aligned}$$

Remark 3. *Unlike in the generative model parametrization of an RNN in which $\{h_t\}_{t=1}^T$ are deterministic functions of $\{x_t\}_{t=1}^T$, the graphical model parameterization do*

not assume such deterministic relationships between the random variables $\{H_t\}_{t=1}^T$ and $\{X_t\}_{t=1}^T$.

These particular parameterizations induce additional conditional independence relationships which do not necessarily hold for general graphical models based on G.

Proposition 4. *Under the graphical model parametrization, the graphical model $M(G;\alpha,\beta)$ has the following additional properties.*

$$H_t \perp\!\!\!\perp X_{\geq t+1} \mid H_{t-1}, X_t \ \forall t$$

which implies that the current hidden state is not only independent to the past observations, but it is also independent to the future observations, given the past hidden state and current observation.

Proof. Conditioned on $(H_{t-1} = h_{t_1}, X_t = x_t)$, H_t has a point mass on $f_\alpha(h_{t-1}, x_t)$. Since a constant variable is independent to any other variables, the result follows. □

Based on Proposition 4, we can show that the graphical model $M(G;\alpha,\beta)$ is equivalent to the RNN in the generative model point of view.

Theorem 5. *The graphical model $M(G;\alpha,\beta)$ described above induces the same conditional distribution of $(X_2,\ldots,X_T)$ given X_1 as one induced by the corresponding RNN.*

Proof. To distinguish the RNN and the graphical model cases, let $\{h_s^*\}_{s=0}^t$ be the set of inner states in the RNN calculated on a realization $\{x_s\}_{s=1}^t$ of $\{X_s\}_{s=1}^t$. To prove the theorem, it is enough to show that

$$P(X_{t+1} = 1 | X_{\leq t} = x_{\leq t}) = g_\beta(h_t^*)$$

which is followed by

$$\begin{aligned}
&P(X_{t+1} = 1 | x_{\leq t}) \\
&= \int P(X_{t+1} = 1 | h_t, x_{\leq t}) P(H_t = h_t | x_{\leq t})\, dh_t \\
&= \int P(X_{t+1} = 1 | h_t, x_{\leq t}) \times \\
&\qquad P(H_t = h_t | h_{t-1}, x_{\leq t}) P(H_{t-1} = h_{t-1} | x_{\leq t})\, dh_t h_{t-1} \\
&= \int P(X_{t+1} = 1 | h_t) \times \\
&\qquad P(H_t = h_t | h_{t-1}, x_t) P(H_{t-1} = h_{t-1} | x_{\leq t})\, dh_t dh_{t-1} \\
&= \int g_\beta(h_t) \times \\
&\qquad \delta(h_t = f_\alpha(h_{t-1}, x_t)) P(H_{t-1} = h_{t-1} | x_{\leq t})\, dh_t dh_{t-1} \\
&= \int g_\beta(f_\alpha(h_{t-1}, x_t)) P(H_{t-1} = h_{t-1} | x_{\leq t})\, dh_{t-1} \\
&= \int g_\beta(f_\alpha(h_{t-1}, x_t)) P(H_{t-1} = h_{t-1} | h_{t-2}, x_{\leq t}) \times \\
&\qquad P(H_{t-2} = h_{t-2} | x_{\leq t})\, dh_{t-1} dh_{t-2} \\
&= \int g_\beta(f_\alpha(h_{t-1}, x_t)) P(H_{t-1} = h_{t-1} | h_{t-2}, x_{t-1}) \times \\
&\qquad P(H_{t-2} = h_{t-2} | x_{\leq t})\, dh_{t-1} dh_{t-2} \\
&= \int g_\beta(f_\alpha(h_{t-1}, x_t)) \delta(h_{t-1} = f_\alpha(h_{t-2}, x_{t-1}) \times \\
&\qquad P(H_{t-2} = h_{t-2} | x_{\leq t})\, dh_{t-1} dh_{t-2} \\
&= \int g_\beta(f_\alpha(f_\alpha(h_{t-2}, x_{t-1}), x_t)) \times \\
&\qquad P(H_{t-2} = h_{t-2} | x_{\leq t})\, dh_{t-2} \\
&\vdots \\
&= g_\beta(h_t^*)
\end{aligned}$$

□

3.2. RNNs as Nonlinear Approximations of Forward Algorithms in HMMs with a Large Number of Hidden States

Let us assume that we have an HMM with binary observable random sequence $\{X_1,\ldots,X_T\} \in \{0,1\}^T$ and hidden units with m-hidden states $\{H_1,\ldots,H_T\} \in \{1,\ldots,m\}^T$. The forward algorithm is used to update the belief state, which is the probability of a hidden unit given the previous and current observations. Formally, we define the belief state at time t by

$$\alpha_t(h_t) := P(H_t = h_t, X_{\leq t} = x_{\leq t}), \ \forall t$$

Since each H_t has m possible hidden states, we can represent $\alpha_t(\cdot) := \alpha_t \in [0,1]^m$ as a vector in the m-dimensional unit cube. Then, update rules for belief states can be expressed by linear mappings $\tilde{f}_x : [0,1]^m \to [0,1]^m, \ x \in \{0,1\}$ defined by

$$\alpha_{t+1} = D_{x_{t+1}} V \alpha_t := \tilde{f}_{x_{t+1}}(\alpha_t), \ \forall t$$

where $D_x, V \in [0,1]^{m\times m}$ are given by

$$\begin{aligned}
D_1 &= \mathrm{diag}\,[P(X_t = 1 | H_t = 1), \ldots, P(X_t = 1 | H_t = m)] \\
D_0 &= \mathrm{diag}\,[P(X_t = 0 | H_t = 1), \ldots, P(X_t = 0 | H_t = m)] \\
V_{ij} &= P(H_{t+1} = i | H_t = j)
\end{aligned}$$

Note that D_x and V are well-defined because emission and transition probabilities do not depend on time t in

the HMM. Once we have $\tilde{f}_x$ and the initial belief state $\alpha_0 := (P(H_1 = 1), \dots, P(H_1 = m)) \in [0,1]^m$, we can calculate all belief states $\{\alpha_t\}_{t=1}^T$. Then, by using each belief state, we can calculate the conditional distribution of X_{t+1} given the previous observations by

$$\begin{aligned}
&P(X_{t+1} = 1 | X_{\leq t} = x_{\leq t}) \\
&= \frac{P(X_{t+1} = 1, X_{\leq t} = x_{\leq t})}{P(X_{\leq t} = x_{\leq t})} \\
&= \frac{\sum_{j=1}^m P(H_{t+1} = j, X_{t+1} = 1, X_{\leq t} = x_{\leq t})}{\sum_{j=1}^m P(H_t = j, X_{\leq t} = x_{\leq t})} \\
&= \frac{\mathbf{1}^\top \tilde{f}_1(\alpha_t)}{\mathbf{1}^\top \alpha_t} \\
&=: \tilde{g}(\alpha_t)
\end{aligned}$$

If the number of hidden states m is extremely large, the HMM can model any dependency structure over the observable sequence with enough precision. However, to describe the corresponding high-dimensional linear dynamics of the sequence of belief states $\{\alpha_t\}_{t=1}^T$, we need estimate $O(m^2)$ parameters which become intractable for extremely large m.

In this case, we can interpret RNNs as procedures providing low-dimensional non-linear approximations for high-dimensional linear dynamics of the belief states.

Proposition 6. *Let $\{a_t\}_{t=1}^T$ be a sequence of belief states induced by $\tilde{f}_x, \tilde{g}$ in an HMM with m hidden states. For a given class of RNNs with f_α, g_β, if there exist α^*, β^* and $\phi : \mathbf{R}^m \to \mathbf{R}^k$ satisfying the following conditions*

C.1 $\phi(\alpha_0) = h_0$

C.2 $\phi \circ \tilde{f}_x = f_{\alpha^*, x} \circ \phi, \ \ \forall x$

C.3 $\tilde{g}_\beta = g_{\beta^*} \circ \phi$

then $\{a_t\}_{t=1}^T$ and $\{h_t\}_{t=1}^T$ are equivalent in the sense that the two sequences yield the same calculations of conditional probabilities $\{P(X_{t+1} = 1 | X_{\leq t} = x_{\leq t})\}_{t=1}^{T-1}$. Here, $f_{\alpha,x}(\cdot) = f_\alpha(\cdot, x)$, k is the dimension of h_t in the RNN, and $\{h_t\}_{t=1}^T$ is the sequence of hidden units in the RNN with parameters (α^*, β^*) calculated using a given observation sequence $\{x_t\}_{t=1}^T$.

Proof. Let us first show that $h_t = \phi(\alpha_t), \ \ \forall t$. By the first condition, it holds for $t = 0$. For $t > 0$, note that

$$\begin{aligned}
\phi(\alpha_{t+1}) &= \phi\left(\tilde{f}_x(\alpha_t)\right) \\
&= f_{\alpha^*, x}\left(\phi(\alpha_t)\right) \ \text{ by the } C.2 \text{ condition} \\
&= f_{\alpha^*, x}(h_t) \ \text{ by the induction assumption} \\
&= h_{t+1}
\end{aligned}$$

Therefore, by the mathematical induction, we get $h_t = \phi(\alpha_t), \ \ \forall t$. From this relationship and the $C.3$ condition, the result is followed by

$$\begin{aligned}
P_{\text{HMM}}(X_{t+1} = 1 | x_{\leq t}) &= \tilde{g}(\alpha_t) \\
&= g_{\beta^*}\left(\phi(\alpha_t)\right) \\
&= g_{\beta^*}(h_t) \\
&= P_{\text{RNN}}(X_{t+1} = 1 | x_{\leq t})
\end{aligned}$$

□

Remark 7. *Once we can guarantee that the conditions in Proposition 6 are satisfied by a function ϕ, we do not need to find ϕ explicitly to construct the RNN.*

If we use sufficiently rich non-linear function classes for f_α and g_β, the conditions in Proposition 6 can be satisfied. Thus, the RNN can model HMMs with a large number of hidden states exactly. However, if function class used is too rich, RNNs can easily over-fit the data which make large variances in estimations. If the function class used is not the rich enough, the conditions in Proposition 6 could be significantly violated, resulting in large biases in prediction. In practice, we can control sizes of functions classes by varying the dimension of hidden units in RNNs with fixed activation functions.

3.3. A Connection Between RNNs and Particle Filters for HMMs/SSMs

In this section, we briefly comment on connections between the particle filter algorithm for HMMs and recurrent neural networks. In particular, we show that if we treat the particles of the particle filter as proxies for the hidden states in a state space model, we get a graphical model structure that is identical to that of a RNN. We also elaborate on the functional form of this RNN.

Let $H_t^{(1)}, \dots H_t^{(M)}$ denote the the states of M particles of a particle filter at time t (after re-sampling) which is targeting an HMM (or SSM). By definition of the particle filter, $H_t^{(1)}, \dots, H_t^{(M)}$ depends on both the previous particles $H_{t-1}^{(1)}, \dots H_{t-1}^{(M)}$ (through the propagation step) and x_t through the re-sampling step.

If when predicting x_{t+1}, we let $H_t^{(1)}, \dots, H_t^{(M)}$ act as the true hidden state of the HMM/SSM, then our prediction for x_{t+1} depends on $H_t^{(1)}, \dots, H_t^{(M)}$. Combining these dependencies, we get the graphical model depicted in Figure 4 below.

The graphical model in Figure 4 is equivalent to the one depicted in Figure 3. Thus, we can interpret the defined model as an RNN as follows.

The particles $H_t^{(1)}, \dots H_t^{(M)}$ (after re-sampling) are the hidden units of the RNN at time t. The hidden state up-

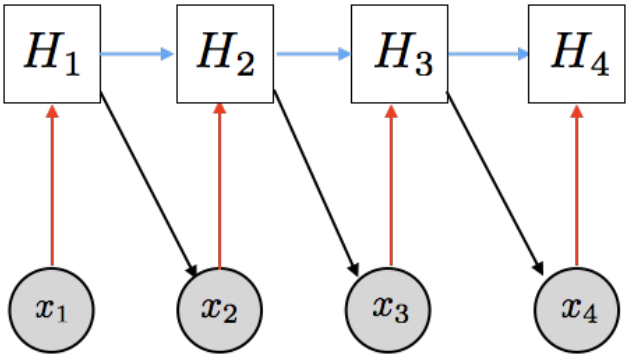

Figure 4. A graphical model depiction of a particle filter for a HMM where the particles at the previous state act as a proxy for hidden state of the HMM when doing forward prediction. Blue, red, and black lines indicate propagation, re-sampling, forward simulation, respectively.

date $h_t = f_\alpha(h_{t-1}, x_t)$ can be decomposed into two parts, f^p_α (propagation) and f^r_α (re-sampling), the two steps of the particle filter. Here, f^p_α takes h_{t-1} as an input, and f^r_α takes (G_t, h_{t-1}, x_t) as an input, where G_t is the output of $f^p_\alpha(h_{t-1})$ (the proposed state).

To make the connection with the particle filter more explicit, $f^p_\alpha(h_{t-1}) = (q^*_\alpha(H^{(i)}_{t-1}, h_0))_{i=1\ldots M}$, where $q^*_\alpha(H^{(i)}_{t-1}, h_0)$ is the inverse cdf of a proposal distribution $q(\cdot|h_{s-1})$ parameterized by α. Here, h_0 can be thought of as the random seed for the proposal, which is injected at the beginning of the chain.

We define $f^r_\alpha(h^*_t, h_{t-1}, x_t) = (v_i(G_t, h_{t-1}, x_t))_{i=1\ldots M}$, where

$$v_i(G_t, h_{t-1}, x_t) = \sum_{j=1}^{M} G^{(j)}_t I(a_i(G_t, h_{t-1}, x_t) = j), \quad (1)$$

$$a_i(G_t, h_{t-1}, x_t) \sim \text{Multinomial}\left(W^{(1)}, \ldots, W^{(M)}\right). \quad (2)$$

The stochasticity of the multinomial is removed by the seed h_0, and the values W are defined as in the weight calculations in the particle filter in Section 2.3. Note that α includes the parameters required to derive the weights.

We define $g_\beta(h_t)$ in our RNN to also involve two steps. First, it samples a hidden state from the existing particles. Then, it emulates a forward sample from the SSM given the hidden state. We can use a similar inverse cdf approach as above to make the function deterministic.

By construction, for a given α, β this definition of an RNN emulates the performance of a particle filter with a set seed.

It has been difficult to further establish an RNN which targets a particle filter explicitly. We continue to explore this avenue for future work.

4. Experiments

Given the theoretical nature of our project, we focus on verifying our claims of equivalence between models and/or inference procedures using synthetic data. We implemented recurrent neural networks using TensorFlow v1.1 (Abadi et al., 2015) and hidden Markov models using hmmlearn v0.2 (`http://hmmlearn.readthedocs.org`). The particle filter was implemented in the open-source statistical package R.

In all of our experiments, we use $n = 20000$ when training RNNs, mainly for computational reasons. Note that HMM forward and particle filters rely on the knowledge of true HMM parameters, i.e. they use infinitely many data points. We expect that some of the large RNNs that appear to perform sub-optimally below can in fact perform as well given more data points.

4.1. RNNs and the Forward Algorithm for HMMs

As our first experiment, we demonstrate the ability of RNNs to replicate the forward algorithm for HMMs without explicitly encoding the inference procedure. This experiment reproduces the results from (Andreas, 2016), which categorizes RNNs as monferences based on their ability to imitate inference algorithms.

We start with a known hidden Markov model with 2 hidden states and 3 observed states. The initial, transition, and emission probabilities were sampled from a Dirichlet distribution with all parameters equal to 1. The goal of the experiment is to infer the current hidden state H_t given current and previous observations $X_1, \ldots, X_t$, for each timestep $t = 1, \ldots, 8$. The performance is evaluated on a fixed test set of size $n_{test} = 100$ that is generated from the true model.

Given that we know all parameters of the model, the ideal inference procedure here is to apply the Bayes' rule, which for our particular model would correspond to the forward algorithm. The algorithm computes the forward probability $p(H_t \mid X_1, \ldots, X_t)$ for each t, using the model parameters (already known in this case) as well as the observation sequence up to time t. Once we have the forward probabilities, we can make a prediction $\hat{H}_t$ by choosing the hidden state with the highest forward probability.

Alternatively, also because we know the true model, we can generate some training data from the true model and train a unidirectional recurrent neural network, with the observation sequences as inputs and their corresponding hidden states as outputs. We do not use the model parameters except during data generation. This gives an inference proce-

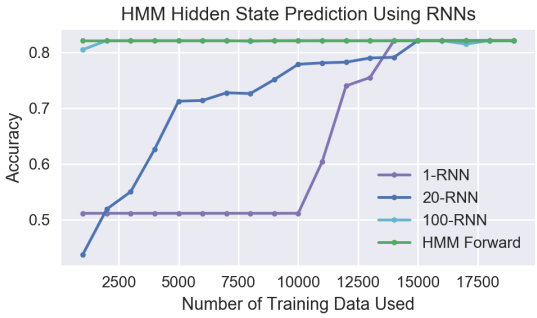

Figure 5. Hidden state prediction accuracy of k-RNNs, where k is the number of RNN hidden units ($H_t \in \mathbb{R}^k$). Each RNN is trained using the hidden and observed sequences generated from a known 2-state, 3-outcome HMM. After training, the predicted hidden states on a test set were exactly the same as those from the HMM forward algorithm.

dure for the hidden states, in the sense that it also estimates the forward probability $p(H_t \mid X_1, \ldots, X_t)$. If the neural network is able to imitate the ideal inference procedure, then we know that the neural network is capable of figuring out the forward inference algorithm by itself (given adequate amount of training data).

Figure 5 shows the results of this experiment using RNNs containing different number of hidden units ($k = 1, 20, 100$). It shows that, given sufficiently many samples generated from the true model (e.g. $n = 20000$), even an RNN with a single hidden unit can recover the predictions of the HMM forward algorithm exactly. Further, by increasing the capacity of the RNN and properly adjusting the learning rate, we can recover the HMM forward predictions with much less training data. These results confirm that RNNs can indeed replicate the forward algorithm given sufficient amount of data from the true model instead of its parameters.

It is important to note in Figure 5 that neither the HMM forward predictions nor the RNN predictions are perfect. In fact, the accuracy of HMM forward predictions can be viewed as the information-theoretical optimum for the n_{test} observation sequences, since the conditional probability is computed exactly from the true model itself. As a result, while one may falsely assume that an RNN with large enough capacity can achieve a 100% accuracy, it cannot outperform this optimal prediction in this case. This means that an RNN is able to capture the optimal inference procedure by means of observing sufficiently many samples instead of directly using the true parameters.

4.1.1. Increasing the Number of Hidden States

An interesting follow-up task from the previous subsection is to fix the capacity of the recurrent neural network and

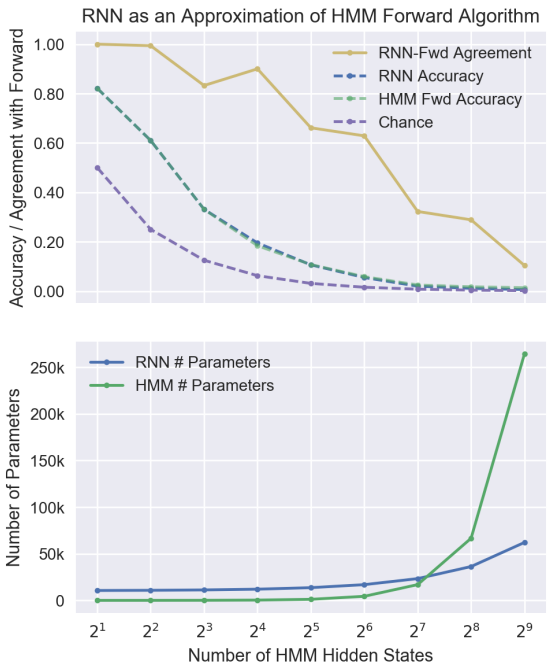

Figure 6. Hidden state prediction using RNNs and the forward algorithm for m-hidden-state HMMs across different m's. *Top:* Prediction accuracy of an RNN and the forward algorithm, as well as the agreement of the two predictions. Purple line indicates making the correct or identical prediction by chance. *Bottom:* Number of parameters of an RNN and an HMM corresponding to the prediction task.

see how well it can replicate the forward algorithm as the number of hidden states in the true hidden Markov model, denoted as m, increases. In Figure 6, we show the results of this experiment for $m = 2, 4, 8, \ldots, 512$.

In the top plot of Figure 6, it is shown that even the information-theoretically optimal forward algorithm (green) performs very poorly as the number of hidden states in the HMM increases exponentially. The purple line indicating prediction by chance also shows that the prediction task gets exponentially harder at the same time. What is important here is that the RNN prediction accuracy (blue) still matches the optimal accuracy, although the agreement between the two predictions (yellow) appear to decrease. The agreement drops significantly when the number of HMM parameters surpasses that of RNN (shown in the bottom plot).

This first suggests that the RNN is still able to replicate an inference algorithm that matches the performance of the optimal algorithm. While the result may further appear to suggest that the RNN is no longer capable of fully replicating the forward algorithm, we believe that the disagreement is more likely due to the existence of many local optima, with the forward algorithm being one of them. As a re-

sult, the RNN instead gives an approximation of the HMM forward algorithm, as we described in Section 3.2, that performs equally well in the inferential task.

We finally note that the forward probability prediction task is only the first half of eventually estimating the conditional probability of the next sequence item, i.e. $p(X_{t+1} \mid X_1, \ldots, X_t)$. We believe that it would be an interesting future work to further compare this probability between the classical and RNN-based approaches.

4.2. RNNs and Particle Filters

Building on the connection we showed in Section 3.3, we now empirically investigate the interpretation of the hidden unit of an RNN as particles in a particle filter. Again, we consider the task of estimating the hidden states of an HMM. Here, we consider an HMM with five hidden states and ten observed states. Again, the sequences observed are of length eight.

Figure 7 demonstrates the performance of both an RNN and a particle filter with varying numbers of particles/hidden units. Like in the previous section, the RNN is trained using many observations of hidden state/observed state pairs, and the particle filter uses the parameters of the true model.

Clearly, for any fixed dimension shown, the RNN outperforms the particle filter in both prediction and learning optimal inference. This is evidence that the RNN is able to learn a more efficient representation than the particle filter given the same dimension. This does not necessarily mean that an RNN is "better" than a particle filter, just that it is not equivalent.

One obvious explanation for this is that the RNN has real-valued hidden units whereas a particle filter has discrete-valued hidden states. This means that a single RNN hidden unit is much more expressive than a single particle. Thus, despite the fact that the graphical model shown in Section 3.3 corresponds to an RNN, an RNN's hidden states do not correspond to particles of a particle filter.

It is worth noting that with enough particles, a particle filter will perform identically to the forward algorithm.

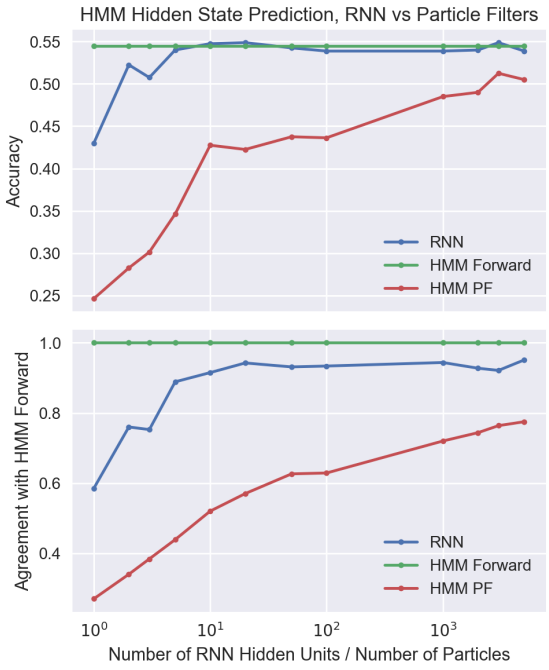

Figure 7. Comparison of RNNs with the particle filters for a fixed number of hidden units/particles. *Top*: Prediction accuracy of true hidden states from RNN and from the particle filter *Bottom:* The percentage of agreeance with an optimal inference procedure (the forward algorithm)

5. Conclusion and Discussion

In summary, we presented, formalized and empirically validated three different probabilistic interpretations of recurrent neural networks. First, we showed that RNNs can be interpreted as probabilistic graphical models that involve deterministic transition probabilities between the hidden variables. Second, RNNs can be understood as non-linear approximations of the forward algorithm for hidden Markov models. Third, RNNs can also be viewed as having connections to the particle filter algorithm for hidden Markov models and state space models, though the two are not identical.

While these characterizations of RNNs are by no means exhaustive, the three disparate views of RNNs already suggest that there is more work to be done in building a clear framework for these model inference pairs and, ultimately, in better understanding what these neural networks are actually learning. We believe that further understanding these characterizations and unifying them can help us comprehend and improve upon the shortcomings of existing RNN architectures, as well as provide better theoretical guarantees of their performance.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mané, Dan, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

Andreas, Jacob. A neural network is a monference, not a model. http://blog.jacobandreas.net/monference.html, 2016.

Baum, Leonard E and Petrie, Ted. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

Bengio, Yoshua, Ducharme, Réjean, Vincent, Pascal, and Jauvin, Christian. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb): 1137–1155, 2003.

Boulanger-Lewandowski, Nicolas, Bengio, Yoshua, and Vincent, Pascal. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In Langford, John and Pineau, Joelle (eds.), *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pp. 1159–1166, New York, NY, USA, July 2012. Omnipress. ISBN 978-1-4503-1285-1.

Cho, Kyunghyun, van Merriënboer, Bart, Gülçehre, Çalar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1179.

El Hihi, Salah and Bengio, Yoshua. Hierarchical recurrent neural networks for long-term dependencies. In *Nips*, volume 409, 1995.

Graves, Alex. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

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

Kalman, Rudolph Emil et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

Kaparthy, Andrej. http://karpathy.github.io/2015/05/21/rnn-effectiveness/. http://karpathy.github.io/2015/05/21/rnn-effectiveness/, 2015.

Mohamed, Shakir. A statistical view of deep learning, 2015.

Pascanu, Razvan, Gulcehre, Caglar, Cho, Kyunghyun, and Bengio, Yoshua. *How to construct deep recurrent neural networks*. 2014.

Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

Schmidhuber, Jürgen. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

Srivastava, Nitish, Mansimov, Elman, and Salakhutdinov, Ruslan. Unsupervised learning of video representations using lstms. In *ICML*, pp. 843–852, 2015.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Vinyals, Oriol, Kaiser, Łukasz, Koo, Terry, Petrov, Slav, Sutskever, Ilya, and Hinton, Geoffrey. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pp. 2773–2781, 2015.

Werbos, Paul J. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988.