Chapter 1
Recurrent Neural Networks and Related Models

Abstract A recurrent neural network (RNN) is a class of neural network models where many connections among its neurons form a directed cycle. This gives rise to the structure of internal states or memory in the RNN, endowing it with the dynamic temporal behavior not exhibited by the DNN discussed in earlier chapters. In this chapter, we first present the state-space formulation of the basic RNN as a nonlinear dynamical system, where the recurrent matrix governing the system dynamics is largely unstructured. For such basic RNNs, we describe two algorithms for learning their parameters in some detail: 1) the most popular algorithm of backpropagation through time (BPTT); and 2) a more rigorous, primal-dual optimization technique, where constraints on the RNN's recurrent matrix are imposed to guarantee stability during RNN learning. Going beyond basic RNNs, we further study an advanced version of the RNN, which exploits the structure called long short term memory (LSTM), and analyze its strengths over the basic RNN both in terms of model construction and of practical applications including some latest speech recognition results. Finally, we analyze the RNN as a bottom-up, discriminative, dynamic system model against the top-down, generative counterpart of dynamic system as discussed in Chapter **??**. The analysis and discussion lead to potentially more effective and advanced RNN-like architectures and learning paradigm where the strengths of discriminative and generative modeling are integrated while their respective weaknesses are overcome.

1.1 Introduction

As discussed in previous chapters, for many years and until the recent rise of deep learning technology as we reviewed in the several preceding chapters, automatic speech recognition (ASR) technology had been dominated by a "shallow" architecture - hidden Markov models (HMMs) with each

state characterized by a Gaussian mixture model (GMM), which we covered in some detail in Chapters ?? and ??. While significant technological successes had been achieved using complex and carefully engineered variants of GMM-HMMs and acoustic features suitable for them, researchers had for long anticipated that the next generation of ASR would require solutions to many new technical challenges under diversified deployment environments and that overcoming these challenges would likely require *deep* architectures that can at least functionally emulate the human speech recognition system known to have dynamic and hierarchical structure in both speech production and speech perception [37, 29, 18, 96]. An attempt to incorporate a primitive level of understanding of this deep speech structure, initiated at the 2009 NIPS Workshop on Deep Learning for Speech Recognition and Related Applications [24], has helped create an impetus in the ASR community to pursue a deep representation learning approach based on the deep neural network (DNN) architecture, which was pioneered by the machine learning community only a few years earlier [54, 53] but rapidly evolved into the new state of the art in speech recognition with industry-wide adoption; e.g., [24, 77, 110, 12, 94, 103, 13, 62, 52, 87, 23, 49, 26, 88, 89, 86, 102, 93]

In the mean time, however, it has been realized by many that the DNN-HMM approach has not modeled speech dynamics properly. This relates to the same type of limitations as we analyzed in Chapter ?? on the topic of HMM where several variants of the HMM were discussed aiming to overcome such limitations. The deep and temporally recurrent neural network (RNN), which is the focus of this chapter, has been developed in the past few years by deep learning and ASR researchers to overcome the dynamic-modeling challenge; e.g., [45, 68, 48, 47, 101, 10, 21, 90, 91, 80, 46, 98]. In the RNN, the internal representation of dynamic speech features is discriminatively formed by feeding the low-level acoustic features into the hidden layer together with the recurrent hidden features from the past history. In contrast, there is no internal representation of speech dynamics in the DNN-HMM. The RNN is a class of neural network models where connections among many of its units form a directed cycle, hence the term *recurrent*. Such a cycle or recurrence is associated with the time-delay operation. The use of time-delayed recurrence over the temporal dimension gives rise to the *memory* structure, expressed as internal states, in the RNN, permitting it to exhibit the type of dynamic temporal behavior not exhibited by the DNN and DNN-HMM discussed in previous chapters.

Even without stacking RNNs one on top of another as carried out in [48, 47, 90, 91] or feeding DNN features into RNNs as explored in [10, 21], an RNN itself is a deep model since temporal unfolding of the RNN creates as many layers in the network as the length of the input speech utterance. Recent progress on speech recognition has seen excellent speech recognition accuracy achieved by RNNs, including the long-short-term-memory (LSTM) version of the RNN which started in as early as 1997 by neural network researchers [55, 41, 42, 47, 46, 105]. One focus of this chapter is to present

the background and mathematical formulation of the RNN (Section 1.2), as well as the learning methods including the most popular Backpropagation Through Time (BPTT) technique (Section 1.3).

The use of RNNs or related neural predictive models for speech recognition dates back to late 1980's and early 1990's; e.g., [104, 84, 22], which achieved relatively low recognition accuracy. Since deep learning became popular in recent years, much more research has been devoted to the RNN, including the applications to both speech [48, 47, 90, 91] and languages [73, 74, 72, 71, 70, 75, 69, 11], and its stacked versions, also called deep RNNs [48, 47, 90, 79, 50]. Most work on RNNs made use of the method of BPTT to train their parameters, and empirical tricks need to be exploited (e.g., truncate gradients when they become too large [72, 71]) in order to make the training effective. It is not until recently that careful analysis was made to fully understand the source of difficulties in learning RNNs and somewhat more principled, but still rather heuristic, solutions were developed. For example, in [6, 80, 3] strategies of gradient norm clipping was proposed to deal with the gradient exploding problem during BPTT training. There are other solutions offered to improve learning methods for the RNN; e.g., [58, 10]. The method described in [10] is based on more principled optimization techniques than most other methods, and will be reviewed in Section 1.4. Further, the LSTM version of the RNN has recently been shown to perform extremely well in both small and large scale ASR, and its structure is well motivated. We will devote Section 1.5 to this topic.

It is important to note that before the recent rise of deep learning for speech modeling and recognition, a number of earlier attempts had been made, which we briefly discussed in Section 7 of Chapter **??** on HMM variants, to develop computational architectures that are "deeper" than the conventional GMM-HMM architecture. One prominent class of such models are hidden dynamic models where the internal representation of dynamic speech features is generated probabilistically from the higher levels in the overall deep speech model hierarchy [30, 15, 82, 9, 64, 99, 17, 109, 35, 107]. Despite separate developments of the RNNs and of the hidden dynamic or trajectory models, they share a very similar motivation - representing aspects of dynamic structure in human speech. Nevertheless, a number of different ways in which these two types of deep dynamic models are constructed endow them with distinct pros and cons. Careful analysis of the contrast between these two model types and of the similarity to each other will help provide insights into the strategies for developing new types of deep dynamic models with the hidden representations of speech features superior to both existing RNNs and hidden dynamic models. We will devote Section 1.6 of this chapter to a contrastive analysis between (discriminative) RNNs and (generative) hidden dynamic models. In this multi-faceted analysis, we will focus mainly on the most prominent contrasts between the two types of dynamic models in terms of the opposing top-down versus bottom-up information flow, and in terms of

the opposing local versus distributed representations adopted by the latent vectors in these two types of models.

1.2 State-Space Formulation of the Basic Recurrent Neural Network

An RNN is fundamentally different from the feed-forward DNN in that the RNN operates not only based on inputs, as for the DNN, but also on internal-states. The internal states encode the past information in the temporal sequence that has already been processed by the RNN. In this sense, the RNN is a dynamic system, more general than the DNN which performs static input-output transformation. The use of the state space in the RNN enables its representation and learning of sequentially extended dependencies over a long time span, at least in principle.

Let us now formulate the simple one-hidden-layer RNN in terms of the (noise-free) nonlinear state space model commonly used in signal processing. This formulation allows us to later compare the RNN with the same state space formulation of nonlinear dynamic systems used as generative models for speech acoustics. The contrast between the discriminative RNN and the use of the same mathematical model in the generative model will be made to shed light onto why one approach works better than another and how a combination of the two would be desirable.

At each time point t, let$\mathbf{x}_t$ be the $K \times 1$ vector of inputs, $\mathbf{h}_t$ be the $N \times 1$ vector of hidden state values, and $\mathbf{y}_t$ be the $L \times 1$ vector of outputs, the simple one-hidden-layer RNN can be described as

$$\mathbf{h}_t = f(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1}) \tag{1.1}$$

$$\mathbf{y}_t = g(\mathbf{W}_{hy}\mathbf{h}_t), \tag{1.2}$$

where $\mathbf{W}_{hy}$ is the $L \times N$ matrix of weights connecting the N hidden units to the L outputs, $\mathbf{W}_{xh}$ is the $N \times K$ matrix of weights connecting the K inputs to the N hidden units, and $\mathbf{W}_{hh}$ is the $N \times N$ matrix of weights connecting the N hidden units from time $t-1$ to time t, $\mathbf{u}_t = \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1}$ is the $N \times 1$ vector of hidden layer potentials, $\mathbf{v}_t = \mathbf{W}_{hy}\mathbf{h}_t$ is the $L \times 1$ vector of output layer potentials, $f(\mathbf{u}_t)$ is the hidden layer activation function, and $g(\mathbf{v}_t)$ is the output layer activation function. Typical hidden layer activation functions are Sigmoid, tanh, and rectified linear units while the typical output layer activation functions are linear and softmax functions. Eqs. (1.1) and (1.2) are often called the observation and state equations, respectively.

Note that, outputs from previous time frames can also be used to update the state vector, in which case the state equation becomes

$$\mathbf{h}_t = f(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{\mathrm{yh}}\mathbf{y}_{\mathrm{t-1}}), \tag{1.3}$$

where $\mathbf{W}_{yh}$ denotes the weight matrix connecting from output layer to the hidden layer. For simplicity and without loss of generality, we only consider the case without output feedback in this chapter.

1.3 The Backpropagation-Through-Time Learning Algorithm

The standard BPTT method, which was well explained in the tutorial material [58, 7] and in the original paper [85], for learning the weight matrices of an RNN *unfolds* the network in time and propagates error signals backwards through time. It is an extension of the classic backpropagation algorithm for feedforward networks, where the stacked hidden layers for the same training frame, t, are replaced by the T same single hidden layers across time, $t = 1, 2, \ldots, T$.

Referring to Eqs. (1.1) and (1.2), we denote $h_t(j)$ the jth hidden unit where $j = 1, 2, \ldots, N$, $w_{hy}(i, j)$ as the weight connecting the jth hidden unit to the $i - th$ output unit for $i = 1, 2, \ldots, L$ and $j = 1, 2, \ldots, N$.

1.3.1 Objective Function for Minimization

As for the classic backpropagation, we begin by defining the cost function (or training criterion). In this section, we use the sum-square error

$$E = c\sum_{t=1}^{T} \|\mathbf{l}_t - \mathbf{y}_t\|^2 = c\sum_{t=1}^{T}\sum_{j=1}^{L}(l_t(j) - y_t(j))^2 \tag{1.4}$$

between the actual output, $\mathbf{y}_t$, and the target vector, $\mathbf{l}_t$, over all time frames as the cost function, where $l_t(j)$ and $y_t(j)$ are the jth units in the target and output vectors, respectively, and $c = 0.5$ is a conveniently chosen scale factor.

We seek to minimize this cost with respect to the weights using the gradient descent algorithm. For a specific weight, w, in the RNN, the update rule for gradient descent is

$$w^{new} = w - \gamma\frac{\partial E}{\partial w}, \tag{1.5}$$

where γ is the learning rate. To compute the gradient, we define the error terms

$$\delta_t^y(j) = -\frac{\partial E}{\partial v_t(j)}, \quad \delta_t^h(j) = -\frac{\partial E}{\partial u_t(j)} \tag{1.6}$$

as the gradient of the cost with respect to the unit's input potential. The error terms and gradients can be recursively computed as we will explain next.

1.3.2 Recursive Computation of Error Terms

In the error propagation part of the BPTT algorithm, all RNN weights are duplicated spatially for an arbitrary number of time steps. That is, they are tied over time. Therefore the standard backpropagation algorithm for feedforward neural networks needs to be modified by incorporating this tying constraint.

At the final time frame $t = T$, we can calculate the error terms at the output as

$$\delta_T^y(j) = -\frac{\partial E}{\partial y_T(j)}\frac{\partial y_T(j)}{\partial v_T(j)} = (l_T(j) - y_T(j))g^{'}(v_T(j)) \quad \text{for } j = 1, 2, \ldots, L$$

$$\text{or} \qquad \boldsymbol{\delta}_T^y = (\mathbf{l}_T - \mathbf{y}_T) \odot g^{'}(\mathbf{v}_T), \tag{1.7}$$

and that at the hidden layer as

$$\delta_T^h(j) = -\left(\sum_{i=1}^{L} \frac{\partial E}{\partial v_T(i)}\frac{\partial v_T(i)}{\partial h_T(j)}\frac{\partial h_T(j)}{\partial u_T(j)}\right) = \sum_{i=1}^{L} \delta_T^y(i) w_{hy}(i,j) f^{'}(u_T(j)) \quad \text{for } j = 1, 2, \ldots, N$$

$$\text{or} \qquad \delta_T^h = \mathbf{W}_{hy}^T \boldsymbol{\delta}_T^y \odot f^{'}(\mathbf{u}_T) \tag{1.8}$$

where $\odot$ is the element-wise multiplication operator.

For all other time frames, $t = T-1, T-2, \ldots, 1$, we can compute the error terms as

$$\delta_t^y(j) = (l_t(j) - y_t(j))g^{'}(v_t(j)) \quad \text{for } j = 1, 2, \ldots, L$$

$$\text{or} \qquad \boldsymbol{\delta}_t^y = (\mathbf{l}_t - \mathbf{y}_t) \odot g^{'}(\mathbf{v}_t) \tag{1.9}$$

for the output units and

$$\delta_t^h(j) = -\left[\sum_{i=1}^{N} \frac{\partial E}{\partial u_{t+1}(i)} \frac{\partial u_{t+1}(i)}{\partial h_t(j)} + \sum_{i=1}^{L} \frac{\partial E}{\partial v_t(i)} \frac{\partial v_t(i)}{\partial h_t(j)}\right] \frac{\partial h_t(j)}{\partial u_t(j)}$$

$$= \left[\sum_{i=1}^{N} \delta_{t+1}^h(i) w_{hh}(i,j) + \sum_{i=1}^{L} \delta_t^y(i) w_{hy}(i,j)\right] f^{'}(u_t(j)) \quad \text{for } j = 1, 2, \ldots, N$$

$$\text{or} \qquad \boldsymbol{\delta}_t^h = \left[\mathbf{W}_{hh}^T \boldsymbol{\delta}_{t+1}^h + \mathbf{W}_{hy}^T \boldsymbol{\delta}_t^y\right] \odot f^{'}(\mathbf{u}_t) \tag{1.10}$$

for the hidden units, recursively, where the error term$\boldsymbol{\delta}_t^y$ is propagated back from the output layer at time frame t, and $\boldsymbol{\delta}_{t+1}^h$ is propagated back from the hidden layer at time frame $t+1$.

1.3.3 Update of RNN Weights

Given all the error terms and gradients computed above, we can easily update the weights. For the output weight matrices we have

$$w_{hy}^{new}(i,j) = w_{hy}(i,j) - \gamma \sum_{t=1}^{T} \frac{\partial E}{\partial v_t(i)} \frac{\partial v_t(i)}{\partial w_{hy}(i,j)} = w_{hy}(i,j) - \gamma \sum_{t=1}^{T} \delta_t^y(i) h_t(j)$$

$$\text{or} \qquad \mathbf{W}_{hy}^{new} = \mathbf{W}_{hy} + \gamma \sum_{t=1}^{T} \boldsymbol{\delta}_y^t \mathbf{h}_t^T. \tag{1.11}$$

For the input weight matrices we get

$$w_{xh}^{new}(i,j) = w_{xh}(i,j) - \gamma \sum_{t=1}^{T} \frac{\partial E}{\partial u_t(i)} \frac{\partial u_t(i)}{\partial w_{xh}(i,j)} = w_{xh}(i,j) - \gamma \sum_{t=1}^{T} \delta_t^h(i) x_t(j)$$

$$\text{or} \qquad \mathbf{W}_{xh}^{new} = \mathbf{W}_{xh} + \gamma \sum_{t=1}^{T} \boldsymbol{\delta}_h^t \mathbf{x}_t^T. \tag{1.12}$$

For the recurrent weight matrices we have

$$w_{hh}^{new}(i,j) = w_{hh}(i,j) - \gamma \sum_{t=1}^{T} \frac{\partial E}{\partial u_t(i)} \frac{\partial u_t(i)}{\partial w_{hh}(i,j)} = w_{hh}(i,j) - \gamma \sum_{t=1}^{T} \delta_t^h(i) h_{t-1}(j)$$

$$\text{or} \qquad \mathbf{W}_{hh}^{new} = \mathbf{W}_{hh} + \gamma \sum_{t=1}^{T} \boldsymbol{\delta}_h^t \mathbf{h}_{t-1}^T. \tag{1.13}$$

Note that different from the BP algorithm used in the DNN system, here the gradients are summed over all the time frames since the same weight ma-

trices are used across time. Algorithm 1.1 summarizes the BPTT algorithm for the single-hidden-layer RNN described above.

Algorithm 1.1 The Backpropagation Through Time Algorithm for the Single-Hidden Layer RNN with the Sum of Squared Error Cost Function

procedure BPTT($\{\mathbf{x}_t, \mathbf{I_t}\} 1 \leq t \leq T$)
▷ $\mathbf{x}_t$ is the input feature sequence
▷ $\mathbf{I_t}$ is the label sequence
▷ forward computation
  for $t \leftarrow 1; t \leq T; t \leftarrow t+1$ **do**
    $\mathbf{u}_t \leftarrow \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1}$
    $\mathbf{h}_t \leftarrow f(\mathbf{u}_t)$
    $\mathbf{v}_t \leftarrow \mathbf{W}_{hy}\mathbf{h}_t$
    $\mathbf{y}_t \leftarrow g(\mathbf{v}_t)$
  end for
▷ backpropagation through time
  $\boldsymbol{\delta}_T^y \leftarrow (\mathbf{l}_T - \mathbf{y}_T) \odot g^{'}(\mathbf{v}_T)$ ▷ $\odot$: element-wise multiplication
  $\delta_T^h \leftarrow \mathbf{W}_{hy}^T \boldsymbol{\delta}_T^y \odot f^{'}(\mathbf{u}_T)$
  for $t \leftarrow T-1; t \geq 1T; t \leftarrow t-1$ **do**
    $\boldsymbol{\delta}_t^y \leftarrow (\mathbf{l}_t - \mathbf{y}_t) \odot g^{'}(\mathbf{v}_t)$
    $\boldsymbol{\delta}_t^h \leftarrow \left[\mathbf{W}_{hh}^T \boldsymbol{\delta}_{t+1}^h + \mathbf{W}_{hy}^T \boldsymbol{\delta}_t^y\right] \odot f^{'}(\mathbf{u}_t)$ ▷ propagate from $\boldsymbol{\delta}_t^y$ and $\boldsymbol{\delta}_{t+1}^h$
  end for
▷ model update
  $\mathbf{W}_{hy} \leftarrow \mathbf{W}_{hy} + \gamma \sum_{t=1}^{T} \boldsymbol{\delta}_y^t \mathbf{h}_t^T$
  $\mathbf{W}_{hh} \leftarrow \mathbf{W}_{hh} + \gamma \sum_{t=1}^{T} \boldsymbol{\delta}_h^t \mathbf{h}_{t-1}^T$
end procedure

The computational complexity of the BPTT described above can be shown to be $O(M^2)$ per time step where $M = LN + NK + N^2$ is the total number of weight parameters that need to be learned. Compared to the classic feedforward backpropagation, BPTT converges slower due to dependencies between frames, and is more likely to converge to a poor local optimum due to exploding and vanishing gradients [80, 98] and utterance-level (instead of frame-level) randomization. It is far from trivial to achieve good results without much experimentation and tuning. The training speed can be improved if we truncate the past history to no more than the last p time steps.

1.4 A Primal-Dual Technique for Learning Recurrent Neural Networks

1.4.1 Difficulties in Learning RNNs

It is well known that learning RNNs is difficult partly because of the exploding and vanishing gradient problems, as analyzed in [80]. A sufficient condition

for the vanishing gradient problem to occur is

$$\|\mathbf{W}_{hh}\| < d \tag{1.14}$$

where $d = 4$ for sigmoidal hidden units and $d = 1$ for linear units. $\|\mathbf{W}_{hh}\|$ is the L_2-norm (the largest singular value) of the recurrent weight matrix $\mathbf{W}_{hh}$of the RNN. On the other hand, a necessary condition for exploding gradient to occur is

$$\|\mathbf{W}_{hh}\| > d. \tag{1.15}$$

Therefore, the property of recurrent matrix $\mathbf{W}_{hh}$ is essential for learning an RNN. In [80, 6], the proposed method for solving the exploding gradient problem is to empirically clip the gradient so that the norm of the gradient cannot exceed certain threshold. The way to avoid the vanishing gradient is also empirical - either adding a regularization term to push up the gradient or exploiting the information about the curvature of the objective function [68]. Here we review the study described in [10], which proposed and successfully experimented a more rigorous and effective approach to learning RNNs by directly exploiting the constraints that need to be imposed on $\mathbf{W}_{hh}$.

1.4.2 Echo-State Property and Its Sufficient Condition

We now show that conditions described in Eqns. 1.14 and 1.15 are closely related to whether the RNN satisfies the echo-state property, which, following [58], states that "if the network has been run for a very long time, the current network state is uniquely determined by the history of the input and the (teacher-forced) output." It is also shown in [57] that this echo-state property is equivalent to the *state contracting* property. For networks with no feedback from the output, a network is state contracting if for all right-infinite input sequences $\{\mathbf{x}_t\}$, where $t = 0, 1, 2, \ldots$, there exists a null sequence $(\epsilon_t)_{t\geq 0}$ such that for all starting states $\mathbf{h}_0$ and $\mathbf{h}'_0$ and for all $t > 0$ it holds that $\|\mathbf{h}_t - \mathbf{h}'_t\| < \epsilon_t$, where $\mathbf{h}_t$ and $\mathbf{h}'_t$ are two hidden state vectors at time t obtained when the network is driven by $\mathbf{x}_t$ up to time t after having been stated in $\mathbf{x}_0$ and $\mathbf{x}'_0$, respectively. It is further shown that a sufficient condition for the *non-existence*, or a necessary condition for the *existence*, of echo-state property is that the spectral radius of the recurrent matrix $\mathbf{W}_{hh}$ is greater than one when tanh nonlinear units are used in the RNN's hidden layer.

In echo-state machines, the reservoir or recurrent weight matrix $\mathbf{W}_{hh}$ is randomly generated and normalized according to the rule above and will remain unchanged over time in the training. The input weight matrix $\mathbf{W}_{xh}$ is fixed as well. To improve the learning, we here learn both $\mathbf{W}_{hh}$ and $\mathbf{W}_{xh}$ subject to the constraint that the RNN satisfies the echo-state property. To this end, the following sufficient condition for the echo-state property was

recently developed in [10], which can be more easily handled in the training procedure than the original definition:

Let $d = 1/\max_x |f'(x)|$. Then the RNN satisfies the echo-state property if

$$\|\mathbf{W}_{hh}\|_\infty < d \tag{1.16}$$

where $\|\mathbf{W}_{hh}\|_\infty$denote the ∞-norm of matrix $\mathbf{W}_{hh}$ (i.e., maximum absolute row sum), $d = 1$ for tanh units, and $d = 4$ for sigmoid units.

An important consequence of condition 1.16 is that it naturally avoids the exploding gradient problem. If the condition 1.16can be enforced in the training process, there is no need to clip the gradient in a heuristic way.

1.4.3 Learning RNNs as a Constrained Optimization Problem

Given the sufficient condition for the echo-state property, we can now formulate the problem of learning the RNN that preserves the echo-state property as the following constrained optimization problem:

$$\min_{\Theta} \quad E(\Theta) = E(\mathbf{W}_{hh,}\mathbf{W}_{xh}, \mathbf{W}_{hy}) \tag{1.17}$$

$$\text{subject to} \quad \|\mathbf{W}_{hh}\|_\infty \leq d \tag{1.18}$$

That is, we need to find the set of RNN parameters that best predict the target values on average while preserving the echo-state property. Recall that $\|\mathbf{W}_{hh}\|_\infty$ is defined as the maximum absolute row sum. Therefore, the above RNN learning problem is equivalent to the following constrained optimization problem:

$$\min_{\Theta} \quad E(\Theta) = E(\mathbf{W}_{hh}, \mathbf{W}_{xh}, \mathbf{W}_{hy}) \tag{1.19}$$

$$\text{subjec to} \quad \sum_{j=1}^{N} |W_{ij}| \leq d, \quad i = 1, \ldots, N \tag{1.20}$$

where W_{ij} denotes the (i, j)-th entry of the matrix $\mathbf{W}_{hh}$. Next, we proceed to derive the learning algorithm that can achieve this objective.

1.4.4 A Primal-Dual Method for Learning RNNs

A brief introduction to primal-dual method

Here let us solve the constrained optimization problem above by the primal-dual method, a popular technique in modern optimization literature; e.g., [8]. First, the Lagrangian of the problem can be written as

$$L(\Theta, \boldsymbol{\lambda}) = E(\mathbf{W}_{hh}, \mathbf{W}_{xh}, \mathbf{W}_{hy}) + \sum_{i=1}^{N} \lambda_i \left(\sum_{j=1}^{N} |W_{ij}| - d \right) \tag{1.21}$$

where λ_i denotes the ith entry of the Lagrange vector $\boldsymbol{\lambda}$ (i.e., dual variable) and is required to be non-negative. Let the dual function $q(\boldsymbol{\lambda})$ be defined as the following *unconstrained* optimization problem

$$q(\boldsymbol{\lambda}) = \min_{\boldsymbol{\Theta}} L(\Theta, \boldsymbol{\lambda}) \tag{1.22}$$

The dual function $q(\boldsymbol{\lambda})$ in the above unconstrained optimization problem is always concave, even when the original cost $E(\Theta)$ is non-convex [8]. In addition, the dual function is always a lower bound of the original constrained optimization problem. That is,

$$q(\boldsymbol{\lambda}) \leq E(\Theta^{\star}) \tag{1.23}$$

Maximizing $q(\boldsymbol{\lambda})$ subject to the constraint $\lambda_i \geq 0, i = 1, \ldots, N$ will be the best lower bound that can be obtained from the dual function [8]. This new problem is called the dual problem of the original optimization problem:

$$\max_{\boldsymbol{\lambda}} \quad q(\boldsymbol{\lambda}) \tag{1.24}$$

$$\text{subjec to} \quad \lambda_i \geq 0, \quad i = 1, \ldots, N \tag{1.25}$$

which is a convex optimization problem since we are maximizing a concave objective with linear inequality constraints. After solving $\boldsymbol{\lambda}^{\star}$ in Eqns. 1.24 and 1.21, we can substitute the corresponding $\boldsymbol{\lambda}^{\star}$ into the Lagrangian 1.21 and then solve the corresponding set of parameters $\Theta^o = \{\mathbf{W}_{hh}^0, \mathbf{W}_{xh}^0, \mathbf{W}_{hy}^0\}$ that minimizes $L(\Theta, \boldsymbol{\lambda})$ for this given $\boldsymbol{\lambda}^{\star}$:

$$\Theta^o = \arg\min_{\Theta} L(\Theta, \boldsymbol{\lambda}^{\star}) \tag{1.26}$$

Then, the obtained $\Theta^o = \{\mathbf{W}_{hh}^0, \mathbf{W}_{xh}^0, \mathbf{W}_{hy}^0\}$ will be an approximation to an optimal solution to the original constrained optimization problem. For convex optimization problems, this approximate solution will be the same global optimal solution under some mild conditions [8]. This property is called

strong duality. However, in general non-convex problems, it will not be the exact solution. But since finding the globally optimal solution to the original problem of 1.24 and 1.21 is not realistic, it would be satisfactory if it can provide a good approximation.

Now let us return to the problem of 1.24 and 1.21. We are indeed solving the following problem

$$\max_{\boldsymbol{\lambda} \succeq \mathbf{0}} \min_{\Theta} L(\Theta, \boldsymbol{\lambda}) \tag{1.27}$$

where the notation $\boldsymbol{\lambda} \succeq \mathbf{0}$ denotes that each entry of the vector $\boldsymbol{\lambda}$ is greater than or equal to zero. The preceding analysis shows that in order to solve the problem, we need to first minimize the Lagrangian $L(\Theta, \boldsymbol{\lambda})$ with respect to Θ, while in the mean time, maximize the dual variable $\boldsymbol{\lambda}$ subjected to the constraint that $\boldsymbol{\lambda} \succeq \mathbf{0}$. Therefore, as we will proceed next, updating the RNN parameters consists of two steps:

- *primal update* - minimization of $L(\Theta, \boldsymbol{\lambda}^*)$ with respect to Θ, and
- *dual update* - maximization of $L(\Theta^*, \boldsymbol{\lambda})$ with respect to $\boldsymbol{\lambda}$.

Primal-dual method applied to RNN learning: Primal update

We may directly apply the gradient descent algorithm to the primal update rule to minimize the Lagrangian $L(\Theta, \boldsymbol{\lambda}^*)$ with respect to Θ. However, it is better to exploit the structure in this objective function, which consists of two parts: $E(\Theta)$ that measures the prediction quality, and the part that penalizes the violation of the constraint expressed in 1.25. The latter part is a sum of many ℓ_1 regularization terms on the rows of the matrix $\mathbf{W}_{hh}$:

$$\sum_{j=1}^{N} |W_{ij}| = \|\mathbf{w}_i\|_1 \tag{1.28}$$

where $\mathbf{w}_i$ denotes the ith row vector of the matrix $\mathbf{W}_{hh}$. With this observation, the Lagrangian in 1.21 can be written in the equivalent form of

$$L(\Theta, \boldsymbol{\lambda}) = E(\mathbf{W}_{hh}, \mathbf{W}_{xh}, \mathbf{W}_{hy}) + \sum_{i=1}^{N} \lambda_i \left(\|\mathbf{w}_i\|_1 - d \right). \tag{1.29}$$

In order to minimize $L(\Theta, \boldsymbol{\lambda})$ with the structure above with respect to $\Theta = \{\mathbf{W}_{hh}, \mathbf{W}_{xh}, \mathbf{W}_{hy}\}$, we can use a technique similar to the one proposed in [2] to derive the following iterative soft-thresholding algorithm for the primal update of $\mathbf{W}_{hh}$:

$$\mathbf{W}_{hh}^{\{k\}} = S_{\boldsymbol{\lambda}\mu_k} \left\{ \mathbf{W}_{hh}^{\{k-1\}} - \mu_k \frac{\partial E(\mathbf{W}_{hh}^{\{k-1\}}, \mathbf{W}_{xh}^{\{k-1\}}, \mathbf{W}_{hy}^{\{k-1\}})}{\partial \mathbf{W}_{hh}} \right\}, \tag{1.30}$$

where $S_{\boldsymbol{\lambda}\mu_k}(\mathbf{X})$ denote a component-wise shrinkage (soft-thresholding) operator on a matrix $\mathbf{X}$, defined as

$$[S_{\boldsymbol{\lambda}\mu_k}(\mathbf{X})]_{ij} = \begin{cases} X_{ij} - \lambda_i\mu_k & \mathbf{X}_{ij} \geq \lambda_i\mu_k \\ \mathbf{X}_{ij} + \lambda_i\mu_k & \mathbf{X}_{ij} \leq -\lambda_i\mu_k \\ 0 & \text{otherwise} \end{cases} \tag{1.31}$$

The above primal update 1.30 for $\mathbf{W}_{hh}$ is implemented by a standard stochastic gradient descent followed by a shrinkage operator. On the other hand, the primal updates for $\mathbf{W}_{xh}$ and $\mathbf{W}_{hy}$ will follow the standard stochastic gradient descent rule since there is no constraints on them. In order to accelerate the convergence of the algorithm, one can, for example, add momentum or use Nesterov method to replace the gradient descent steps, as was carried out in the experiments reported in [10, 21].

Primal-dual method applied to RNN learning: Dual update

The dual update step is aimed to maximize $L(\Theta, \boldsymbol{\lambda})$ with respect to $\boldsymbol{\lambda}$ subject to the constraint that $\boldsymbol{\lambda} \succeq \mathbf{0}$. To this end, we use the following rule of gradient ascent with projection, which increases the function value of $L(\Theta, \boldsymbol{\lambda})$ while enforcing the constraint:

$$\lambda_{i,k} = [\lambda_{i,k-1} + \mu_k \left(\|\mathbf{w}_{i,k-1}\|_1 - d\right)]_+ \tag{1.32}$$

where $[x]_+ = \max\{0, x\}$. Note that λ_i is a regularization factor in $L(\Theta, \boldsymbol{\lambda})$ that penalizes the violation of constraint for the ith row of $\mathbf{W}_{hh}$. The dual update can be interpreted as a rule to adjust the regularization factor in an adaptive manner. When the sum of the absolute values of the ith row of $\mathbf{W}_{hh}$ exceeds d, i.e., violating the constraint, the recursion 1.32 will increase the regularization factor $\lambda_{i,k}$ on the ith row in 1.21. On the other hand, if the constraint for a certain i is not violated, then the dual update 1.21 will decrease the value of the corresponding λ_i. The projection operator $[x]_+$ makes sure that once the regularization factor λ_i is decreased below zero, it will be set to zero and the constraint for the ith row in 1.25 will not be penalized in 1.21.

1.5 Recurrent Neural Networks Incorporating LSTM Cells

1.5.1 Motivations and Applications

The basic RNN described so far does not have sufficient structure in modeling sophisticated temporal dynamics, and thus in practice has been shown not capable of looking far back into the past in many types of input sequences. These problems were first carefully analyzed in the early work published in [55], and subsequently in [41, 42, 46] One solution is to impose a *memory* structure into the RNN, resulting in the so-called *long short-term memory* cells in the RNN proposed first in the above early publications. Such LSTM version of the RNN was shown to overcome some fundamental problems of traditional RNNs, and be able to efficiently learn to solve a number of previously un-learnable tasks involving recognition of temporally extended patterns in noisy input sequences and of the temporal order of widely separated events in noisy input streams. Like the basic RNNs described so far in this chapter, the LSTM version can be shown to be an universal computing machine in the sense that given enough network units it can compute anything a conventional computer can compute and if it has proper weight matrices. But unlike the basic RNNs, the LSTM version is better-suited to learn from input sequence data to classify, process and predict time series when there are very long time lags of unknown lengths between important events.

Over the years since the early establishment of the LSTM-RNN, it has been shown to perform very well in many useful tasks such as handwriting recognition, phone recognition, keyword spotting, reinforcement learning for robot localization and control (in partially observable environments), online learning for protein structure prediction, learning music composition, and learning of grammars, etc. An overview of these progresses has been provided in [92]. Most recently, excellent results on large vocabulary speech recognition by the use of the LSTM-RNN have been reported in [90, 91]. A simpler version of the LSTM-RNN is also recently found effective for language identification [44], speech synthesis [39, 38], and for environment-robust speech recognition [40, 105]. In the latter study, it was shown that the LSTM architecture can be effectively used to exploit temporal context in learning the correspondences of noise-distorted and reverberant speech features and effectively handle highly non-stationary, convolutive noise involved in the task of 2013 Computational Hearing in Multisource Environments (CHiME) Challenge (track 2).

1.5.2 The Architecture of LSTM Cells

The basic idea behind the LSTM cell in the RNN is to use various types of gating (i.e., element-wise multiplication) structure to control the information flow in the network. An LSTM-RNN is an advanced version of the RNN that contains LSTM cells instead of, or in addition to, regular network units. An LSTM cell can be regarded as a complex and smart network unit capable of remembering information for a long length of time. This is accomplished by the gating structure that determines when the input is significant enough to remember, when it should continue to remember or forget the information, and when it should output the information.

Mathematically, a set of LSTM cells can be described by the following forward operations iteratively over time $t = 1, 2, ..., T$, following [55, 46, 90, 41, 42]:

$$\mathbf{i}_t = \sigma\left(\mathbf{W}^{(xi)}\mathbf{x}_t + \mathbf{W}^{(hi)}\mathbf{h}_{t-1} + \mathbf{W}^{(ci)}\mathbf{c}_{t-1} + \mathbf{b}^{(i)}\right) \tag{1.33}$$

$$\mathbf{f}_t = \sigma\left(\mathbf{W}^{(xf)}\mathbf{x}_t + \mathbf{W}^{(hf)}\mathbf{h}_{t-1} + \mathbf{W}^{(cf)}\mathbf{c}_{t-1} + \mathbf{b}^{(f)}\right) \tag{1.34}$$

$$\mathbf{c}_t = \mathbf{f}_t \bullet \mathbf{c}_{t-1} + \mathbf{i}_t \bullet \tanh\left(\mathbf{W}^{(xc)}\mathbf{x}_t + \mathbf{W}^{(hc)}\mathbf{h}_{t-1} + \mathbf{b}^{(c)}\right) \tag{1.35}$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}^{(xo)}\mathbf{x}_t + \mathbf{W}^{(ho)}\mathbf{h}_{t-1} + \mathbf{W}^{(co)}\mathbf{c}_t + \mathbf{b}^{(o)}\right) \tag{1.36}$$

$$\mathbf{h}_t = \mathbf{o}_t \bullet \tanh\left(\mathbf{c}_t\right), \tag{1.37}$$

where $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{c}_t$, $\mathbf{o}_t$, and $\mathbf{h}_t$ are vectors, all with the same dimensionality, which represent five different types of information at time t of the input gate, forget gate, cell activation, output gate, and hidden layer, respectively. $\sigma(.)$ is the logistic sigmoid function, $\mathbf{W}$'s are the weight matrices connecting different gates, and $\mathbf{b}$'s are the corresponding bias vectors. All the weight matrices are full except the weight matrix $\mathbf{W}^{(ci)}$ is diagonal. Note functionally, the above LSTM set, with input vector $\mathbf{i}_t$ and hidden vector $\mathbf{h}_t$ is akin to the input and hidden layers of the basic RNN as described by Eq. 1.1. An additional output layer need to be provided (not included in the above set of equations) on top of the LSTM-RNN's hidden layer. In [48], a straightforward linear mapping from LSTM-RNN's hidden layer to the output layer was exploited. In [90], two intermediate, linear projection layers were created to reduce the large dimensionality of the LSTM-RNN hidden layer's vector$\mathbf{h}_t$. They were then linearly combined to form the final output layer.

1.5.3 Training the LSTM-RNN

All LSTM-RNN parameters can be learned in a similar way to learning the basic RNN parameters using the BPTT as described in Section 3. That is, to minimize LSTM's total loss on a set of training sequences, stochastic gradient descent methods can be used to update each weight parameter where the error derivative with respect to the weight parameter can be computed by the BPTT. As we recall, one major difficulty associated with the BPTT for the basic RNN is that error gradients vanish exponentially quickly with the size of the time lag between important events, or the gradients explode equally fast, both making the learning ineffective unless heuristic rules are applied or principled constrained optimization is exploited as described in Section 1.4. With the use of LSTM cells that replaces the basic RNN's input-to-hidden layer mapping and hidden-to-hidden layer transition, this difficulty is largely overcome. The reason is that when error signals are back-propagated from the output, they become trapped in the memory portion of the LSTM cell. Therefore, meaningful error signals are continuously fed back to each of the gates until the RNN parameters are well trained. This makes the BPTT effective for training LSTM cells, which can remember information in the input sequence for a long time when such pattern is present in the input data and is needed to perform the required sequence-processing task.

Of course, due to the greater structural complexity of the LSTM cells than its counterpart in the basic RNN where the two nonlinear mappings (i.e., input-to-hidden and hidden-to-hidden) typically have no designed structure, gradient computation as required in the BPTT is expectedly more involved than that for the basic RNN. We refer readers to Chapter **??** on computational networks in this book for additional discussions on this topic.

1.6 Analyzing Recurrent Neural Networks - A Contrastive Approach

We devote this section to the important topic of analyzing the capabilities and limitations of the RNN, which is mathematically formulated as a state-space dynamic system model described in Section 1.2. We take a contrastive approach to the analysis by examining the similarities to and differences from the closely related state-space formulation of the hidden dynamic model (HDM) described in Section 3.7 of Chapter **??** on the HMM variants. The main goal of performing such a comparative analysis is to understand the respective strengths and weaknesses of these two types of speech models derived from quite distinct motivations yet sharing striking commonality in their mathematical formulation of the models. Based on this understanding, it is possible to construct potentially more effective RNN-like dynamic archi-

tectures as well as new learning paradigms to further advance the state of the art in acoustic modeling for ASR.

1.6.1 Direction of Information Flow: Top-Down or Bottom-Up

The first aspect examined in our contrastive analysis between the basic RNN model and the HDM is the opposing direction in which information flows in these two types of models. In the HDM, the speech object is modeled by a generative process from the top linguistic symbol or label sequence to the bottom continuous-valued acoustic observation sequence, mediated by the intermediate latent dynamic process which is also continuous valued. That is, this top-down generative process starts with specification of the latent linguistic sequence at the top level. Then the label sequence generates the latent dynamic vector sequence, which in turn generates the visible acoustic sequence at the bottom level in the model hierarchy. In contrast, in bottom-up modeling paradigm as adopted by the RNN, information flow starts at the bottom level of acoustic observation, which activates the hidden layer of the RNN modeling temporal dynamics via the recurrent matrix. Then the output layer of the RNN computes the linguistic label or target sequence as a numerical-vectorial sequence at the top level of the model hierarchy. Since the top layer determines the speech-class distinction, this bottom-up processing approach adopted by the RNN can be appropriately called (direct) discriminative learning. See more detailed discussions on discriminative learning versus generative learning in [27]. Let us now further elaborate on the top-down versus bottom-up comparisons as connected to generative versus discriminative learning here.

The hidden dynamic model as a top-down generative process

To facilitate the comparison with the RNN, let us rewrite the HDM state equation ?? and observation equation ?? both in Section 3.7.2 of Chapter ??, into the following form most compatible with the state-space formulation of the basic RNN described in Section 1.2 of this chapter:

$$\mathbf{h}_t = \boldsymbol{q}(\mathbf{h}_{t-1}; \mathbf{W}_{l_t}, \boldsymbol{\Lambda}_{l_t}) + \text{StateNoiseTerm} \tag{1.38}$$

$$\mathbf{x}_t = \boldsymbol{r}(\mathbf{h}_t, \boldsymbol{\Omega}_{l_t}) + \text{ObsNoiseTerm} \tag{1.39}$$

where$\mathbf{W}_{l_t}$ is the system matrix that shapes the (articulatory-like) state dynamics, which can be easily structured to follow physical constraint in speech production; e.g., [16, 15]. The parameter set$\boldsymbol{\Lambda}_{l_t}$ include those of"phonetic tar-

gets as correlates to the components of the phonological units (e.g., symbolic articulatory features), which can also be interpreted as the input driving force derived from speech production's motor control to the articulatory state dynamics. Both sets of parameters, $\mathbf{W}_{l_t}$ and $\boldsymbol{\Lambda}_{l_t}$, are dependent on the label l_t at time t with segmental properties; hence the model is also called a (segmental) switching dynamic system. The system matrix $\mathbf{W}_{l_t}$is analogous to $\mathbf{W}_{hh}$ in the RNN. The parameter set $\boldsymbol{\Omega}_{l_t}$, on the other hand, governs the nonlinear mapping from the hidden (articulatory-like) states in speech production to continuous-valued acoustic features $\mathbf{x}_t$, which is the output of the HDM, on a frame-by-frame basis. In some early implementations, $\boldsymbol{\Omega}_{l_t}$ took the form of shallow neural network weights [28, 82, 9, 99, 100]. In another implementation, $\boldsymbol{\Omega}_{l_t}$ took the form of a set of matrices in a mixture of linear experts [65, 67].

The state equation in several previous implementations of the HDM of speech did not take nonlinear forms. Rather, the following linear form was used (e.g., [28, 65, 67]):

$$\mathbf{h}_t = \mathbf{W}_{hh}(l_t)\mathbf{h}_{t-1} + [\mathbf{I} - \mathbf{W}_{hh}(l_t)]\mathbf{t}_{l_t} + \text{StateNoiseTerm} \tag{1.40}$$

which exhibits the target-directed property for articulatory-like dynamics. Here, the parameters $\mathbf{W}_{hh}$ are a function of the (phonetic) label l_t at a particular time frame t, and $\mathbf{t}_{l_t}$ is a mapping from the symbolic quantity l_t of a linguistic unit to the continuous-valued target vector with the segmental property. To make the comparisons easy with the RNN, let us keep the nonlinear form and remove both the state and observation noise, yielding the state-space generative model of

$$\mathbf{h}_t = \boldsymbol{q}(\mathbf{h}_{t-1}; \mathbf{W}_{l_t}, \mathbf{t}_{l_t}) \tag{1.41}$$

$$\mathbf{x}_t = \boldsymbol{r}(\mathbf{h}_t, \boldsymbol{\Omega}_{l_t}) \tag{1.42}$$

The recurrent neural net as a bottom-up discriminative classifier

Likewise, to facilitate the comparison with the HDM, let us rewrite the RNN's state and observation equations 1.1 and 1.2 into a more general form:

$$\mathbf{h}_t = \boldsymbol{f}(\mathbf{h}_{t-1}; \mathbf{W}_{hh}, \mathbf{W}_{xh,}, \mathbf{x}_t) \tag{1.43}$$

$$\mathbf{y}_t = \boldsymbol{g}(\mathbf{h}_t; \mathbf{W}_{hy}) \tag{1.44}$$

where information flow starts from observation data $\mathbf{x}_t$, going to hidden vectors $\mathbf{h}_t$, and then going to the predicted target label vectors $\mathbf{y}_t$, often coded in a one-hot manner, in the bottom-up direction.

This contrasts with the HDM's corresponding state and observation equations 1.41 and 1.42, which describe the information flow from the top-level label-indexed phonetic target vector $\mathbf{t}_{l_t}$ to hidden vectors $\mathbf{h}_t$ and then to observation data $\mathbf{x}_t$, where we clearly see top-down information flows, opposite to the RNN's bottom-up information flow.

To further examine other differences between the HDM and the RNN beyond the top-down versus bottom-up difference identified above, we can keep the same mathematical description of the RNN but exchange the variables of input $\mathbf{x}_t$ and output $\mathbf{y}_t$ in Eqns. 1.43 and 1.44. This yields

$$\mathbf{h}_t = \boldsymbol{f}(\mathbf{h}_{t-1}; \mathbf{W}_{hh}, \mathbf{W}_{yh}, \mathbf{y}_t) \tag{1.45}$$

$$\mathbf{x}_t = \boldsymbol{g}(\mathbf{h}_t; \mathbf{W}_{hx}). \tag{1.46}$$

After normalizing the RNN into the generative form of Eqs. 1.45 and 1.46 via input-output exchange, with the same direction of information flow as the HDM, we will analyze below the remaining contrasts between the RNN and HDM with respect to the different nature of the hidden-space representations (while keeping the same generative form of the models). We will then analyze other aspects of the contrast between them including different ways of exploiting model parameters.

1.6.2 The Nature of Representations: Localist or Distributed

Localist and distributed representations are important concepts in cognitive science as two distinct styles of information representation. In the localist representation, each neuron represents a single concept on a stand-alone basis. That is, localist units have their own meaning and interpretation, not so for the units in distributed representation. The latter pertains to an internal representation of concepts in such a way that they are modeled as being explained by the interactions of many hidden factors. A particular factor learned from configurations of other factors can often generalize well to new configurations, not so in localist representations.

Distributed representations, based on vectors consisting of many non-zero elements or units, naturally occur in a "connectionist" neural network, where a concept is represented by a pattern of activity across a number of many units and where at the same time a unit typically contributes to many concepts. One key advantage of such many-to-many correspondence is that they provide robustness in representing the internal structure of the data in terms of graceful degradation and damage resistance. Such robustness is enabled by redundant storage of information. Another advantage is that they facilitate automatic generalization of concepts and relations, thus enabling rea-

soning abilities. Further, distributed representations allow similar vectors to be associated with similar concepts and thus allow efficient use of representational resources. These attractive properties of distributed representations, however, come with a set of weaknesses — non-obviousness in interpreting the representations, difficulties with representing hierarchical structure, and inconvenience in representing variable-length sequences. Distributed representations are also not directly suitable for input and output to a network and some translation with localist representations are needed.

Local representations, on the other hand, have advantages of explicitness and ease of use; i.e., the explicit representation of the components of a task is simple and the design of representational schemes for structured objects is easy. But the weaknesses are many, including inefficiency for large sets of objects, highly redundant use of connections, and undesirable growth of units in networks which represent complex structure.

The HDM discussed above adopts "localist" representations for the symbolic linguistic units, and the RNN makes use of distributed representations. This can be seen directly from Eq. 1.41 for the HDM and from Eq. 1.45 for the RNN. In the former, symbolic linguistic units l_t as a function of time t are coded implicitly in a stand-alone fashion. The connection of symbolic linguistic units to continuous-valued vectors is made via a one-to-one mapping, denoted by $\mathbf{t}_{l_t}$ in Eq. 1.41, to the hidden dynamic's asymptotic "targets" denoted by vector $\mathbf{t}$. This type of mapping is common in phonetic-oriented phonology literature, and is called the "interface between phonology and phonetics" in a functional computational model of speech production [15]. Further, the HDM uses the linguistic labels that are represented in a localist manner to index separate sets of time-varying parameters $\mathbf{W}_{l_t}$ and $\boldsymbol{\Omega}_{l_t}$, leading to "switching" dynamics which considerably complicates the decoding computation. This kind of parameter specification isolates the parameter interactions across different linguistic labels, gaining the advantage of explicit interpretation of the model but losing on direct discrimination across linguistic labels.

In contrast, in the state equation of the RNN model shown in Eq. 1.45, the symbolic linguistic units are directly represented as vectors of $\mathbf{y}_t$ (one-hot or otherwise) as a function of time t. No mapping to separate continuous-valued "phonetic" vectors are needed. Even if the one-hot coding of $\mathbf{y}_t$ vectors is localist, the hidden state vector $\mathbf{h}$ provides a distributed representation and thus allows the model to store a lot of information about the past in a highly efficient manner. Importantly, there is no longer a notion of label-specific parameter sets of $\mathbf{W}_{l_t}$ and $\boldsymbol{\Omega}_{l_t}$ as in the HDM. The weight parameters in the RNN are shared across all linguistic label classes. This enables direct discriminative learning for the RNN. In addition, the distributed representation used by the hidden layer of the RNN allows efficient and redundant storage of information, and has the capacity to automatically disentangle variation factors embedded in the data. However, as inherent in distributed representations discussed earlier, the RNN also carries with them the difficulty of

interpreting the parameters and hidden states, and the difficulties of modeling structure and of exploiting explicit knowledge of articulatory dynamics into the model which we discuss next.

1.6.3 Interpretability: Inferring Latent Layers or End-to-End Learning

An obvious strength of the localist representation as adopted by the HDM for modeling deep speech structure is that the model parameters and the latent state variables are explainable and easy to diagnose. In fact, one main motivation of many of such models is that the knowledge of hierarchical structure in speech production in terms of articulatory and vocal tract resonance dynamics can be directly (but approximately with a clear sense of the degree of approximation) incorporated into the design of the models [30, 15, 9, 82, 107, 25, 78]. Practical benefits of using interpretable, localist representation of hidden state vectors include sensible ways of initializing the parameters to be learned — e.g., with extracted formants to initialize hidden variables composed of vocal tract resonances. Another obvious benefit is the ease with which one can diagnose and analyze errors during model implementation via examination of the inferred latent variables. Since localist representations, unlike their distributed counterpart, do not superimpose patterns for signaling the presence of different linguistic labels, the hidden state variables not only are explanatory but also unambiguous. Further, the interpretable nature of the models allows complex causal and structured relationships to be built into them, free from the difficulty of doing so that is associated with distributed representations. In fact, several versions of the HDM has been constructed with many layers in the hierarchical latent space, all with clear physical embodiment in human speech production; e.g., Chapter 2 in [18]. However, the complex structure makes it very difficult to do discriminative parameter learning. As a result, nearly all versions of HDMs have adopted maximum-likelihood learning or data fitting approaches. For example, the use of linear or nonlinear Kalman filtering (E step of the EM algorithm) for learning the parameters in the generative state-space models has been applied to only maximum likelihood estimates [95, 99].

In contrast, the learning algorithm of BPTT commonly used for end-to-end training of the RNN with distributed representations for the hidden states performs discriminative training by directly minimizing linguistic label prediction errors. It is straightforward to do so in the formulation of the learning objective because each element in the hidden state vector contributes to all linguistic labels due to the very nature of the distributed representation. It is very unnatural and difficult to do so in the generative HDM based on localist representations of the hidden states, where each state and the associated model parameters typically contribute to only one particular linguistic unit,

which is used to index the set of model parameters in most generative models including HDMs and HMMs.

1.6.4 Parameterization: Parsimonious Conditionals or Massive Weight Matrices

The next aspect of comparisons between the HDM and the RNN concerns very different ways to parameterize the HDM and the RNN — Characterization of conditional distributions using parsimonious sets of parameters in the conditionals or representing the complex mapping using largely non-structured massive weight matrices. Due to the interpretable latent states in the HDM as well as the parameters associated with them, speech knowledge can be used in the design of the model, leaving the size of free parameters to be relatively small. For example, when vocal tract resonance vectors are used to represent the hidden dynamics, a dimension of eight appears to be sufficient to capture the prominent dynamic properties responsible for generating the observed acoustic feature sequences. Somewhat higher dimensionality, about 13, is needed with the use of the hidden dynamic vectors associated with the articulators' configuration in speech production. The use of such parsimonious parameter sets to characterize conditional distributions in the HDM, a special of the dynamic Bayesian network, which is sometimes called "small is good", is also facilitated by the localist representation of hidden state components and the related parameters that are connected or indexed to a specific linguistic unit. This contrasts with the distributed representation adopted by the RNN where both the hidden state vector elements and the connecting weights are shared across all linguistic units, thereby demanding many folds more model parameters.

The ability to use speech-domain knowledge to construct generative models with a parsimonious set of parameters is both a blessing and a curse. On the one hand, such knowledge has been usefully exploited to constrain the target-directed and smooth (i.e., non-oscillatory) hidden dynamics within each phone segment [65, 66], to represent the analytical relationship between the latent vocal tract resonance vector (both resonance frequencies and bandwidths) and cepstral coefficients [1, 20, 35], and to model both anticipatory and regressive types of coarticulation expressed in the latent space as a result of hidden dynamics [15, 34]. With the right prediction of time-varying trajectories in the hidden space and then causally in the observed acoustic space, powerful constraints can be placed in the model formulation to reduce over-generation in the model space and to avoid unnecessarily large model capacity. On the other hand, the use of speech knowledge limits the growth of the model size as more data are made available in training. For example, when the dimensionality of the vocal tract resonance vectors goes beyond eight, many advantages of interpretable hidden vectors no longer hold. Since

speech knowledge is necessarily incomplete, the constraints imposed on the model space may be outweighed by the opportunity lost with increasingly large amounts of training data and by the incomplete knowledge.

In contrast, the RNN usually do not use any speech knowledge to constrain the model space due to the inherent difficulty of interpreting the ambiguous hidden state represented in a distributed manner. As such, the RNN in principle has the freedom to use increasingly larger parameters in keeping with the growing size of the training data. Lack of constraints may cause the model to over-generalize. This, together with the known difficulties of the various learning algorithms for the RNN as analyzed in [5, 80], has limited the progress of using RNNs in speech recognition for many years until recently. More recent progresses of applying RNNs to speech recognition have involved various methods of introducing constraints either in the model construction or in the implementation of learning algorithms. For example, in the studies reported in [48, 47, 90, 91], the RNN's hidden layers are designed with memory units based on a very clever LSTM structure. While strongly constraining possible variations of hidden layer activities, the LSTM-RNN nevertheless allows massive weight parameters to be used by simply increasing the total number of the memory units as well as the complexity in the LSTM cells. Separately, the RNN can also be constrained during the learning stage, where the size of the gradient computed by BPTT is limited by a threshold to avoid explosion as reported in [72, 5] or where the range of the permissible RNN parameters are constrained to be within what the "echo-state property" would allow as explored in[10] and we reviewed in Section 4 in this chapter.

The different ways of parameterizing the HDM and the RNN also lead to distinct styles of computation associated with both training and run-time of the two models. In particular, the computation of the RNN is structured to be highly regular, large matrix multiplications. This is well matched with the capability of GPUs, which is designed for commodity high-performance supercomputing with hardware optimizability. This computational advantage of the RNN and other neural network based deep learning methods, unfortunately, is lacking in the HDM and in most other deep generative models.

1.6.5 Methods of Model Learning: Variational Inference or Gradient Descent

The final aspect of contrastive analysis between the HDM and the RNN is on the very different methods in learning model parameters. [1]

The HDM is a deep and directed generative model, also known as deep, dynamic Bayesian network or belief network, with heavily loopy structure

[1] Most of the contrasts discussed here can be generalized to the differences in learning general deep generative models (i.e., those with latent variables) and in learning deep discriminative models with neural network architectures.

and with both discrete-valued and continuous-valued latent variables, and is hence highly intractable in inference and learning. Many empirically motivated approximate solutions have been explored both in model structure and in learning methods; see comprehensive reviews in [18, 31] and Chapters 10 and 12 of [29]. The more principled approximate approach, called variational inference and learning [81, 43, 59], has also been explored in learning the HDM. In [63, 19], it was found that variational inference works surprisingly well in inferring or estimating the intermediate, continuous-valued latent variables, i.e., vocal tract resonances or formants. But it does not do so well in inferring or decoding the top-level, discrete-valued latent variables, i.e., the phonetic label sequences, both in terms of decoding accuracy and computational cost. More recent developments in variational inference, especially the exploitation of the DNN in implementing efficient exact sampling from the variational posterior [76, 56, 4, 61, 60], hold promise to overcome some earlier weaknesses of HDM inference and learning.

On the other hand, rather than a wide range of inference and learning algorithms available to deep generative models, the RNN, as with most other neural network models, usually exploits a single, unchallenged inference and learning algorithm of backpropagation with at most some not so drastic variants. To bridge the above completely different styles of learning with strengths derived from both, the RNN and HDM need to be re-parameterized so that model parameterization will become similar to each other. The preceding subsection on parameterization touched on this topic from the general computational and modeling viewpoint. Recent work in machine learning also discussed a way of transformation between Bayesian networks and neural networks [61, 60]. The main idea is that although the posterior required in the E-step of the variational learning is often hard to approximate well in many intractable Bayesian network (such as the HDM), one can exploit a powerful DNN with high capacity to drastically improve the degree of the approximation.

1.6.6 Recognition Accuracy Comparisons

Given the analysis on and comparisons presented so far in this section between the generative HDM using localist representations and the discriminative RNN using distributed representations, the respective strengths and weaknesses associated with these two types of models are apparent. Here, let us compile and compare the empirical performance of the HDM and the RNN in terms of speech recognition accuracy. For calibration reasons, we use the standard TIMIT phone recognition task for the comparison since no other common tasks have been used to assess both types of models in a consistent manner. It is worthwhile mentioning that both types of the dynamic models are more difficult to implement than other models in use

for speech recognition such as the GMM-HMM and DNN-HMM. While the HDM has been evaluated on the large vocabulary tasks involving Switchboard databases [9, 82, 65, 64, 67], the RNN has been mainly evaluated on the TIMIT task [83, 48, 47, 10, 21], and most recently a non-standard large task [90, 91].

One particular version of the HDM, called the hidden trajectory model, was developed and evaluated after careful design with approximations aimed to overcome various difficulties associated with localist representations as discussed earlier in this section [35, 32]. The main approximation involves using the finite impulse response filter to replace the infinite impulse response filter with recurrent structure as in the original state equation of the state space formulation of the model. This version gives 75.2% phone recognition accuracy as reported in [32], somewhat higher than 73.9% obtained by the basic RNN (with very careful engineering) as reported in Table I (on page 303) of [83] and somewhat lower than 76.1% obtained by an elaborated version of the RNN with LSTM memory units without stacking as reported in Table I (on page 4) of [48]. [2] This comparison shows that the top-down generative HDM based on localist representation of the hidden state performs similarly to the bottom-up discriminative RNN based on distributed representation of the hidden state. This is understandable due to the pros and cons of these different types of models analyzed throughout this section.

1.7 Discussions

Deep hierarchical structure with multiple layers of hidden space in human speech is intrinsically connected to its dynamic character manifested in all levels of speech production and perception. The desire and an attempt to capitalize on a (superficial) understanding of this deep speech structure helped ignite the recent surge of interest in the deep learning approach to speech recognition and related applications [24, 51, 108, 23], and a more thorough understanding of the deep structure of speech dynamics and their representations is expected to further advance the research progress in speech modeling and in ASR. In Chapter **??** (Section 7 on HMM variants), we surveyed a series of deep as well as shallow generative models incorporating speech dynamics at various levels, including notably the HDM. In this chapter, we study the discriminative counterpart of the HDM, the RNN, both expressed mathematically in the state-space formalism. With detailed examinations of and comparisons between these two types of models, we focus on the top-down versus bottom-up information flows and localist versus distributed representations as their respective hallmarks and defining characteristics.

[2] With less careful engineering, the basic RNN accepting inputs of raw speech features could only achieve 71.8% accuracy as reported in [21]before using DNN-derived input features.

In the distributed representation adopted by the RNNs, we cannot interpret the meaning of activity on a single unit or neuron in isolation. Rather, the meaning of the activity on any particular unit depends on the activities of other units. Using distributed representations, multiple concepts (i.e., phonological/linguistic symbols) can be represented at the same time on the same set of neuronal units by superimposing their patterns together. The strengths of distributed representations used by the RNN include robustness, representational and mapping efficiency, and the embedding of symbols into continuous-valued vector spaces which enable the use of powerful gradient-based learning methods. The localist representation adopted by the generative HDM has very different properties. It offers very different advantages — easy to interpret, understand, diagnose, and easy to work with.

The interpretability of the generative HDM allows the design of appropriate structure in the dynamic system matrix that governs the articulatory-like dynamic behavior, where the constraint is imposed that no oscillatory dynamics occur within a regime of phone-like units.[3] It is much harder to develop and impose structural constraints in the discriminative RNN, where the hidden layer does not lend itself to physical interpretation. The LSTM structure described in Section 5 is a rare and interesting exception, motivated by very different considerations from those in structuring the HDM.

Both the HDM and the RNN are characterized by the use of dynamic recursion in the hidden space not directly observed. The temporal unfolding of these dynamic sequence models make the related architectures deep, with the depth being the length of the speech feature sequence to be modeled. In the HDM, the hidden state adopts the localist representation with explicit physical interpretation and the model parameters are indexed with respect to each separate linguistic/phonetic class in a parsimonious manner. In the RNN, the hidden state adopts the distributed representation where each unit in the hidden layer contributes to all linguistic classes via the shared use of regular and massive weight matrices.

The comprehensive comparisons between the RNN and the HDM conducted in Section 6 and summarized above are aimed to shed insights into the question of how to leverage the strengths of both types of models while overcoming their respective weaknesses. The integration of these two distinct types of generative and discriminative models may be done blindly as in the case of using the generative DBN to pre-train the discriminative DNN. However, much better strategies can be pursued a future research direction, given our sufficient understanding by now of the nature of the pros and cons associated with the two model types. As an example, one weakness associated with the discriminative RNN is that distributed representations are not suitable for providing direct input to the network. This difficulty has been circumvented in the preliminary work reported in [21] by first using the DNN to extract input features, which gains the advantages of distributed representa-

[3] For example, in [15], second-order dynamics with critical damping were used to incorporate such constraints.

tions embedded in the hidden layers of the DNN. Then the DNN-extracted features equipped with distributed representations of the data are fed into the subsequent RNN, producing dramatic improvement of phone recognition accuracy from 71.8% to as high as 81.2%. Other ways to overcome the difficulties associated with localist representations in the generative, deep, and dynamic model and those with distributed representations in the discriminative model counterpart are expected to also improve ASR performance. As a further example, given the strength of the localist representation in interpreting the hidden space of the model and thus in integrating domain knowledge, we can exploit the generative model and create new features extracted from the latent variables or even the generated visible variables that may be effectively combined with other features based on distributed representations.

Moreover, when the highly intractable deep generative models of speech such as the HDM are difficult to integrate with RNN, simpler and less deep or even shallow generative models may be exploited even though they incorporate speech-specific properties and constraints in a cruder manner. In this case, the shallow models can be stacked layer-by-layer to form a deep architecture, similar to stacking RBMs to form the DBN and to stacking neural nets with one hidden layer to form a deep stack net [36, 33]. Then the weights on different layers can be relaxed to independent of each other and the whole deep network can be trained using discriminative backpropagation with possible constraints. In this way, the original properties of speech embedded into the generative model will remain valid in the final deep model after backpropagation training.

We expect more advanced deep learning architectures in the future for more effective ASR to be superior to the current best RNN discussed in this chapter in several ways — Realistic properties of speech dynamics (based on human speech production and perception) in the continuous, latent, articulatory-like space will be beneficially exploited in the integrated generative and discriminative deep architectures; The learning of such integrated architecture will have more than a simple pass of BPTT but rather several iterative steps of top-down followed by bottom-up, and left-to-right followed by right-to-left; And a variety of effective deep generative-discriminative learning algorithms generalizing, expanding, or integrating into the current BPTT, along the line of [106, 97, 61, 76, 14], will be developed for the integrated generative-discriminative model emulating the deep and dynamic process of human speech in a functionally effective and computationally efficient manner.

References

1. Bazzi, I., Acero, A., Deng, L.: An expectation-maximization approach for formant tracking using a parameter-free nonlinear predictor. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2003)
2. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences **2**(1), 183–202 (2009)
3. Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. In: Neural Networks: Tricks of the Trade, pp. 437–478. Springer (2012)
4. Bengio, Y.: Estimating or propagating gradients through stochastic neurons. CoRR (2013)
5. Bengio, Y., Boulanger, N., Pascanu, R.: Advances in optimizing recurrent networks. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP). Vancouver, Canada (2013)
6. Bengio, Y., Boulanger-Lewandowski, N., Pascanu, R.: Advances in optimizing recurrent networks. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP). Vancouver, Canada (2013)
7. Boden, M.: A guide to recurrent neural networks and backpropagation. Tech. rep., TECHNICAL REPORT T2002:03, SICS (2002)
8. Boyd, S.P., Vandenberghe, L.: Convex Optimization. Cambridge university press (2004)
9. Bridle, J., Deng, L., Picone, J., Richards, H., Ma, J., Kamm, T., Schuster, M., Pike, S., Reagan, R.: An investigation fo segmental hidden dynamic models of speech coarticulation for automatic speech recognition. Final Report for 1998 Workshop on Langauge Engineering, CLSP, Johns Hopkins (1998)
10. Chen, J., Deng, L.: A primal-dual method for training recurrent neural networks constrained by the echo-state property. In: Proc. ICLR (2014)
11. Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. In: Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014)
12. Dahl, G.E., Yu, D., Deng, L., Acero, A.: Large vocabulary continuous speech recognition with context-dependent DBN-HMMs. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4688–4691 (2011)
13. Dahl, G.E., Yu, D., Deng, L., Acero, A.: Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Transactions on Audio, Speech and Language Processing **20**(1), 30–42 (2012)
14. Danilo Jimenez Rezende Shakir Mohamed, D.W.: Stochastic backpropagation and approximate inference in deep generative models. In: Proc. International Conference on Machine Learning (ICML) (2014)

15. Deng, L.: A dynamic, feature-based approach to the interface between phonology and phonetics for speech modeling and recognition. Speech Communication **24**(4), 299–323 (1998)
16. Deng, L.: Computational models for speech production. In: Computational Models of Speech Pattern Processing, pp. 199–213. Springer-Verlag, New York (1999)
17. Deng, L.: Switching dynamic system models for speech articulation and acoustics. In: Mathematical Foundations of Speech and Language Processing, pp. 115–134. Springer-Verlag, New York (2003)
18. Deng, L.: DYNAMIC SPEECH MODELS — Theory, Algorithm, and Applications. Morgan and Claypool (2006)
19. Deng, L., Attias, H., Lee, L., Acero, A.: Adaptive kalman smoothing for tracking vocal tract resonances using a continuous-valued hidden dynamic model. IEEE Transactions on Audio, Speech and Language Processing **15**, 13–23 (2007)
20. Deng, L., Bazzi, I., Acero, A.: Tracking vocal tract resonances using an analytical nonlinear predictor and a target-guided temporal constraint. In: Proc. Annual Conference of International Speech Communication Association (INTERSPEECH) (2003)
21. Deng, L., Chen, J.: Sequence classification using high-level features extracted from deep neural networks. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2014)
22. Deng, L., Hassanein, K., Elmasry, M.: Analysis of correlation structure for a neural predictive model with application to speech recognition. Neural Networks **7**, 331–339 (1994)
23. Deng, L., Hinton, G., Kingsbury, B.: New types of deep neural network learning for speech recognition and related applications: An overview. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP). Vancouver, Canada (2013)
24. Deng, L., Hinton, G., Yu, D.: Deep learning for speech recognition and related applications. In: NIPS Workshop. Whistler, Canada (2009)
25. Deng, L., Lee, L., Attias, H., Acero, A.: Adaptive kalman filtering and smoothing for tracking vocal tract resonances using a continuous-valued hidden dynamic model. Audio, Speech, and Language Processing, IEEE Transactions on **15**(1), 13–23 (2007)
26. Deng, L., Li, J., Huang, J.T., Yao, K., Yu, D., Seide, F., Seltzer, M., Zweig, G., He, X., Williams, J., Gong, Y., Acero, A.: Recent advances in deep learning for speech research at microsoft. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP). Vancouver, Canada (2013)
27. Deng, L., Li, X.: Machine learning paradigms in speech recognition: An overview. IEEE Transactions on Audio, Speech and Language Processing **21**(5), 1060–1089 (2013)
28. Deng, L., Ma, J.: Spontaneous speech recognition using a statistical coarticulatory model for the hidden vocal-tract-resonance dynamics. Journal Acoustical Society of America **108**, 3036–3048 (2000)
29. Deng, L., O'Shaughnessy, D.: SPEECH PROCESSING — A Dynamic and Optimization-Oriented Approach. Marcel Dekker Inc, NY (2003)
30. Deng, L., Ramsay, G., Sun, D.: Production models as a structural basis for automatic speech recognition. Speech Communication **33**(2-3), 93–111 (1997)
31. Deng, L., Togneri, R.: Deep dynamic models for learning hidden representations of speech features. In: Speech and Audio Processing for Coding, Enhancement and Recognition. Springer (2014)
32. Deng, L., Yu, D.: Use of differential cepstra as acoustic features in hidden trajectory modelling for phonetic recognition. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 445–448 (2007)
33. Deng, L., Yu, D.: Deep convex network: A scalable architecture for speech pattern classification. In: Proc. Annual Conference of International Speech Communication Association (INTERSPEECH) (2011)

34. Deng, L., Yu, D., Acero, A.: A bidirectional target filtering model of speech coarticulation: two-stage implementation for phonetic recognition. IEEE Transactions on Speech and Audio Processing **14**, 256–265 (2006)
35. Deng, L., Yu, D., Acero, A.: Structured speech modeling. IEEE Transactions on Speech and Audio Processing **14**, 1492–1504 (2006)
36. Deng, L., Yu, D., Platt, J.: Scalable stacking and learning for building deep architectures. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2012)
37. Divenyi, P., Greenberg, S., Meyer, G.: Dynamics of Speech Production and Perception. IOS Press (2006)
38. Fan, Y., Qian, Y., Xie, F., Soong, F.K.: TTS synthesis with bidirectional lstm based recurrent neural networks. In: Proc. Annual Conference of International Speech Communication Association (INTERSPEECH) (2014)
39. Fernandez, R., Rendel, A., Ramabhadran, B., Hoory, R.: Prosody contour prediction with long short-term memory, bi-directional, deep recurrent neural networks. In: Proc. Annual Conference of International Speech Communication Association (INTERSPEECH) (2014)
40. Geiger, J., Zhang, Z., Weninger, F., Schuller, B., Rigoll, G.: Robust speech recognition using long short-term memory recurrent neural networks for hybrid acoustic modelling. In: Proc. Annual Conference of International Speech Communication Association (INTERSPEECH) (2014)
41. Gers, F., Schmidhuber, J., Cummins, F.: Learning to forget: continual prediction with lstm. Neural Computation **12**, 2451–2471 (2000)
42. Gers, F., Schraudolph, N., Schmidhuber, J.: Learning precise timing with lstm recurrent networks. Journal Machine Learning Research **3**, 115–143 (2002)
43. Ghahramani, Z., Hinton, G.E.: Variational learning for switching state-space models. Neural Computation **12**, 831–864 (2000)
44. Gonzalez, J., Lopez-Moreno, I., Sak, H., Gonzalez-Rodriguez, J., Moreno, P.: Automatic language identification using long short-term memory recurrent neural networks. In: Proc. Annual Conference of International Speech Communication Association (INTERSPEECH) (2014)
45. Graves, A.: Sequence transduction with recurrent neural networks. In: ICML Representation Learning Workshop (2012)
46. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
47. Graves, A., Jaitly, N., Mahamed, A.: Hybrid speech recognition with deep bidirectional lstm. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP). Vancouver, Canada (2013)
48. Graves, A., Mahamed, A., Hinton, G.: Speech recognition with deep recurrent neural networks. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP). Vancouver, Canada (2013)
49. Heigold, G., Vanhoucke, V., Senior, A., Nguyen, P., Ranzato, M., Devin, M., Dean, J.: Multilingual acoustic models using distributed deep neural networks. Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2013)
50. Hermans, M., Schrauwen, B.: Training and analysing deep recurrent neural networks. In: Proc. Neural Information Processing Systems (NIPS) (2013)
51. Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., Kingsbury, B.: Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Processing Magazine **29**(6), 82–97 (2012)
52. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine **29**(6), 82–97 (2012)

53. Hinton, G., Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets. Neural Computation **18**, 1527–1554 (2006)
54. Hinton, G., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504 – 507 (2006)
55. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
56. Hoffman, M.D., Blei, D.M., Wang, C., Paisley, J.: Stochastic variational inference
57. Jaeger, H.: Short term memory in echo state networks. GMD Report 152, GMD - German National Research Institute for Computer Science (2001)
58. Jaeger, H.: Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach. GMD Report 159, GMD - German National Research Institute for Computer Science (2002)
59. Jordan, M., Sudderth, E., Wainwright, M., Wilsky, A.: Major advances and emerging developments of graphical models, special issue. IEEE Signal Processing Magazine **27**(6), 17,138 (2010)
60. Kingma, D., Welling, M.: Auto-encoding variational bayes. In: arXiv:1312.6114v10 (2014)
61. Kingma, D., Welling, M.: Efficient gradient-based inference through transformations between bayes nets and neural nets. In: Proc. International Conference on Machine Learning (ICML) (2014)
62. Kingsbury, B., Sainath, T.N., Soltau, H.: Scalable minimum bayes risk training of deep neural network acoustic models using distributed hessian-free optimization. In: Proc. Annual Conference of International Speech Communication Association (INTERSPEECH) (2012)
63. Lee, L., Attias, H., Deng, L.: Variational inference and learning for segmental switching state space models of hidden speech dynamics. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP), vol. 1, pp. I–872 – I–875 (2003)
64. Ma, J., Deng, L.: A path-stack algorithm for optimizing dynamic regimes in a statistical hidden dynamic model of speech. Computer Speech and Language **14**, 101–104 (2000)
65. Ma, J., Deng, L.: Efficient decoding strategies for conversational speech recognition using a constrained nonlinear state-space model. IEEE Transactions on Audio and Speech Processing **11**(6), 590–602 (2003)
66. Ma, J., Deng, L.: Efficient decoding strategies for conversational speech recognition using a constrained nonlinear state-space model. IEEE Transactions on Audio, Speech and Language Processing **11**(6), 590–602 (2004)
67. Ma, J., Deng, L.: Target-directed mixture dynamic models for spontaneous speech recognition. IEEE Transactions on Audio and Speech Processing **12**(1), 47–58 (2004)
68. Maas, A.L., Le, Q., O'Neil, T.M., Vinyals, O., Nguyen, P., Ng, A.Y.: Recurrent neural networks for noise reduction in robust asr. In: Proc. Annual Conference of International Speech Communication Association (INTERSPEECH). Portland, OR (2012)
69. Mesnil, G., He, X., Deng, L., Bengio, Y.: Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In: Proc. Annual Conference of International Speech Communication Association (INTERSPEECH). Lyon, France (2013)
70. Mikolov, T.: Rnntoolkit http://www.fit.vutbr.cz/ imikolov/rnnlm/ (2012). URL http://www.fit.vutbr.cz/ imikolov/rnnlm/
71. Mikolov, T.: Statistical language models based on neural networks. Ph.D. thesis, Brno University of Technology (2012)
72. Mikolov, T., Deoras, A., Povey, D., Burget, L., Cernocky, J.: Strategies for training large scale neural network language models. In: Proc. IEEE Workshop on Automfatic Speech Recognition and Understanding (ASRU), pp. 196–201. IEEE, Honolulu, HI (2011)

73. Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., Khudanpur, S.: Recurrent neural network based language model. In: Proc. Annual Conference of International Speech Communication Association (INTERSPEECH), pp. 1045–1048. Makuhari, Japan (2010)
74. Mikolov, T., Kombrink, S., Burget, L., Cernocky, J., Khudanpur, S.: Extensions of recurrent neural network language model. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5528–5531. Prague, Czech (2011)
75. Mikolov, T., Zweig, G.: Context dependent recurrent neural network language model. In: Proc. IEEE Spoken Language Technology Workshop (SLT), pp. 234–239 (2012)
76. Mnih, A., K. Gregor, .: Neural variational inference and learning in belief networks. In: Proc. International Conference on Machine Learning (ICML) (2014)
77. Mohamed, A.r., Dahl, G.E., Hinton, G.: Deep belief networks for phone recognition. In: NIPS Workshop on Deep Learning for Speech Recognition and Related Applications (2009)
78. Ozkan, E., Ozbek, I., Demirekler, M.: Dynamic speech spectrum representation and tracking variable number of vocal tract resonance frequencies with time-varying dirichlet process mixture models. IEEE Transactions on Audio, Speech and Language Processing **17**(8), 1518 –1532 (2009)
79. Pascanu, R., Gulcehre, C., Cho, K., Bengio, Y.: How to construct deep recurrent neural networks. In: The 2nd International Conference on Learning Representation (ICLR) (2014)
80. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: Proc. International Conference on Machine Learning (ICML). Atlanta, GA (2013)
81. Pavlovic, V., Frey, B., Huang, T.: Variational learning in mixed-state dynamic graphical models. In: UAI, pp. 522–530. Stockholm (1999)
82. Picone, J., Pike, S., Regan, R., Kamm, T., bridle, J., Deng, L., Ma, Z., Richards, H., Schuster, M.: Initial evaluation of hidden dynamic models on conversational speech. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP) (1999)
83. Robinson, A.J.: An application of recurrent nets to phone probability estimation. IEEE Transactions on Neural Networks **5**(2), 298–305 (1994)
84. Robinson, A.J., Cook, G., Ellis, D.P., Fosler-Lussier, E., Renals, S., Williams, D.: Connectionist speech recognition of broadcast news. Speech Communication **37**(1), 27–45 (2002)
85. Rumelhart, D.E., Hintont, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
86. Sainath, T., Kingsbury, B., Soltau, H., Ramabhadran, B.: Optimization techniques to improve training speed of deep neural networks for large speech tasks. IEEE Transactions on Audio, Speech, and Language Processing **21**(11), 2267–2276 (2013)
87. Sainath, T.N., Kingsbury, B., Mohamed, A.r., Dahl, G.E., Saon, G., Soltau, H., Beran, T., Aravkin, A.Y., Ramabhadran, B.: Improvements to deep convolutional neural networks for lvcsr. In: Proc. IEEE Workshop on Automfatic Speech Recognition and Understanding (ASRU), pp. 315–320 (2013)
88. Sainath, T.N., Kingsbury, B., Mohamed, A.r., Ramabhadran, B.: Learning filter banks within a deep neural network framework. In: Proc. IEEE Workshop on Automfatic Speech Recognition and Understanding (ASRU) (2013)
89. Sainath, T.N., Kingsbury, B., Sindhwani, V., Arisoy, E., Ramabhadran, B.: Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6655–6659 (2013)
90. Sak, H., Senior, A., Beaufays, F.: Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: Proc. Annual Conference of International Speech Communication Association (INTERSPEECH) (2014)

91. Sak, H., Vinyals, O., Heigold, G., Senior, A., McDermott, E., Monga, R., Mao, M.: Sequence discriminative distributed training of long short-term memory recurrent neural networks. In: Proc. Annual Conference of International Speech Communication Association (INTERSPEECH) (2014)
92. Schmidhuber, J.: Deep learning in neural networks: An overview. CoRR **abs/1404.7828** (2014)
93. Seide, F., Fu, H., Droppo, J., Li, G., Yu, D.: On parallelizability of stochastic gradient descent for speech dnns. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2014)
94. Seide, F., Li, G., Yu, D.: Conversational speech transcription using context-dependent deep neural networks. In: Proc. Annual Conference of International Speech Communication Association (INTERSPEECH), pp. 437–440 (2011)
95. Shen, X., Deng, L.: Maximum likelihood in statistical estimation of dynamical systems: Decomposition algorithm and simulation results. Signal Processing **57**, 65–79 (1997)
96. Stevens, K.: Acoustic Phonetics. MIT Press (2000)
97. Stoyanov, V., Ropson, A., Eisner, J.: Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. Proc. International Conference on Artificial Intelligence and Statistics (AISTATS) (2011)
98. Sutskever, I.: Training recurrent neural networks. Ph.D. thesis, University of Toronto (2013)
99. Togneri, R., Deng, L.: Joint state and parameter estimation for a target-directed nonlinear dynamic system model. Signal Processing, IEEE Transactions on **51**(12), 3061–3070 (2003)
100. Togneri, R., Deng, L.: A state-space model with neural-network prediction for recovering vocal tract resonances in fluent speech from mel-cepstral coefficients. Speech communication **48**(8), 971–988 (2006)
101. Triefenbach, F., Jalalvand, A., Demuynck, K., Martens, J.P.: Acoustic modeling with hierarchical reservoirs. IEEE Transactions on Audio, Speech, and Language Processing **21**(11), 2439–2450 (2013)
102. Vanhoucke, V., Devin, M., Heigold, G.: Multiframe deep neural networks for acoustic modeling
103. Vanhoucke, V., Senior, A., Mao, M.Z.: Improving the speed of neural networks on CPUs. In: Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning (2011)
104. Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K.J.: Phoneme recognition using time-delay neural networks. IEEE Transactions on Speech and Audio Processing **37**(3), 328–339 (1989)
105. Weninger, F., Geiger, J., Wollmer, M., Schuller, B., Rigoll, G.: Feature enhancement by deep lstm networks for ASR in reverberant multisource environments. Computer Speech and Language pp. 888–902 (2014)
106. Xing, E., Jordan, M., Russell, S.: A generalized mean field algorithm for variational inference in exponential families. In: Proc. Uncertainty in Artificial Intelligence (2003)
107. Yu, D., Deng, L.: Speaker-adaptive learning of resonance targets in a hidden trajectory model of speech coarticulation. Computer Speech and Language **27**, 72–87 (2007)
108. Yu, D., Deng, L.: Deep-structured hidden conditional random fields for phonetic recognition. In: Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2010)
109. Yu, D., Deng, L., Acero, A.: A lattice search technique for a long-contextual-span hidden trajectory model of speech. Speech Communication **48**, 1214–1226 (2006)
110. Yu, D., Deng, L., Dahl, G.: Roles of pre-training and fine-tuning in context-dependent DBN-HMMs for real-world speech recognition. In: Proc. Neural Information Processing Systems (NIPS) Workshop on Deep Learning and Unsupervised Feature Learning (2010)