Fall 11-14-2016

Towards Deeper Understanding in Neuroimaging

Rex Devon Hjelm
Mind Research Network

Rex Devon Hjelm
Candidate

Computer Science
Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Trilce Estrada , Chairperson

Vince Calhoun

Sergey Plis

Kyunghyun Cho

Ruslan Salakhutdinov

Towards Deeper Understanding in Neuroimaging

by

(Rex) Devon Hjelm

B.S., Astrophysics, University of New Mexico, 2004
M.S., Physics, University of New Mexico, 2007
M.A., Linguistics, University of New Mexico, 2009

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Ph.D., Computer Science

The University of New Mexico

Albuquerque, New Mexico

November, 2016

Towards Deeper Understanding in Neuroimaging

by

(Rex) Devon Hjelm

B.S., Astrophysics, University of New Mexico, 2004

M.S., Physics, University of New Mexico, 2007

M.A., Linguistics, University of New Mexico, 2009

Ph.D., Computer Science, University of New Mexico, 2016

Abstract

Neuroimaging is a growing domain of research, with advances in machine learning having tremendous potential to expand understanding in neuroscience and improve public health. Deep neural networks have recently and rapidly achieved historic success in numerous domains, and as a consequence have completely redefined the landscape of automated learners, giving promise of significant advances in numerous domains of research. Despite recent advances and advantages over traditional machine learning methods, deep neural networks have yet to have permeated significantly into neuroscience studies, particularly as a tool for discovery. This dissertation presents well-established and novel tools for unsupervised learning which aid in feature discovery, with relevant applications to neuroimaging. Through our works within, this dissertation presents strong evidence that deep learning is a viable and important tool for neuroimaging studies.

Contents

1 Introduction **1**

1.1 Deep Learning 3

1.2 Overview of Dissertation Research 5

2 Generative Models **8**

2.1 Undirected graphical models 11

2.2 Directed graphical models 12

2.3 Training graphical models 14

3 Undirected Graphical Models: Applications to Neuroimaging and Methods **15**

3.1 Restricted Boltzmann Machines
for Identifying Intrinsic Networks 17

3.1.1 RBM and ICA for IN Analysis 18

3.1.2 Spatial map and timecourse analysis 20

3.1.3 Synthetic data 21

3.1.4 fMRI data 24

3.2 Recurrent Neural Networks as Generative Samplers of Non-Sequential Data 29

3.2.1 Recurrent Neural Networks for Sequence Modeling 30

3.2.2 RNN-based Sampler of I.I.D Data 32

3.2.3 RNN Chains Results 37

3.2.4 An RNN to Explore the Manifold of Energy-Based Models 38

4 Directed Graphical Models, Applications, and Extensions 42

4.1 Variational Lowerbound of a Directed Belief Network 43

4.2 Helmholtz machines 44

4.3 Variational Autoencoders for Feature Detection of Magnetic Resonance Imaging Data 45

4.3.1 Re-parameterization of logistic distribution 45

4.3.2 Visualizing latent variables 46

4.3.3 Experimental setup 46

4.3.4 Results 47

4.4 Recurrent Neural Networks for Spatiotemporal Dynamics of Intrinsic Networks from fMRI Data 49

4.4.1 Independent Component Analysis 50

4.4.2 RNN ICA 52

4.4.3 Time-dependent Spatial Maps 53

4.4.4 Experiments 55

5 Iterative Refinement Inference Techniques 62

5.1 Discrete latent variables
and improving the approximate posterior 63

5.2 Iterative refinement of the approximate posterior 65

5.2.1 Adaptive Importance Refinement (AIR) 66

5.2.2 Algorithm and Complexity 67

5.3 Related Work 67

5.4 Experiments and Results 69

5.4.1 Settings 69

5.4.2 Variance Reduction and Choosing the AIR Objective 71

5.4.3 Training and Density Estimation 72

5.4.4 Posterior Improvement 74

5.4.5 Continuous Latent Variables 76

5.4.6 Refinement of the Lowerbound and Effective Sample Size 78

5.4.7 Updates and Wall-Clock Times 79

5.4.8 Bidirectional Helmholtz Machines and AIR 80

5.5 Iterative Refinement for Generative Adversarial Networks 81

5.5.1 Generative Adversarial Networks 81

5.5.2 Adaptive Refinement Adversarial Networks 82

5.5.3 Results 86

6 Conclusions **88**

6.1 Summary of Work 88

6.2 Acknowledgements 90

Chapter 1

Introduction

Machine learning has become an integral part of biological and biomedical research, with automated learners being necessary to address many of the challenges associated with large and complex data. While offering tremendous power for overcoming data complexity, applications require reducing some of the complexity through domain knowledge, as most automated learners are unable to process raw, unadulterated data.

While domain knowledge and expertise guides applications, in many domains machine learning is a tool for exposing unknown structure in or fundamentals of the data. This is especially true in neuroscience research, as domain knowledge is still relatively incomplete and only peripherally understood from a long-running and combined effort from physiological, psychological, and epidemiological studies. For example, intrinsic networks (INs, Biswal, Zerrin Yetkin, Haughton, and Hyde, 1995) are hypothetical functionally and spatially coherent regions of the brain which can be verified from temporally coherent blood oxygenation level-dependent (BOLD) signals obtained from function magnetic resonance imaging (fMRI). INs can be realized as the solution of a "blind source separation" problem, where the observed BOLD signal is hypothesized to be a linear mixture of sources. The most widely-used and successful approach for separating BOLD signal is independent

component analysis (ICA, Bell and Sejnowski, 1995), and the resulting INs as sources have been shown to be diagnostic (Smith, Fox, Miller, Glahn, Fox, Mackay, Filippini, Watkins, Toro, Laird, et al., 2009; Swanson, Eichele, Pearlson, Kiehl, Yu, and Calhoun, 2011; Calhoun, Adali, Pearlson, and Pekar, 2001). However, observations from BOLD signal are not the only evidence for functionally-coherent, localized regions of the brain; in fact these regions are an active hypothesis from a variety of biological, biomedical, and psychological domains, including Voxel-Based Morphometry (VBM) or electroencephalogram (EEG).

Traditionally, the most successful machine learning methods (e.g., ICA) represent only linear, simple nonlinear (e.g., well-defined non-linear kernels), or other carefully designed structures. These design choices are usually not primarily in response to any fundamental structure in the data, but are frequently conveniences that makes optimizations possible. However, such constraints place a fundamental limit on structural capacity and applicability of these models. Moreover, traditional learning algorithms are strongly reliant on preprocessing steps, which necessarily remove possibly important information from raw data. Finally, extending to accommodate limitations in capacity and applicability often requires creative, hand-tuned, or case-driven approaches, and care must be taken so that such extensions do not violate constraints that make learning tractable.

Recent advances in deep neural networks, or more broadly in *deep learning*, have begun to surpass successful constrained machine learning methods, in many cases completely replacing them in many important domains. These models can act as universal function approximators (Hornik, Stinchcombe, and White, 1989) and can generalize over very large sets of data, making them both more flexible and powerful with large and complex datasets. As a research tool, the added flexibility can reduce the overall dependence on domain knowledge, allowing for an overall more data-driven approach. Despite these advantages, applications of deep learning to biomedical data and neuroscience is relatively new, particularly in the context of feature discovery and studies that reveal ground truth.

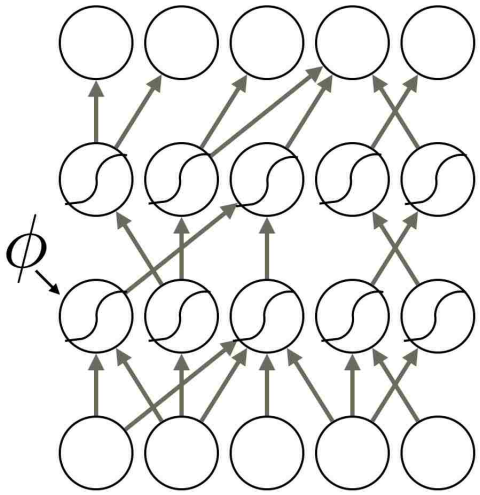

Figure 1.1: A simple neural network with sigmoid nonlinearities. Edges represent affine transformations, $\mathbf{x} \mapsto \mathbf{W}\mathbf{x} + \mathbf{b}$, where $\mathbf{x}$ is a vector in the space of inputs, $\mathbf{W}$ is a linear transformation on the space of inputs, and $\mathbf{b}$ is a vector in the space of outputs. The nodes represent a nonlinear mapping, $\mathbf{x} \mapsto \phi(\mathbf{x})$, where $\phi(.)$ is a nonlinear map on the input space (output space of the affine transformation).

This gives us strong motivation to explore the potential of deep learning in neuroimaging settings.

1.1 Deep Learning

Deep learning is a subdomain of machine learning where the solution space is made up of "deep" nonlinear maps. Each map is performed by alternating an affine transformation with a nonlinear (often exponential) function:

1. **Affine transformation:** $\mathbf{x} \mapsto \mathbf{W}\mathbf{x} + \mathbf{b}$, where $\mathbf{x}$ is a vector in the space of inputs, $\mathbf{W}$ is a linear transformation on the space of inputs, and $\mathbf{b}$ is a vector in the space of

outputs.

2. **Nonlinearity:** $\mathbf{x} \mapsto \phi(\mathbf{x})$, where $\mathbf{x}$ is a vector in the input space, and $\phi(\mathbf{x})$ is a nonlinear map on the output space of the affine transformation to the input space of the next affine map (if applicable).

A canonical example is the widely used *feed-forward network* (FFN), a.k.a the *multi-layer perceptron* (MLP) model. The objective of a typical FFN is to predict outputs given inputs for every input-output pair, $(\mathbf{i}^{(n)}, \mathbf{o}^{(n)})$ in the data. In FFN models, the relationship between inputs and outputs is a deterministic mapping, $\mathbf{o}^{(n)} = f(\mathbf{i}^{(n)}; \boldsymbol{\theta})$, with parameters, $\boldsymbol{\theta}$. The output of a FFN is computed by L iterative applications of an affine transformation with parameters $\boldsymbol{\theta}_l = \{\mathbf{W}_l, \mathbf{b}_l\} \subseteq \boldsymbol{\theta}$ and a nonlinear function, $\phi_l(.)$ (Figure 1.1):

$$\mathbf{o}_l^{(n)} = \phi_l(\mathbf{W}_l \mathbf{o}_{l-1}^{(n)} + \mathbf{b}_l), \tag{1.1}$$

where $\mathbf{o}_0^{(n)} := \mathbf{i}^{(n)}$, $\mathbf{W}_l$ is a matrix of "weights" for the l-th layer, and $\mathbf{b}_l$ is a bias vector. The non-linearity, $\phi(x)$, is commonly in the exponential family, such as sigmoid ($\sigma(x) = \frac{1}{1+\exp(-x)}$) or hyperbolic tangent ($\tanh(x)$), but other families of functions are also used, such as softplus ($\log(1 + \exp(x))$) or the rectified linear approximation, ReLU (Nair and Hinton, 2010).

In order to satisfy its objective, a learning algorithm is employed to ensure that the output of the FNN given an input, $\mathbf{i}^{(n)}$, "fits" the target, $\mathbf{o}^{(n)}$. This is typically realized through a squared-error or logistic regression loss function. The FFN parameters, $\boldsymbol{\theta}$, are found by iteratively applying stochastic gradient descent (SGD) along with back-propagation, which is simply an application of the chain rule iteratively applied through layers from the error signal to the input.

Neither neural networks nor back-propogation algorithms are new, however (Schmidhuber, 2015). Historically, neural networks were not able to deliver on promised power, as overfitting and local minima made learning difficult, and neural networks were quickly

replaced by simpler and more successful models such as support vector machines (SVMs). It wasn't until quite recently that the potential of FNN became sufficiently realized, thanks to hardware advancements with graphic processor units (GPUs) and additional tricks to reduce overfitting, such as dropout (Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov, 2014).

Now, basic FFNs, convolutional neural networks (CNN, LeCun and Bengio, 1995), and recurrent neural networks (RNNs) have become dominant in prediction tasks in many important domains with complex large-scale datasets (LeCun, Bengio, and Hinton, 2015). CNNs also encode translational symmetries in the data, relaxing the need for registration or other preprocessing in many practical cases, and have completely supplanted SVMs in many complex computer-vision tasks (Krizhevsky, Sutskever, and Hinton, 2012). RNNs allow for modeling of sequential data, and gated extensions such as long short-term memory (LSTM, Hochreiter and Schmidhuber, 1997) units and gated recurrent units (GRUs, Cho, Van Merriënboer, Gulcehre, Bahdanau, Bougares, Schwenk, and Bengio, 2014) have been very successful in natural language processing, speech synthesize and recognition, and translation (Sutskever, Vinyals, and Le, 2014; Cho et al., 2014; Bahdanau, Cho, and Bengio, 2014; Graves, Mohamed, and Hinton, 2013; Ze, Senior, and Schuster, 2013; Oord, Dieleman, Zen, Simonyan, Vinyals, Graves, Kalchbrenner, Senior, and Kavukcuoglu, 2016; Graves, 2013). The clear advantage in design flexibility and scalability continues to show that deep learning approaches are effective and powerful for big data analysis.

1.2 Overview of Dissertation Research

Conceptually, machine learning is split into two related, yet distinct subdomains: supervised and unsupervised learning.[1] Supervised learning amounts to specialized function approximation, mapping from a set of inputs to a corresponding set of targets, and training

[1] Ignoring semi-supervised learning

FFNs for classification or regression are an examples of this.

The goal of unsupervised learning is far broader, and I attempt to summarize it as: *to reduce the complexity of the data, oftentimes for the purposes of extrapolating characteristics, learning hidden or latent structure, or gaining understanding*. Clustering and density estimation are examples of unsupervised learning. Unsupervised learning often involves *inference*, where, given the data, the task is to find or infer the more likely hidden or latent structure that describes the data. This is relatively difficult in most practical cases, and most unsupervised learning methods require some combination of approximations, clever algorithms, and model constraints.

While recent advances in deep learning have completely changed the landscape of supervised learning, advancements in unsupervised learning have been slow and inference still requires strong constraints and tricks. Many of the recent advances in unsupervised learning (Salakhutdinov and Hinton, 2009; Kingma and Welling, 2013; Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, and Bengio, 2014) have yet to become a part of a general framework in scientific discovery, and this is no less true in brain research. Our recent work (Hjelm, Calhoun, Salakhutdinov, Allen, Adali, and Plis, 2014; Hjelm, Plis, and Calhoun, 2016c; Plis, Hjelm, Salakhutdinov, and Calhoun, 2013; Castro, Hjelm, Plis, Dihn, Turner, and Calhoun, 2016) and subsequent work by others (Suk, Lee, Shen, Initiative, et al., 2014; Kim, Calhoun, Shim, and Lee, 2016) have pushed into this relatively unexplored domain of research, but to our knowledge no unsupervised deep learning approach has yet to achieve widespread acceptance in any one medical imaging field.

As most of the interesting problems involve inferring latent structure for the sake of discovery within a domain, further advances in unsupervised learning are crucial to biomedical and neuroscience research. Successes of deep learning in supervised settings and advances in unsupervised settings strongly suggest that advances in structure discovery in neuroimaging research will involve adaptation of and improvement on established deep

learning methods.

Supporting this hypothesis, this dissertation will first introduce unsupervised graphical models that incorporate deep learning approaches in training and inference. Next, it will cover applications of undirected graphical models to neuroimaging (Hjelm et al., 2014) as well as some approaches for using recurrent neural networks (RNNs) as generative samplers. In the fifth chapter, this dissertation will cover directed graphical models, some applications to magnetic resonance imaging (MRI) data (Hjelm et al., 2016c), and a novel method for combining RNNs with ICA with an application to modeling dynamics in intrinsic networks (INs) (Hjelm, Plis, and Calhoun, 2016b). In the last section, our recent work on training directed graphical models using iterative refinement will be covered (Hjelm, Cho, Chung, Salakhutdinov, Calhoun, and Jojic, 2016a), as well as an extension to generative adversarial networks (GANs, expanding on work from Goodfellow et al., 2014).

Chapter 2

Generative Models

A generative model is a model that represents the density of data, typically through hidden parameters that relate the observations to stochastic latent variables. These models are designed to *generate* random examples of the data with representative statistics determined from interactions between latent and observed random variables. The generative objective is descriptive rather than discriminative, and as a consequence these models are argued to generalize better than discriminative ones.

A well-known example of a generative model is the Gaussian mixture (GMM), the objective of which is to fit the data onto a mixture of Gaussian distributions. GMMs are *multimodal*, and given enough modes, a GMM can be a universal density approximator of data. Real-world data can be highly multimodal, so good generative models need to be able to faithfully model as many modes as is necessary. For instance, the distribution of handwritten digits from the MNIST dataset (LeCun, Cortes, and Burges, 1998), a relatively simple dataset, has at least 10 modes, one for each digit, but possibly many more.

While learning a multimodal distribution is often a task in unsupervised learning, the importance of multimodality is also apparent in supervised tasks as well. The success of classification depends on prior knowledge of a set of well-defined known modes of the data

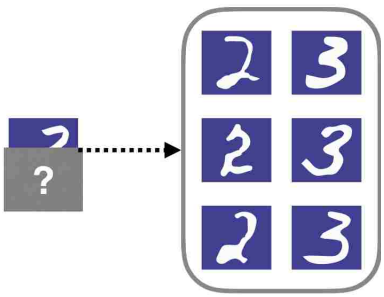

Table 2.1: A multimodal matching problem.

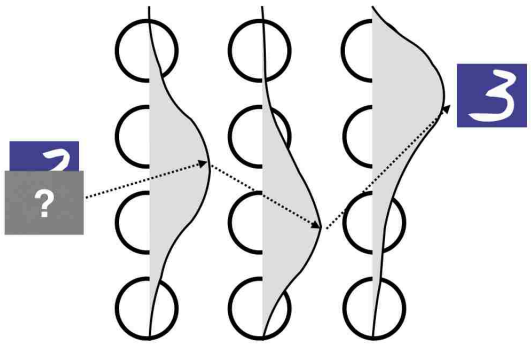

Table 2.2: A modality represented through a series of latent variables.

corresponding to the labels, in this case each of the 10 digits. The deterministic mapping from input observations to output probabilities can be interpreted as inference to a mixture model of digits.

Multimodality is also necessary to perform supervised structured output prediction (SOP, Bakir, 2007) tasks, where the data isn't already broken into distinct classes. Consider the conditional distribution of possible MNIST digits, given half of a digit (Figure 2.1). Given only partial information of a digit, the space of plausible completions should be multimodal, as many very different digits could plausibly fit the observed half. A deterministic FFN can be used to map from the space of the first half to the second half of the digits, but this output distribution is unimodal, unless explicitly a mixture (Bishop, 1994). However, the number of modes in typical mixture models must be set a-priori (with exceptions, such as Dirichlet mixtures (Ferguson, 1973)), which can be problematic when the underlying structure of the data is not well understood. Finally, while more modes can offer a higher modeling capacity, too many a-priori modes in a mixture model can cause overfitting as well as make analysis of modal structure difficult.

In many real-world applications, the number of modes may be difficult to prescribe. Consider the diagnosis of disease, where clinical diagnoses are based on manifestations of

complex and inter-related symptoms, some of which may or may not be detectable for a given disease, and many of which individually can indicate completely different diseases. Diagnosis is an active, often life-critical hypothesis, not absolute ground truth, and different symptoms may or may not be related to a specific diagnosis or set of diagnoses, depending on the subject. One could design a classifier to predict the diagnosis (mode) given the symptoms, including as many hand-picked modes chosen from the domain that helps with our task. However, one must account for every modality that may relate symptoms to diagnosis and other symptoms, and a researcher can only at best validate their hand-picked model design rather than discover realizations of novel or unexpected structure.

A probabilistic model that incorporates stochastic latent variables can be used to learn latent structure and can universally approximate density, even if the latent conditional densities or priors are unimodal (Sutskever and Hinton, 2008). For instance, introducing a set of conditionally independent binomial units between the given and the predicted half of the digit, a model more generally known as a *stochastic feed-forward network* (SFFN, Neal, 1990), can model multiple modalities (Tang and Salakhutdinov, 2013) and are successful in structured output prediction tasks. The number of modes available to the model is exponential in the number of layers, as each modality is represented as a combination of dependent latent variables (Figure 2.2). This encourages generality, as each of the modalities must share common latent structure.

The probabilistic latent variables are what give a generative model the capacity to model complex distributions, and the multimodality of the data as represented by the model is distributed across these variables. In general, the density of the data as represented in generative models with latent variables is a joint distribution with the latent variables marginalized out: $p(\mathbf{x}) = \sum_{\mathbf{h}} p(\mathbf{x}, \mathbf{h})$. Training such models depends on the actual structure of the joint density $p(\mathbf{x}, \mathbf{h})$, which usually must be chosen a-priori. Two types of generative model structures, undirected and directed graphical models (Jordan, 1998), offer some of the highest flexibility in design and the greatest potential for density

estimation.

2.1 Undirected graphical models

An undirected graphical model represents observations with a probabilistic density function, $p(\mathbf{x})$, where each variable, x_i, of N observations, $\mathbf{x}^{(n)} = \{x_i^{(n)}\} \sim p(\mathbf{x})$, are nodes on an undirected graph. Specifically, the type of undirected graphical model of interest here is a conditional independence graph, which forms a *Markov random field*. In a conditional independence graph, the joint density, $p(\mathbf{x}) = p(x_0, x_1, \ldots, x_M)$ [1], can be written as the normalized product of J "clique potentials", $\phi_j(C_j)$:

$$p(\mathbf{x}) = \frac{\prod_j^J \phi_j(C_j(\mathbf{x}))}{\mathcal{Z}}, \tag{2.1}$$

where $C_j(\mathbf{x}) \subseteq \mathbf{x}, \forall j$, and $\mathcal{Z} = \sum_{\mathbf{x}} \prod_j^J \phi_j(C_j(\mathbf{x}))$ is the *partition function*, which is a sum of the unnormalized density over all configurations of $\mathbf{x}$. Variables are independent, conditioned on all other variables in their respective cliques. The modalities of the data are represented by the clique factors, $\phi_j(C_j(\mathbf{x}))$.

Perhaps the most historically important undirected graphical model is the "Ising" model (Figure 2.1 shows one possible Ising model with 2D grid structure), where cliques, $C_j(\mathbf{x})$, are only over a local "neighborhood", so that a node is conditionally independent on non-neighbor nodes given all of its neighbors. This type of undirected model has roots in statistical mechanics and found early success in computer vision (Geman and Graffigne, 1986). This clique structure represents dependence as *locality*, which is similar in spirit to local filters in convolutional neural networks (CNNs, LeCun & Bengio, 1995), which is a de-facto standard in computer vision today.

[1] For simplicity in notation, I omit the sample index, n.

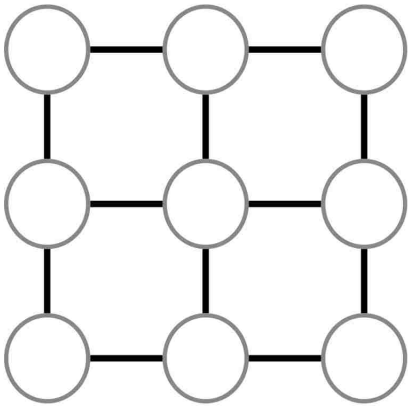

Figure 2.1: Undirected graphical model with 2D grid Ising structure. Each node represents a degree of freedom.

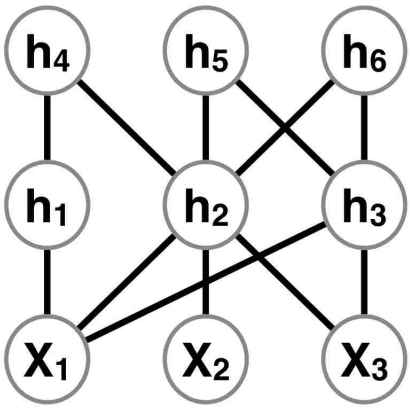

Figure 2.2: A deep Boltzmann machine with two hidden layers.

More recently, there has been some success in undirected graphical models with non-local, parametric exponential-family clique potentials along with latent variables, $\mathbf{h}$:

$$p(\mathbf{x}) = \sum_{\mathbf{h}} p(\mathbf{x}, \mathbf{h}); \quad p(\mathbf{x}, \mathbf{h}) = \frac{e^{-\sum_j^J E(C_j(\mathbf{x}, \mathbf{h})); \phi)}}{\mathcal{Z}}, \tag{2.2}$$

where the function, $E(.)$, is commonly referred to as the "energy function". Models that employ this sort of structure are known as *energy-based models*, and together with a hierarchal clique structure, ones arrives at undirected models with layer-wise conditionally independence (Figure 2.2), such as restricted Boltzmann machines (RBM, Hinton, 2002) and deep Boltzmann machines (DBM, Salakhutdinov & Hinton, 2009). These only have cliques between subsequent layers, omitting inter-layer connections, and these will covered in more detail in the context of neuroimaging data later.

2.2 Directed graphical models

Probabilistic directed graphical models, also known as "Bayesian belief networks", can be represented by an acyclic directed graph (Jordan, 1998) (Figure 2.3), where each of the

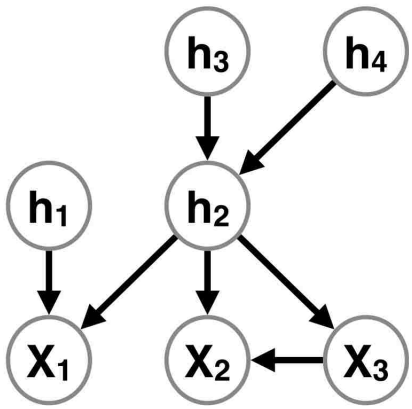

Figure 2.3: An undirected graphical model with observations at the end of the directed graph. Note that observations need not be arranged in this way.

edges represent a conditional probability. The joint probability of the nodes is the product of conditionals:

$$p(\mathbf{x}, \mathbf{h}) = \prod_{i}^{N} p(s_i | \{\mathcal{P}(s_i)\}), \tag{2.3}$$

where $\{\mathcal{P}(s_i)\}$ is the set of all parents of s_i.

Directed models are the most popular type of graphical model, and they are almost synonymous with Bayesian inference. Some of their appeal comes from that they provide a summary of the data density with as a set of simple, often unimodal and factorized factors. They are also easier to sample from than undirected models, which can be done by simple ancestral sampling. Deep latent models, such as deep latent Gaussian models (DLGM, Frey and Hinton, 1999; Rezende, Mohamed, and Wierstra, 2014), sigmoid belief networks (SBN, Neal, 1992), and variational autoencoders (VAE, Dayan, Hinton, Neal, and Zemel, 1995; Kingma & Welling, 2013), and hybrid models such as deep belief networks (DBN, Hinton, Osindero, and Teh, 2006), are highly flexible versions of directed graphical models. These allow for a high capacity, yet interpretable representation of data and have only recently emerged as being tractable and trainable. These will be covered in

more depth with neuroimaging data with some novel training approaches and extensions.

2.3 Training graphical models

Most parametric graphical models are trained by maximum likelihood estimation (MLE) or maximum a-priori (MAP) objectives. Given a directed or undirected model parameterized by ϕ, MLE seeks to maximize the log-probability of the data by finding a corresponding, maximally likely, $\phi^\star$:

$$\phi^\star = \arg\max_{\phi}(\log p(\mathbf{x}; \phi)) \tag{2.4}$$

The process of finding or choosing $\phi^\star$ involves *inference* which, in turn, is dependent on the specified latent structure. While methods such as Markov-chain Monte Carlo (Neal, 1992) and mean-field inference have shown success in some limited settings, there is no known efficient algorithm for solving MLE with a general graph, either directed or undirected.

In the following chapters, I will demonstrate training undirected and directed models as unsupervised learners. Applications will focus on neuroimaging data, training techniques, analysis, and some novel methods for unsupervised learning. I will also demonstrate novel learning algorithms for unsupervised learning, and demonstrate how existing methods can be extended upon.

Chapter 3

Undirected Graphical Models: Applications to Neuroimaging and Methods

In Section 2.1, I introduced *energy-based models*, which are undirected graphical models whose clique potential are an exponential of an "energy function". The fundamental difficulty in training energy-based models, as with all undirected graphical models, lies in the intractable partition function, $\mathcal{Z}$. Exact computation requires marginalization of latent variables over an exponential number of configurations, so learning algorithms must rely on estimates of the likelihood function and/or gradients.

For Boltzmann machines, it is convenient to introduce the *free energy*:

$$\mathcal{F}(\mathbf{x}) = -\log \sum_{\mathbf{h}} e^{-\sum_j^J E(C_j(\mathbf{x},\mathbf{h});\boldsymbol{\psi})}, \tag{3.1}$$

so that:

$$p(\mathbf{x}) = \frac{e^{-F(\mathbf{x})}}{\mathcal{Z}}; \quad \mathcal{Z} = \sum_{\mathbf{x}} e^{-F(\mathbf{x})}, \tag{3.2}$$

and the maximum likelihood gradient can formulated in terms of this free energy:

$$-\frac{\partial \log p(\mathbf{x})}{\partial \boldsymbol{\psi}} = \frac{\partial \mathcal{F}(\mathbf{x})}{\partial \boldsymbol{\psi}} - \sum_{\mathbf{x}'} p(\mathbf{x}') \frac{\partial \mathcal{F}(\mathbf{x}')}{\partial \boldsymbol{\psi}}. \tag{3.3}$$

The first term in the gradient, known as the "positive phase" or the data-driven term, can require approximate inference to compute, except in special cases. A restricted Boltzmann machines (RBMs) is a relatively simple Boltzmann machines with simple clique structure. With binomial random variables, the RBM energy is:

$$E(\mathbf{x}, \mathbf{h}) = -\mathbf{h}^\top \mathbf{W}\mathbf{x} - \mathbf{x} \cdot \mathbf{b} - \mathbf{h} \cdot \mathbf{c}, \tag{3.4}$$

where the model parameters, $\boldsymbol{\phi} = (\mathbf{W}, \mathbf{b}, \mathbf{c})$, are the interactive weights, visible biases, and hidden biases, respectively. With this energy function, inference of the latent variables is exact, as

$$\begin{aligned} \mathcal{F}(\mathbf{x}) &= -\log \sum_{\mathbf{h}} e^{(\mathbf{h}^\top \mathbf{W}\mathbf{x} + \mathbf{x}\cdot\mathbf{b} + \mathbf{h}\cdot\mathbf{c})} = -\log \sum_{\mathbf{h}} \prod_i e^{h_i(W^i\mathbf{x} + c_i)} e^{\mathbf{x}\cdot\mathbf{b}} \\ &= -\mathbf{x}\cdot\mathbf{b} - \sum_i \log(1 + e^{(W^i\mathbf{x} + c_i)}) \end{aligned} \tag{3.5}$$

$$\begin{aligned} p(h_i = 1|\mathbf{x}) &= \frac{\sum_{h_{j\neq i}} p(\mathbf{x}, h_j, h_i = 1)}{p(\mathbf{x})} = \frac{e^{-E(\mathbf{x}, h_i = 1)} \sum_{j\neq i} e^{-E(x, h_j)}}{e^{-\mathcal{F}(\mathbf{x})}} \\ &= \frac{e^{(W^i\mathbf{x} + c_i)} \prod_{j\neq i}(1 + e^{(W^j\mathbf{x} + c_j)})}{\prod_{i'}(1 + e^{(W^{i'}\mathbf{x} + c'_i)})} = \frac{1}{1 + e^{-(W^i\mathbf{x} + c_i)}}. \end{aligned} \tag{3.6}$$

Inference is exact with many energy functions without interaction terms between hidden units, such as Gaussian-Bernoulli or centered Bernoulli RBMs. For other Boltzmann machines with hierarchal hidden-hidden interactions such as deep Boltzmann machines (DBM), approximate mean-field inference can be sufficient to estimate the data-driven term in many cases.

The second term, or "negative phase" or "model expectation" is more problematic, as it requires calculating an expectation over the intractable marginal density. The true density, $p(\mathbf{x})$, is a stationary distribution of alternating block Gibbs sampling, so an estimate of the

negative phase term can be made by starting the Boltzmann machine at some initial state and performing sufficient Gibbs sampling to draw samples, $\mathbf{x}^{(m)}$:

$$\sum_{\mathbf{x}'} p(\mathbf{x}') \frac{\partial \mathcal{F}(\mathbf{x}')}{\partial \boldsymbol{\psi}} \approx \frac{1}{M} \sum_{m} \frac{\partial \mathcal{F}(\mathbf{x}^{(m)})}{\partial \boldsymbol{\psi}}. \tag{3.7}$$

However, drawing samples from a sufficient number of Markov chain Monte Carlo (MCMC) steps can be extremely expensive. Contrastive divergence (CD, Hinton, 2002) is an approximation that works by starting MCMC with data, then sampling for a small number of steps:

$$-\frac{\partial \log p(\mathbf{x})}{\partial \boldsymbol{\psi}} \approx \frac{\partial \mathcal{F}(\mathbf{x})}{\partial \boldsymbol{\psi}} - \frac{1}{M} \sum_{m} \frac{\partial \mathcal{F}(\mathbf{x}_k^{(m)})}{\partial \boldsymbol{\psi}}, \tag{3.8}$$

where $\mathbf{x}_k^{(m)}$ are the samples at the k-th step of the Gibbs chain. This approximation is sufficient for train RBMs, and typically $k = 1$ is sufficient in many practical applications (Hinton, 2010).

3.1 Restricted Boltzmann Machines for Identifying Intrinsic Networks

To demonstrate the potential of undirected models with high-dimensional neuroimaging data, my collaborators and I have applied an RBM as a method to separate intrinsic networks (INs, Biswal et al., 1995) from functional magnetic resonance imaging (fMRI) data (Hjelm et al., 2014). We compare RBMs to independent component analysis (ICA, Bell & Sejnowski, 1995), which is the dominant approach for inferring latent structure in brain imaging data. ICA represents fMRI as a linear mixture of maximally-independent sources (Smith et al., 2009; Swanson et al., 2011; Calhoun et al., 2001) and has been widely successful in separating INs and analyzing connectivity (Zuo, Kelly, Adelstein, Klein, Castellanos, and Milham, 2010; Kim, Burge, Lane, Pearlson, Kiehl, and Calhoun, 2008; Allen,

Erhardt, Wei, Eichele, and Calhoun, 2012; Damoiseaux, Rombouts, Barkhof, Scheltens, Stam, Smith, and Beckmann, 2006; Calhoun, Kiehl, and Pearlson, 2008; Smith et al., 2009).

3.1.1 RBM and ICA for IN Analysis

While RBMs and ICA are very different models, they have similarities: both resemble a bipartite graph between observations and latent variables with a weight matrix representing relationships between observation-latent pairs. For ICA, the data is generated from a linear mixture of the sources:

$$\mathbf{s} = \mathbf{W}\mathbf{x}, \tag{3.9}$$

where $\mathbf{W}$ is a square un-mixing matrix, [1] and $\mathbf{s}$ are stochastic latent variables drawn from a non-Gaussian distribution. In an fMRI context, the sources are called intrinsic networks (INs), their values over a time-series of fMRI data,

$$(\mathbf{s}_1, \mathbf{s}_2 \ldots, \mathbf{s}_T) = (\mathbf{W}\mathbf{x}_1, \mathbf{W}\mathbf{x}_2 \ldots, \mathbf{W}\mathbf{x}_T), \tag{3.10}$$

are called "time courses" (TCs), and the columns of the un-mixing matrix, W^i are "spatial maps" (SMs). Most popular ICA algorithms require the un-mixing matrix, $\mathbf{W}$, be square, and this oftentimes necessitates preprocessing of the data to reduce dimensionality to match the desired number of sources. This is usually done with principle component analysis (PCA), and in fMRI analyses that involves multiple subjects this is done in a two stage manner, first on each subject, then across all subjects. The most successful such approach, group independent component analysis (GICA, Calhoun et al., 2001), applies this reduction along the time-axis and performs ICA un-mixing in time. Also known as *spatial ICA*, the roles of the spatial maps and time courses are flipped, with the sources representing independent components in space.

[1] Some ICA algorithms parameterize the model with an invertible mixing matrix, $\mathbf{M} = \mathbf{W}^{-1}$, instead

In the optimal case, all important variance in the data is represented in ICA as originating from the factorized source distributions. Because the relationship between data and latent variables is represented through a discrete transformation (i.e., noiseless ICA), there is exactly one valid observable configuration for every hidden configuration. Even probabilistic ICA (PICA, Beckmann and Smith, 2004) is relatively limited as the observable configurations must be in a local neighborhood defined by independent noise.

RBMs, on the other hand, represents the data through a nonlinear energy function over joint configurations of observables and latent variables, so that the data optimally lies on minima of an energy manifold. This gives RBMs far more representational power than ICA; a single configuration of the latent variables in RBM can have many distinct low-energy (or valid) configurations over the observable variables. RBMs also do not require the weight matrices to be square, and thus do not require dimensionality-reducing preprocessing.

Linear mixture models such as ICA have been very successful in neuroimaging applications, and their success is in part due to a simple, tractable, and interpretable decomposition of the data. RBM offers a rich, nonlinear representation of data, and while this allows RBM a higher capacity to represent data, the cost is that said representation is not readily decomposable, as it is with ICA. While RBM has stochastic hidden units that resemble the ICA sources, these are not drawn from a factorized prior distribution, and thus cannot be said to be independent. For IN analysis, a researcher would benefit from RBMs powerful representational capacity, as well as not requiring dimensionality reduction preprocessing. However, RBMs need some modifications make the hidden units interpretable.

For modeling real-valued data, such as fMRI, it is first necessary to alter the energy function so that the conditional distribution, $p(\mathbf{x}|\mathbf{h})$, is Gaussian. In general, the variance of $p(\mathbf{x}|\mathbf{h})$ can be learned as well, but in practice it is often set to 1 for simplicity. In addition, RBMs with centered Bernoulli units tend work better, so we treat our hidden

variables as discrete binomial variables with values $\{-1, 1\}$. The final energy is, [2]

$$E(\mathbf{x}, \mathbf{h}) = -\mathbf{h}^\top \mathbf{W}\mathbf{x} - \frac{(\mathbf{x} - \mathbf{b})^2}{2} - \mathbf{h} \cdot \mathbf{c}. \tag{3.11}$$

For comparison to ICA, our analysis relies on interpreting each of the columns of the RBM weight matrix, W^i, or *filters*, as the spatial maps in an ICA framework. However, this may not entirely be appropriate, as the hidden units in RBM do not follow a factorized prior distribution. Our early experiments confirmed this: despite evidence of strong learning, features were highly non-local and overlapping in the space of voxels, making interpretation and comparison difficult. In order to encourage independence, we added strong L_1 regularization to the weights. Unfortunately, we cannot present strong motivating theory here, but it suffices for the work that high L_1 regularization encouraged the columns of the weight matrix to become orthogonal in our experiments. This causes the Gibbs chain to converge almost immediately. This indicates that the low-energy hidden configurations, which correspond to modes of the data distribution, are close to the conditional distributions, $p(\mathbf{h}|\mathbf{x})$ and thus independent.

3.1.2 Spatial map and timecourse analysis

The RBM analysis pipeline in the context of fMRI is illustrated in Figure 3.1. First, masked fMRI data is presented batch-wise to the RBM as i.i.d data, and the log-likelihood is maximized via contrastive divergence. We then use the learned parameters in a linear de-mixing model and forward-propagate the whole masked dataset for each subject to obtain subject-specific time course (TC) estimates. Dual regression is then used to acquire t-statistics for each component to task and novel stimulus. Finally, due to the relevance in current fMRI research (e.g., Allen, Erhardt, Damaraju, Gruner, Segall, Silva, Havlicek,

[2]Note that the energy does not change for centered binomial variables, but the form of the conditional, $p(\mathbf{h}|\mathbf{x})$, changes only by a scaling factor in the argument.

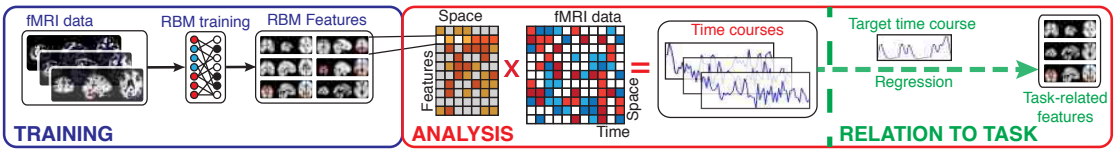

Figure 3.1: Pipeline for RBM training and analysis. An RBM is first used to obtain intrinsic networks (INs), each with a corresponding SM estimate. The model parameters are then used to forward propagate data onto the INs; this is done by matrix-multiplying the data (space$\times$time) with the SM estimates (features$\times$space), producing time course estimates specific to each IN. For task-related data (as with real fMRI data, multiple regression with time courses derived from the experimental protocol is used to identify task-related INs.

Rachakonda, Fries, Kalyanam, et al. (2011)), functional network connectivity (FNC) was computed as cross-correlations using subject-specific TCs.

3.1.3 Synthetic data

In order to objectively evaluate RBM performance compared to ICA, we first considered synthetic data. The SimTB toolbox (Erhardt, Allen, Wei, and Eichele, 2012) was used to generate synthetic $3D$ (x, y, and t) fMRI-like data from linear combinations of 27 distinct spatial sources with $2D$ Gaussian spatial profiles. RBMs were constructed with 16936 Gaussian visible units (one for each voxel), and a variable number of hidden units. The L_1 decay rate was set to 0.1 based on performance over multiple experiments. The RBMs were then trained with a batch size of 5 for approximately 75 epochs to allow for full convergence of the parameters. We constructed multiple RBMs with the number of hidden units ranging from 10 to 80 in steps of 10 in order to study the dependence of performance on model order.

We then compared performance of RBM to linear models by training ICA, PCA, sparse PCA (sPCA, Zou, Hastie, and Tibshirani, 2006), and sparse non-negative matrix factor-

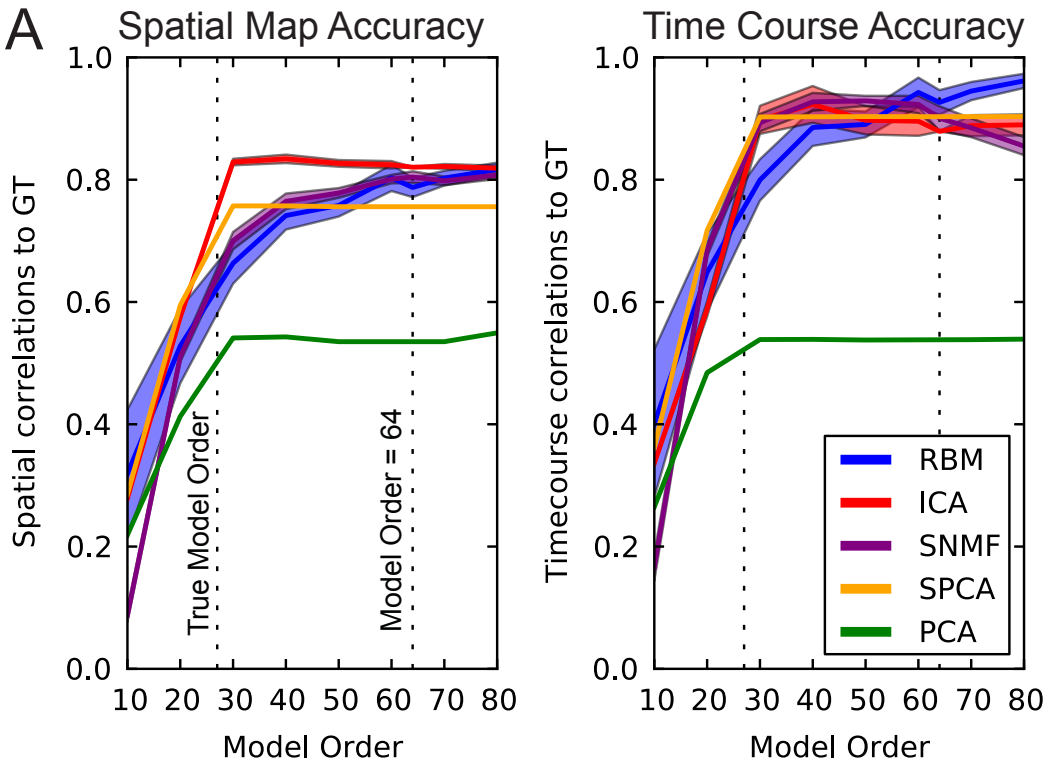

Figure 3.2: Correlation of SM (left) and TC (right) estimates to the ground truth for all models as a function of model order. Lines indicate average correlation across all datasets (and "subjects" for TCs) with a spatial overlap at 0.52, and the color-fill indicates $\pm$ two standard errors around the mean.

ization (sNMF, Hoyer, 2002). For group ICA, the data was preprocessed using a two-step PCA by reducing each subject data to 120 principal components, then reducing the concatenated subject principal components again to the final component number. For sPCA, we chose the sparsity by counting the number of voxels in each ground truth source that were above one standard deviation, then taking a rough average of those counts: 1000. For sNMF, a sparsity of 0.7 was used based on performance over multiple experiments.

Figure 3.2 shows the SM and TC estimation accuracy for RBM, ICA, sNMF, sPCA, and PCA. Considering both SM and TC estimation, all models other than PCA perform rather well, though show different sensitivities to model order. For RBM, both SM and TC

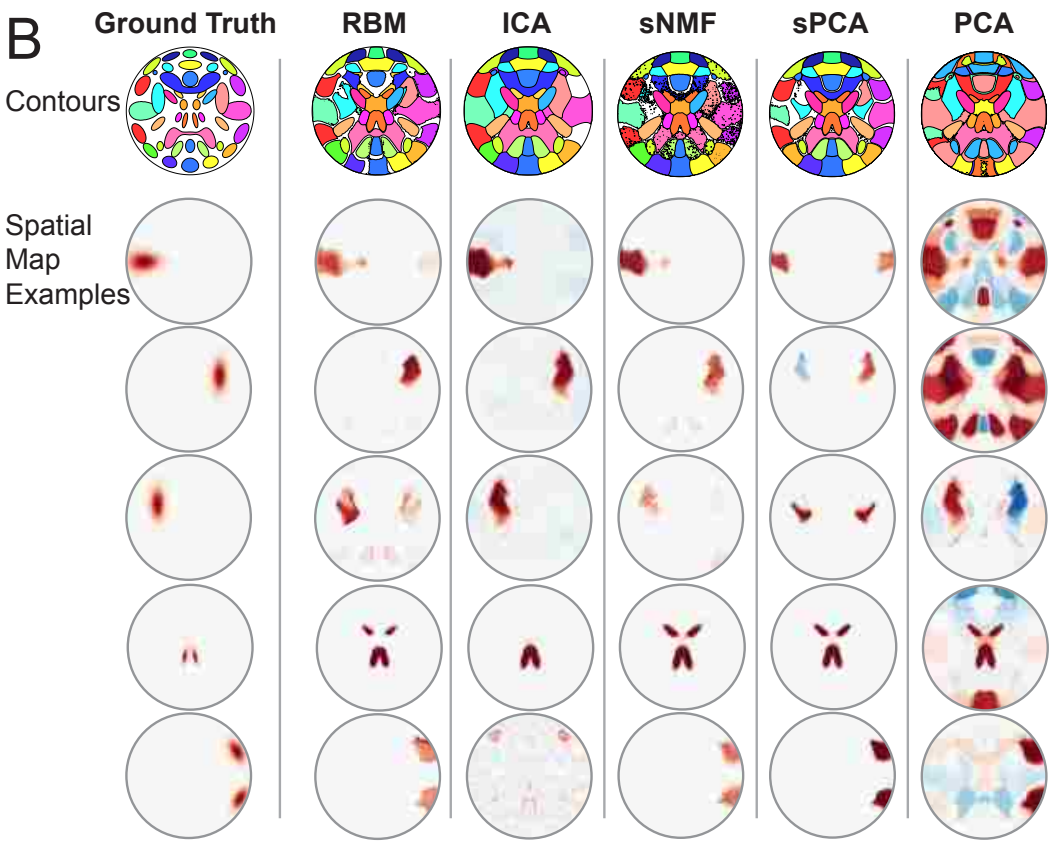

Figure 3.3: Contours of all SMs (top row) given a thresholded of 0.4 and examples of individual SMs for all models (bottom rows) at a model order of 64 and dataset overlap of 0.52. Typical SM quality between models was very similar and notable examples are shown. C) SM and TC accuracy as a function spatial overlap, at a model order of 64.

estimation accuracy increase with model order. For ICA, SM and TC accuracy appears to peak around a model order of 40, and decreases slightly at higher model orders. SNMF, in contrast, shows increased SM but decreased TC accuracy with increasing model order. SPCA is relatively insensitive to increases in model order beyond the true dimensionality. Figure 3.3 demonstrates that types of estimation errors observed. For lower model orders, we observed that RBM, sNMF, and sPCA tended to estimate SMs which appeared to be linear combinations of ground truth spatial maps. ICA failed to estimate a SM as a unique component, and instead merged it into the negative background of another SM.

Our simulations show that RBM performs competitively against all single matrix factorization (SMF) methods tested. Detailed comparisons with synthetic data show relatively comparable performance between RBM and ICA, with RBM providing slightly improved TC estimation and slightly worse SM accuracy over some ranges of model orders. We observe that the loss in spatial accuracy comes not from an inability to identify features, but from a tendency of RBMs to capture strong temporal correlations within the spatial domain. Thus, in the worst case, RBM SM estimates were still highly interpretable as combinations of temporally correlated ground truth sources.

3.1.4 fMRI data

Data used in our work comprised task-related scans from 28 healthy participants (five females), all of whom gave written, informed, IRB-approved consent at Hartford Hospital and were compensated for participation. All participants were scanned during an auditory oddball task (AOD) involving the detection of an infrequent target sound within a series of standard and novel sounds. More detailed information regarding participant demographics and task details are provided by (Swanson et al., 2011) and more details regarding preprocessing can be found in Hjelm et al. (2014). The RBM was constructed using 70969 Gaussian visible units and 64 hidden units, and ICA was trained using GIFT, preprocessed in the same way as with the SimTB data above and 64 components.

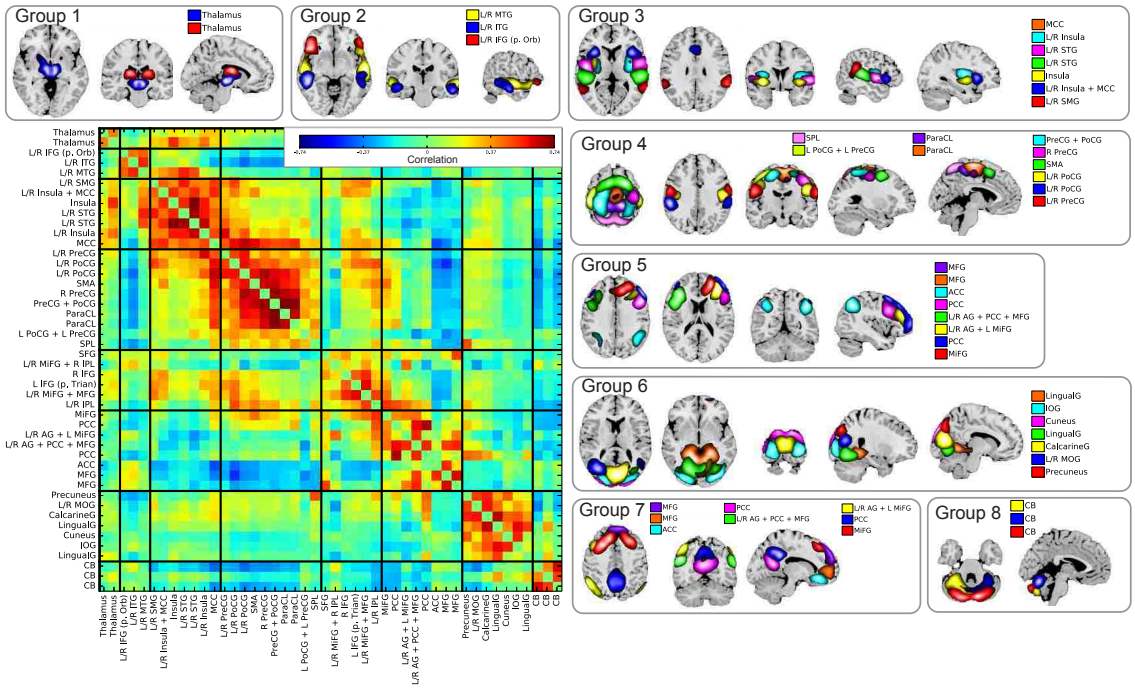

Figure 3.4: Groups of spatial maps of intrinsic networks estimated by RBM from fMRI data. INs were divided into groups based on their anatomical and functional properties. For visualization, each SM was thresholded at greater or less than 2 standard deviations. Estimated intrinsic networks are 1) thalamus; 2) inferior frontal gyrus (IFG); 3) middle temporal gyrus (MTG); 4) inferior temporal gryus (ITG); 5) middle cingulate cortex (MCC, overlapping with insula+MCC IN); 6) insula; 7) superior temporal gyrus (STG); 8) supramaginal gyrus (SMG); 9) superior parietal lobule (SPL); 10) precentral gyrus (PreCG); 11) postcentral gyrus (PoCG); 12) paracentral lobule (ParaCL); 13) supplementary motor area (SMA); 14) inferior parietal lobule (IPL); 15) middle frontal gyrus (MiFG); 16) medial frontal gyrus (MFG); 17) inferior frontal gyrus (IFG); 18) superior frontal gyrus (SFG); 19) lingual gyrus (LingualG); 20) inferior occipital gyrus (IOG); 21) cuneus; 22) calcarine gyrus (CalcarineG); 23) middle occipital gyrus (MOG); 24) precuneus; 25) anterior cingulate cortex (ACC); 26) posterior cingulate cortex (PCC); 27) angular gyrus (AG); 28) cerebellum (CB). The FNC (temporal correlation matrix) averaged across subjects is shown on the left.

After removing artifacts and white matter, we identified 46 INs with peaks in grey matter from RBM. The grouped INs as well as the FNC are shown in Figure 3.4. We labelled groups based on their spatial properties as: 1, subcortical; 2, temporal + frontal cortices;

Table 3.1: Counts of grey matter (GM) features, white matter (WM) features, ventricles, and other artifacts in RBM and ICA results. Component classification was performed manually.

	GM	**WM**	**Ventricles**	**Other**
RBM	46	5	5	8
ICA	38	15	6	5

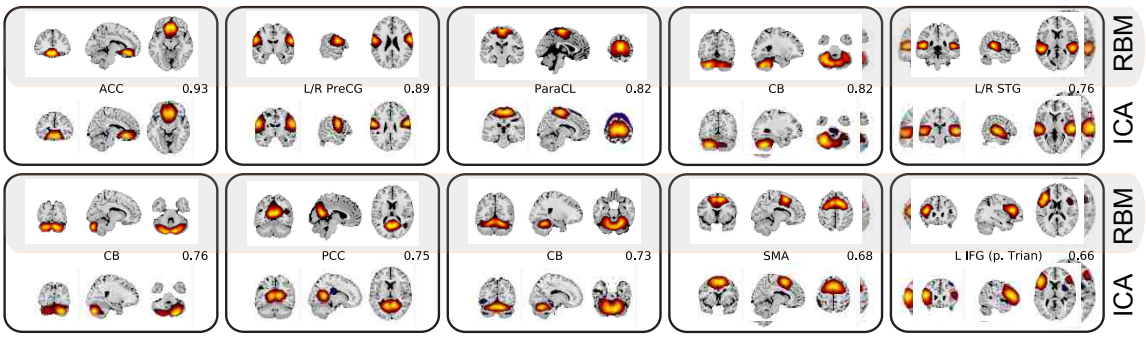

Figure 3.5: Sample pairs of RBM (top) and ICA (bottom) SMs thresholded at 2 standard deviations. Pairing was done with the aid of spatial correlations, temporal properties, and visual inspection. Values indicate the spatial correlation between RBM and ICA SMs.

3, temporal cortex; 4, somatomotor cortex; 5, frontoparietal cortex; 6, visual cortex; 7, default mode network (DMN); and 8, cerebellar networks. Applied to real fMRI data collected during an AOD task, the INs separated by RBM consistent with networks found in a variety of task and resting-state experiments (Allen et al., 2011; Beckmann, DeLuca, Devlin, and Smith, 2005; Calhoun et al., 2008; Damoiseaux et al., 2006; Kiviniemi, Starck, Remes, Long, Nikkinen, Haapea, Veijola, Moilanen, Isohanni, Zang, et al., 2009; Smith et al., 2009; Zuo et al., 2010).

Overall, RBM found a larger proportion of identifiable grey-matter features, while ICA distinguished more distinct white matter regions (Table 3.1). RBM and ICA agreed on most features with some variation (Figures 3.5), though disagreed on some default mode network (DMN) features. The TCs of corresponding RBM/ICA pairs showed no difference in their degree of task-relatedness (Figure 3.6), suggesting that RBM and ICA

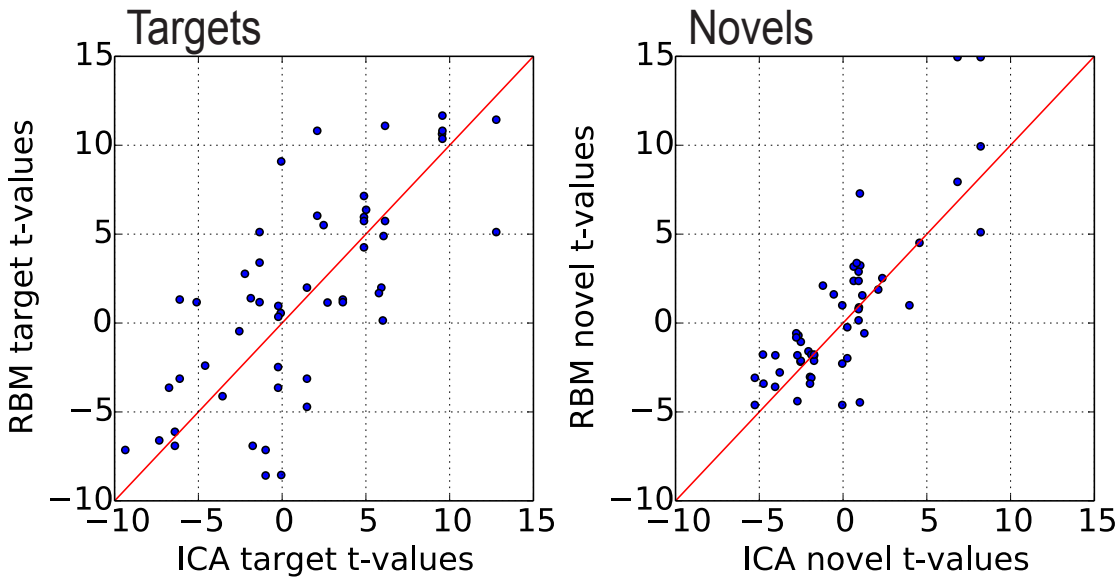

Figure 3.6: Target (left) and novel (right) t-statistics for RBM and ICA feature pairs with spatial correlations above 0.4. The red line indicates unity.

found different decompositions of the data to model the task equally well. Analysis and visual inspection of feature pairs generally shows RBM features are more sparse than those of ICA. In addition, RBM features are predominantly of a single sign, as opposed to ICA features which contain regions of both signs exceeding 2 standard deviations.

The most prominent difference between the models appeared in the FNC (Figure 3.7), where RBM yielded correlations of greater magnitude and significantly more modular connectivity structure, and hence greater ease in grouping components. Aided by similar results in simulations, we understand that this "enhanced" structure is due to the tendency of RBMs to capture strong temporal relations in the spatial domain, combined with the feed-forward mechanism used to produce the TCs. FNC biases are also likely to arise from the TCs estimated by ICA, however these will be dependent on the constraint of spatial independence, making biases difficult to predict when the true spatial dependencies are unknown (Allen et al., 2012).

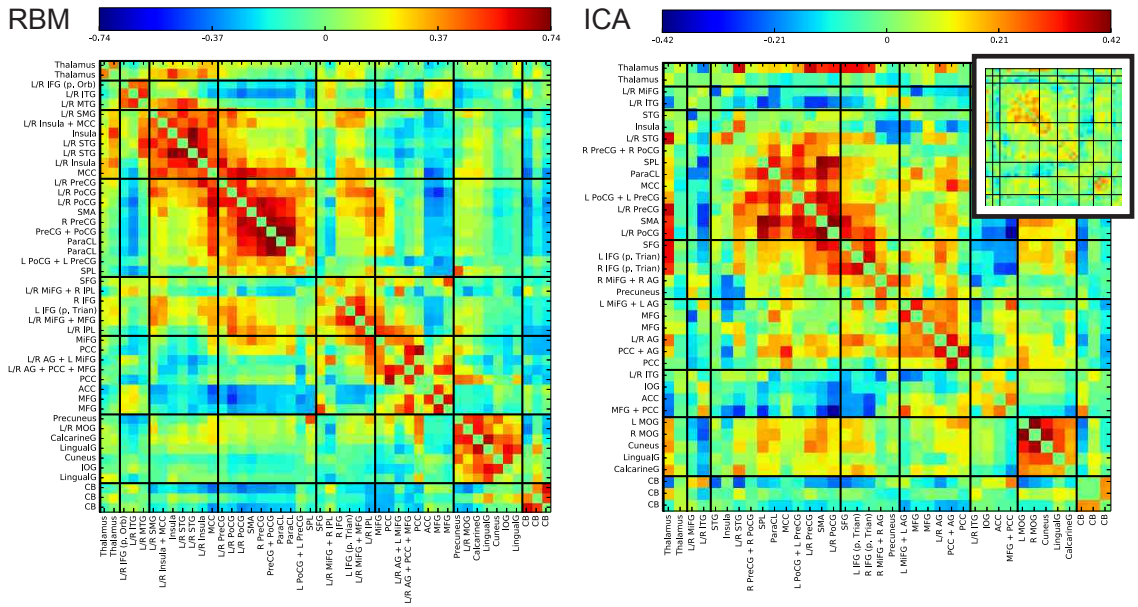

Figure 3.7: FNC determined from RBM (left) and ICA (right), averaged over subjects. Note that the color scales for RBM and ICA are different (RBM shows a larger range in correlations). The FNC matrix for ICA on the same scale as RBM is also provided as an inset (upper right). Feature groupings for RBM and ICA were manually determined separately, using the FNC matrices and known anatomical and functional properties.

While the results from RBM and ICA are surprisingly similar, though there is a theoretical basis for drawing strong parallels between the two approaches (Hinton, Welling, Teh, and Osindero, 2001). We note some key differences between the proposed RBM model and the typical application of spatial ICA to functional imaging data:

1. RBM does not make the same assumptions of linear factorization of the data as single matrix factorization (SMF) models like ICA, as RBM is an *energy-based* probability density model, thus freeing it from some linearity assumptions. However, in RBM, can be interpreted as a matrix factorization after training, making the types of analysis performed by SMF models available for research.

2. Although it operates on thousands of voxels directly, RBM does not require dimensionality reduction. Spatial ICA is largely impractical without dimensionality

reduction (typically via PCA), even when applied to only hundreds of time points. The PCA step in ICA introduces a variance bias, which can diminish our ability to identify certain sources; the avoidance of dimensionality reduction preprocessing represents an advantage of RBMs.

3. RBMs may provide an alternative to ICA as a method of blind source separation, though with enhanced residual cross correlation of sources.

3.2 Recurrent Neural Networks as Generative Samplers of Non-Sequential Data

RBMs typically use contrastive divergence (CD, Hinton, 2002) as an optimization method to approximately maximize the log-likelihood. While the single-step version of CD works in many applications, more steps are always better, as this effectively explores the space of the marginal density more accurately. While efficient with RBMs, CD has seen little success with Boltzmann machines with hidden to hidden interactions, such as with deep Boltzmann machines (DBMs, Salakhutdinov and Larochelle, 2010). CD fails in part because it can take a prohibitively long time for inference procedures to arrive at samples representative of the conditional distribution given the data. Alternatively, an efficient stochastic approximation procedure (SAP, Neal, 1992) known as persistent contrastive divergence (PCD, Tieleman, 2008) makes use of persistent chains to estimate the model expectation in Equation 3.3. PCD uses the same Gibbs chain for the MC estimate of the model expectation by alternating between advancing the chain and updating the parameters. In particular, the CD gradient in Equations 3.8 can be rewritten as a gradient for an incremental step in the persistent chain:

$$-\nabla \log p(\mathbf{x})_t \approx \frac{1}{N}\sum_n \nabla \mathcal{F}(\mathbf{x}^{(n)}) - \frac{1}{M}\sum_m \nabla \mathcal{F}(\mathbf{x}_t^{(m)}), \tag{3.12}$$

where $\mathbf{x}_t^{(m)}$ is the t-th sample in the Gibbs chain. This can greatly improve learning in RBMs and makes learning possible the DBMs.

Still, Gibbs sampling in many cases suffers from poor mixing, which in turn hurts density estimations and makes sampling difficult. Advanced techniques such as annealed importance sampling (AIS, Neal, 2001) or reverse AIS (RAISE, Burda, Grosse, and Salakhutdinov, 2014) provide alternatives in density estimation, but these methods can be expensive and are not efficient enough for learning. Improved MCMC techniques, such as Hamiltonian MCMC (Neal et al., 2011) and parallel tempering (Desjardins, Courville, Bengio, Vincent, and Delalleau, 2010; Cho, Raiko, and Ilin) can improve mixing and provide improved learning in a variety of settings (Salimans, Kingma, Welling, et al., 2015b). As an alternative, we show that a recurrent neural network (RNN) can be used as a generative sampler. The advantages include a parameterization of the sampling transition operator trainable with back-propagation, which can be tuned to fit the target sampling sequence directly.

3.2.1 Recurrent Neural Networks for Sequence Modeling

A recurrent neural network (RNN, Figure 3.8) is a type of neural network with recurrent or cyclic connections. In addition to having some of the same structure as simple feed forward networks, RNNs have an additional set of maps between their hidden recurrent units, $\mathbf{h}_t$. The maps can be "unfolded" across their recurrent connections (Figure 3.9a), which realizes RNNs as a deep FNN, making them easily trainable using back-propagation.

RNNs are well-suited for sequential or time-series data, and have been very successful in the field of natural language processing (NLP), including neural machine translation (NMT, Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014), speech recognition (Graves et al., 2013) and synthesis (Ze et al., 2013; Oord et al., 2016), written language recognition (Graves, 2013), etc. RNNs have many forms, but this work will

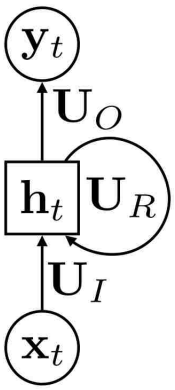

Figure 3.8: A graphical representation of a recurrent neural network. Hidden unit activations are a function of the input $\mathbf{x}_t$ and the previous hidden state, $\mathbf{h}_{t-1}$. Outputs, $\mathbf{y}_t$, are determined by the current hidden state, $\mathbf{h}_t$.

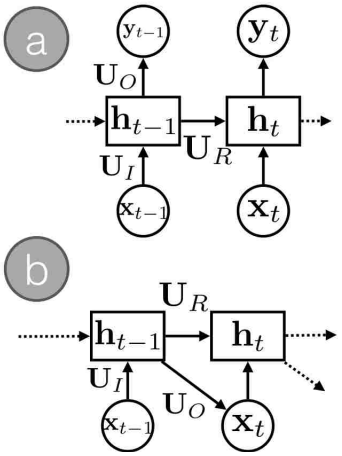

Figure 3.9: a) A recurrent neural network unfolded in time. Conceptually, this is what allows RNNs to be trained with the back-propagation algorithm. b) An RNN for sequence modeling. The RNN generates input at the next step.

focus on those that are probabilistic models of sequences (Figure 3.9b):

$$p(\mathbf{X}) = p(\mathbf{x}_{1:T}) = p(\mathbf{x}_1) \prod_{t=2}^{T} p(\mathbf{x}_t | \mathbf{x}_{1:t-1}). \tag{3.13}$$

The conditional density typically is drawn from a family of transformations,

$$p(\mathbf{x}_t | \mathbf{x}_{1:t-1}) = f(\mathbf{h}_t; \boldsymbol{\psi}_O); \qquad \mathbf{h}_t = g(\mathbf{h}_{t-1}, \mathbf{x}_{t-1}; \boldsymbol{\psi}_R), \tag{3.14}$$

where $\mathbf{h}_t$ are the set of deterministic recurrent units. $g(.; \boldsymbol{\psi}_R)$ are recurrent connections that take the current observation and hidden state as input and output the next recurrent state. The output connections, $f(.; \boldsymbol{\psi}_O)$, take the recurrent states as input at each step and output the parameters for a conditional distribution. Note that the model parameters, $\boldsymbol{\psi} = \{\boldsymbol{\psi}_O, \boldsymbol{\psi}_R\}$, are *recurrent*: the same parameters are used at every time step and do not change across the sequence index, t.

The most canonical RNN for sequence modeling has a simple parameterization:

$$g(\mathbf{h}_t; \boldsymbol{\psi}_R) = \tanh(\mathbf{U}_R \mathbf{h}_{t-1} + \mathbf{U}_I \mathbf{x}_{t-1} + \mathbf{b}), \tag{3.15}$$

where $\mathbf{U}_R$ is a square matrix of recurrent weights, $\mathbf{U}_I$ are the input weights, and $\mathbf{b}$ is a bias term. The mappings between the various variables in the model need not be shallow: deep neural networks can be used to model more complex recurrent transitions. Parameterizations that use gating and other types of memory functions, such as long short-term memory (LSTM, Hochreiter & Schmidhuber, 1997) and gated recurrent units (GRUs, Cho et al., 2014), are also widely used to better model longer sequences.

Training an RNN for simple sequence modeling is done with the back-propagation algorithm, using the negative log-likelihood objective over the output conditional distributions:

$$\mathcal{L}(\boldsymbol{\psi}; \mathcal{D}) \approx -\frac{1}{N} \sum_{n} \left(\sum_{t=2}^{T} \log p(\mathbf{x}_t^{(n)} | \mathbf{x}_{1:t-1}^{(n)}; \boldsymbol{\psi}) + \log p(\mathbf{x}_1^{(n)}; \boldsymbol{\psi}) \right), \tag{3.16}$$

over all sequences, $\mathbf{X}^{(n)} = (\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)} \dots \mathbf{x}_T^{(n)})$. Typically the loss is computed batch-wise instead of over the entire dataset for efficiency. The marginal density, $p(\mathbf{x}_1)$, can be trained by fitting to the average marginal across time, either to parameters of a target distribution directly or training a neural network to predict the hidden state that generates $\mathbf{x}_1$ (Bahdanau et al., 2014).

3.2.2 RNN-based Sampler of I.I.D Data

The Gibbs chain of an RBM is defined by a transition operator, $T(\widehat{\mathbf{x}}, \mathbf{x})$, whose stationary distribution, $\pi(\mathbf{x})$, is the marginal density of the RBM. The RBM objective effectively seeks to fit the stationary distribution to the empirical distribution of the data. In comparison, denoising autoencoders (dAE, Vincent, Larochelle, Lajoie, Bengio, and Manzagol, 2010) and deep generative stochastic networks (GSNs, Thibodeau-Laufer, Alain, and Yosinski, 2014) parameterize T directly, assuming that the transition is composed of alternating noising and reconstruction operations. As an alternative, I explore parameterizing the transition operator directly using an RNN. The goal will be to maximize the likelihood

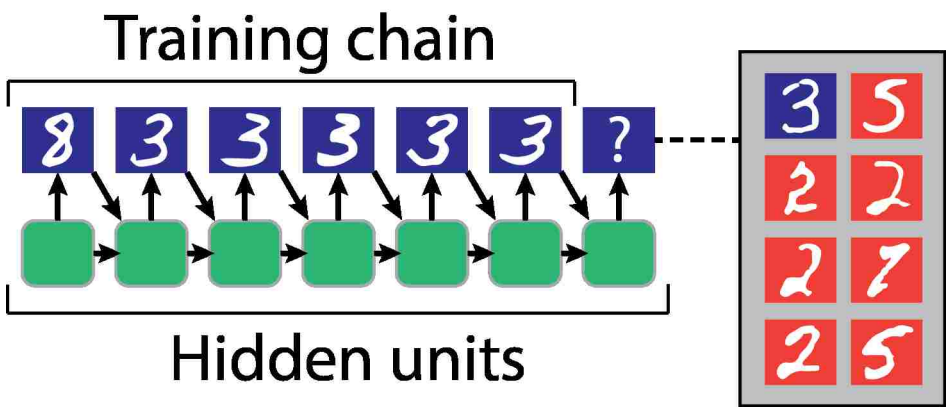

Figure 3.10: Procedure for approximately drawing from the distribution of ordered data as defined by an RNN. First, a samples is randomly drawn from the dataset. The next hidden state is computed via the RNN recurrent parameters. From the data samples not used, the "best" sample is selected as the next sample in the chain. The procedure is repeated until a stopping criteria is reached, either until the dataset is exhausted or the remaining unused samples are all about a threshold of probability.

of the transition operator over the empirical distribution of independent and identically distributed (i.i.d.) data.

Let us hypothesize that there exists a "true" distribution of orderings given an unordered set of data, $p(z|\mathcal{D}_i)$, where $\mathcal{D}_i$ is any unordered set of the complete data ($\mathcal{D}_i \subseteq \mathcal{D}$) of size N_i. z is a random variable indicating an ordering of $\mathcal{D}_i$, which induces an ordered set of N_i random variables, $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_{N_i})$. For now, let us assume that $\mathbf{X}$ has only unique elements, but in general this condition can be relaxed to allow repetitions. Given the hypothetical true distribution of orderings, $p(z|\mathcal{D}_i)$, one could maximize the likelihood of transitions through that ordering from an RNN with parameters, $\boldsymbol{\psi}$, by maximizing the MC estimate of the log-likelihood over all ordereings:

$$\mathcal{L}(\boldsymbol{\psi}; \mathcal{D}_i) = -\frac{1}{K} \sum_{k=1}^{K} \left(\sum_{n=2}^{N} \log p(\mathbf{x}_n^{(k)} | \mathbf{x}_{1:n-1}^{(k)}; \boldsymbol{\psi}) + \log p(\mathbf{x}_1^{(k)}; \boldsymbol{\psi}) \right), \tag{3.17}$$

where $\mathbf{x}_n^{(k)}$ are the ordered data samples with ordering, $z^{(k)}$, drawn from $p(z|\mathcal{D}_i)$.

Unfortunately, the "true" distribution of orderings is not available, and one cannot be

sure one such distribution even exists. However, working forward on this hypothesis, our assumption is that, given the RNN is optimized to the above objective, the conditional density of ordered datapoints should match the hypothetical true ordering density. In other words, given an optimal set of parameters, $\boldsymbol{\psi}^{\star}$:

$$p(\mathbf{x}_1^{(k)}; \boldsymbol{\psi}^{\star}) \prod_{n=2}^{N} p(\mathbf{x}_n^{(k)} | \mathbf{x}_{1:n-1}^{(k)}; \boldsymbol{\psi}^{\star}) \propto p(z^{(k)} | \mathcal{D}_i), \tag{3.18}$$

where the proportionality constant is the marginal over all configurations of the data on the RNN.

The naive approach to drawing samples from the above distribution over all possible configurations of the data is to simply evaluate the product above on the $N!$ total orderings, then sample as a multinomial distribution. However, evaluating the factorial number of configurations is clearly intractable, even for modest datasets. Instead, I use a fast greedy procedure that constructs chains in a sequential manner.

An illustration of the procedure is available in Figure 3.10. First, an initial datapoint, $\mathbf{x}_1^{(k)} \sim \mathcal{D}_i$, is drawn with either a uniform or prior probability. A sequence or "chain" of data, $\mathbf{X}^{(k)}$, is then iteratively constructed by selecting the next unused datapoint in $\mathcal{D}_i$ that maximizes the conditional density as defined by the RNN:

$$\mathbf{x}_n^{(k)} = \arg\max_{\mathbf{x}^{(k)}} p(\mathbf{x}^{(k)} | \mathbf{x}_{1\ldots n-1}; \boldsymbol{\psi}). \tag{3.19}$$

The procedure then iterates until a stopping criteria has been reached, usually when every data point from $\mathcal{D}_i$ has been selected or the selecting criteria or all remaining unused samples is above a threshold.

One reasonable objection to this approach is that the RNN might learn to model exactly one configuration, as the same RNN is being used to deterministically construct sequences over the same set of ordered data. This would drive the transition operator to model a single mode; an undesirable property if our model is to be general. In fact, this procedure no longer optimizes Equation 3.17, as samples from the "true" distribution of orderings have

been replaced by samples drawn from an approximate or proposal distribution defined by the procedure, $q(z|\mathcal{D}_i;\boldsymbol{\psi})$, which leads to a biased estimator the true log-likelihood. Introducing the proposal distribution, q, introduces the *lower bound* to the log-likelihood:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\psi};\mathcal{D}_i) &= -\log \sum_z q(z|\mathcal{D}_i) \left(\frac{p(z|\mathcal{D}_i)}{q(z|\mathcal{D}_i)} p(\mathbf{x}_1^{(k)}) \prod_{n=2}^{N} p(\mathbf{x}_n^{(k)}|\mathbf{x}_{1:n-1}^{(k)}) \right) \\
&\geq -\sum_z q(z|\mathcal{D}_i) \log \left(\frac{p(z|\mathcal{D}_i)}{q(z|\mathcal{D}_i)} p(\mathbf{x}_1^{(k)}) \prod_{n=2}^{N} p(\mathbf{x}_n^{(k)}|\mathbf{x}_{1:n-1}^{(k)}) \right) \\
&\approx -\frac{1}{K} \sum_{k=1}^{K} \left(\sum_{n=2}^{N} \log p(\mathbf{x}_n^{(k)}|\mathbf{x}_{1:n-1}^{(k)}) + \log p(\mathbf{x}_1^{(k)}) \right) \\
&\quad + \mathcal{D}_{KL}(q(z|\mathcal{D}_i)||p(z|\mathcal{D}_i)).
\end{aligned} \tag{3.20}$$

The intractable second term is the KL-divergence of q and p, which is a sum of two terms. The first term maximizes the log-likelihood, which ensures that q is close to the true ordering. This cannot be evaluated, so we need to rely on the soundness of our ordering procedure to account for this. The second term maximizes the *entropy* of q, $\mathcal{H}(q)$, which should regulate q from falling into a single mode, but this is intractable. Instead, an increase in entropy can be incorporated into the procedure the following ways:

- *Construct chains using only subsets of the data.* Given a complete dataset of size N, $\mathcal{D}$, draw $\mathcal{D}_i$ by first shuffling $\mathcal{D}$ and selecting the first N_i samples. N_i should be sufficiently small compared to the total size, N.

- *Evaluate only a subset of data when constructing a chain.* Given a subset of data, $\mathcal{D}_i$, use a random subset, $\mathcal{D}_c \subset \mathcal{D}_i$, when evaluating candidate samples in chain construction.

- *Sample from a multinomial over conditional densities.* Rather than performing the $\arg\max$ to construct $\mathbf{X}^k$, draw each $\mathbf{x}_n^{(k)}$ from the multinomial distribution defined by $p(\mathbf{x}|\mathbf{x}_{1:n-1}^{(k)})$ normalized over all remaining samples in $\mathcal{D}_i$.

Algorithm 1 Chains model

Require: An i.i.d. dataset with N samples.

Require: A chain length, N_i and a sampling size N_c, such that $N_c < N_i < N$.

while training **do**

  while $size(\mathcal{D}) > 0$ **do**

    Draw and remove random $\mathcal{D}_i$ from $\mathcal{D}$ of size $min(size(\mathcal{D}), N_i)$.

    Initialize $\mathbf{X} = (\mathbf{x}_1)$ with $\mathbf{x}_1$ drawn and removed from $\mathcal{D}_i$ with uniform probability.

    for $n = 2 : N_i$ **do**

      Select random $\mathcal{D}_c$ from $\mathcal{D}_i$ of size $min(size(\mathcal{D}_i), N_c)$.

      Evaluate $\alpha_n^{(m)} = p(\mathbf{x}^{(m)}|\mathbf{X}) / \sum_{m'} p(\mathbf{x}^{(m')}|\mathbf{X}) \quad \forall \mathbf{x}^{(m)} \in \mathcal{D}_c$.

      Draw and remove $\mathbf{x}^{(m)}$ from $\mathcal{D}_c$ using the multinomial distribution defined by $\alpha_n^{(m)}$.

      Set $\mathbf{X} \leftarrow (\mathbf{x}_1, \ldots \mathbf{x}_{n-1}, \mathbf{x}^{(m)})$.

    end for

    $\Delta\boldsymbol{\phi} \propto \sum_{n=2}^{N} \log p(\mathbf{x}_n|\mathbf{x}_{1:n-1}; \boldsymbol{\psi}) + \log p(\mathbf{x}_1; \boldsymbol{\psi})$

  end while

end while

The procedure used in this work is summarized in Algorithm 1. While RNNs are not typically used to model i.i.d. data, the possible benefits of this approach may include:

1. When trained with the "correct" orderings, the RNN will represent a compact representation between data points and the distribution of samples from the RNN should match the true distribution with unbound sampling steps.

2. The RNN could provide a faster mixing mechanism than undirected graphical models as it models *dynamics* and represents whole sequences.

3. The RNN provides a natural similarity metric defined by the conditional probabilities. This could be used for unsupervised tasks, such as clustering or inpainting, and supervised tasks such as classification.

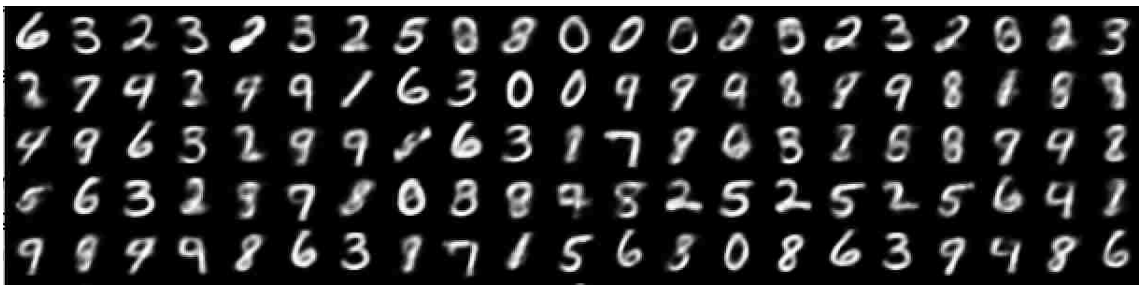

Figure 3.11: Fast moving samples from the chains model trained on the MNIST dataset with a chain length of 5000. Samples show very fast mixing without natural-looking transitions. This indicates that the RNN is probably only learning a mixture of the training set.

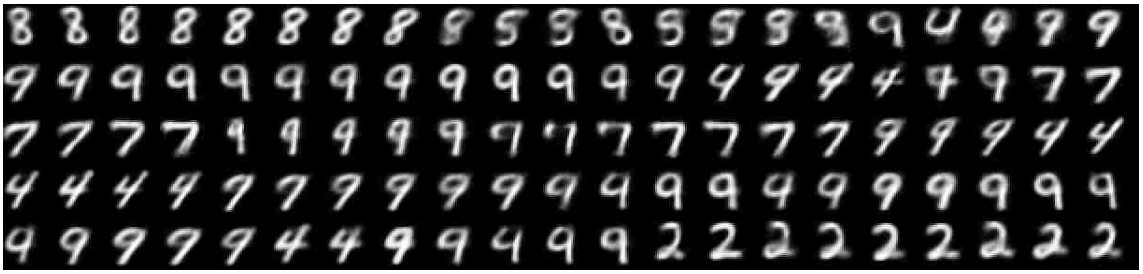

Figure 3.12: Slow sampling version of the chains model trained on the MNIST dataset with a chain length of 1000. Samples show natural-looking transitions, with some jumps. It is likely that the RNN has learned how to generalize and is not simply sampling from a mixture over the training set.

3.2.3 RNN Chains Results

To evaluate the chains procedure, I trained RNNs with gated recurrent units (GRUs, Cho et al., 2014) and 1000 hidden units on the MNIST data. For the input and output parameterizations, I used a feed forward network with one hidden layer with 500 hidden units. For training, I used the RMSProp algorithm Hinton (2012) for 500 epochs. Each chain constructed used 5000 or 1000 samples of the data, querying at most 1000 random samples at each step of building the chain. Each sample was selected by drawing from the multinomial distribution defined by the probability of the next samples given the current chain.

The results show an RNN sampler that is able to generate realistic and diverse MNIST digits with either very fast (Figure 3.11) or slower (Figure 3.12) mixing, depending on the size of the chains. Each of these achieved very good Parzen-window log-likelihood estimates (269.4 and 215) respectively, while longer chains outperformed other samplers in the literature (Goodfellow et al., 2014). However, the Parzen-window log-likelihood estimator can be quite inaccurate: a Gaussian blurring on the MNIST training set can produce estimates nearly identical to ours. It is therefore likely that the fast-sampling RNN is learning a mixture of the training data, and this particularly evident in that concurrent samples are qualitatively dissimilar. However, the slower-mixing RNN trained on smaller chains shows many smooth with few fast transitions while still exhibiting good mixing properties. Therefore, the RNN, together with the greedy procedure, seems to be able to learn a reasonable transition operator through the data similar to sequential samplers, as long as the model is sufficiently constrained. Continued research needs to rely on better evaluation of the model however, and further applications need to be explored to verify the value of this approach, and this is left for future research.

3.2.4 An RNN to Explore the Manifold of Energy-Based Models

Similar to our approach with i.i.d. data above, one can use an RNN to sample from the RBM marginal density, $p(\mathbf{x})$. In some sense, this problem is easier than the i.i.d. case: RBMs already provide sequences from which to train the RNN. However, the slow-mixing in RBMs is exactly the problem to avoid, so one must rely on some other methods to train a faster sampler.

In particular, we wish to train a generative RNN that models the conditional distribution (Figure 3.9):

$$q(\mathbf{X}) = q(\mathbf{x}_1) \prod_{t=2}^{T} q(\mathbf{x}_t | \mathbf{x}_{0:t-1}), \tag{3.21}$$

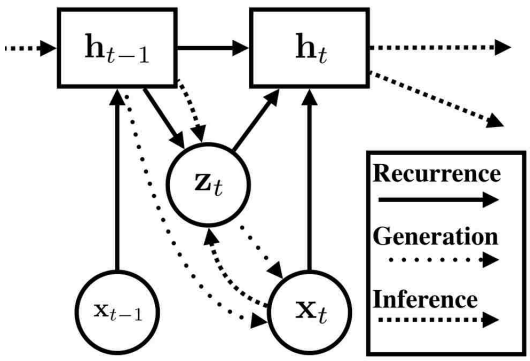

Figure 3.13: A generative RNN with Gaussian latent variables, $\mathbf{z}_t$. The latent variables are inferred through an approximate posterior, $q(\mathbf{z}_t|\mathbf{x}_{0:t})$.

where $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_T)$ is a sequence of persistent Gibbs samples on the RBM. This objective would require that one stores the persistent chain up to some length, T. However, the algorithm can be simplified by storing the last recurrent state, $\mathbf{h}_{t-1}$, from the RNN and performing single-step learning without back-propagating through time.

This approach is similar in intention to Hamiltonian Markov chain Monte Carlo (Hamiltonian MCMC, Neal et al., 2011), in that additional techniques are being used to more efficiently explore the manifold defined by some potential, in this case the RBM energy. However, where Hamiltonian MCMC relies on an explicit momentum variable to aid in mixing, the "momentum" here is embedded in the recurrent hidden states of the RNN. That said, the RNN needs to be trained to have certain momentum: if it is trained on slow mixing, it's effective sampling momentum will be small.

In order to encourage faster mixing on the RNN, RNN chains are allowed to "jump" between chains. A modified objective,

$$\frac{1}{M^2} \sum_k \sum_m \nabla \log q(\mathbf{x}_t^{(k)} | \mathbf{x}_{t-1}^{(m)}, \mathbf{h}_{t-1}), \tag{3.22}$$

for $k = 1 \dots M$, may encourage faster mixing, but may cause the chains to learn the average of all the chains. In order to learn multiple modalities, it is necessary to introduce

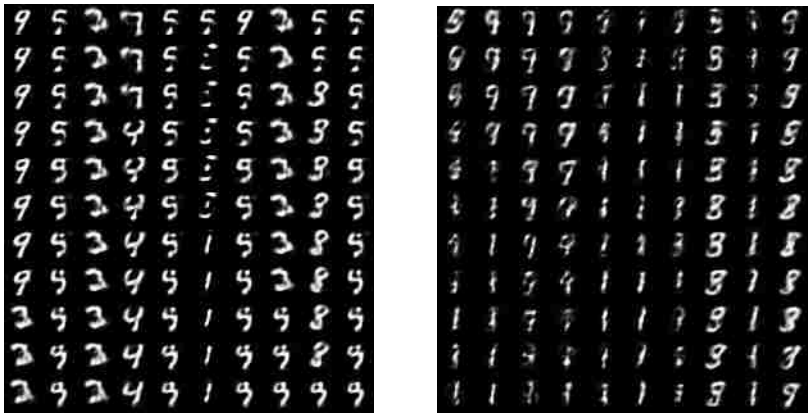

Figure 3.14: Left: RNN samples from an RNN trained on persistent Gibbs chains with random jumps between chains. Right: A persistent RBM Gibbs chain with random jumps between chains.

stochastic recurrent latent variables.

Drawing from recent advances in extending variational inference in directed graphical models to RNNs (Kingma & Welling, 2013; Chung, Kastner, Dinh, Goel, Courville, and Bengio, 2015) (See also Chapter 2.2 below), the transition is modeled with the addition of a Gaussian noise latent variable to the RNN chain (Figure 3.13). While inference of this sort is generally difficult, a simple approximate posterior, $q(\mathbf{z}|\mathbf{x})$ along with continuous latent variables can be easily trained and incorporated into the RNN. Many parameterization for the prior, $p(\mathbf{z})$, the conditional, $p(\mathbf{x}|\mathbf{z})$, and the approximate posterior, $q(\mathbf{z}|\mathbf{x})$, exist, but a simple parameterization provided in Figure 3.13 should suffice.

Our initial experiments showed that sampling from the uniform mixing in Equation 3.22 is too fast to effectively train an RNN. Further experiments showed that a modestly sized RNN with equal number of hidden units and latent variables as an RBM with 200 hidden units trained on RBM could learn from 10 RBM persistent Gibbs chain when each chain skipped to another chain with a probability of 0.1 at each step. An example of samples drawn from an RNN trained to quickly traverse the RBM energy manifold are

provided in Figure 3.14. For the persistent chains, the "jumps" correspond to when the chains were shuffled for RNN training, and are not representative of the actual PCD chains. While the RNN tended to fall into spurious modes (in this case of samples resembling 1s), the mixing is faster than with RBMs, yet still smooth and somewhat interpretable. These preliminary results indicate that RNNs may be able to efficiently explore the space defined by the RBM energy, providing a parametric alternative to methods for improving MCMC sampling.

Chapter 4

Directed Graphical Models, Applications, and Extensions

While our work on separating intrinsic networks (INs) with restricted Boltzmann machines was successful, further extensions and applications are limited due to challenges in training deeper undirected models. Our initial experiments with deep Boltzmann machines (DBMs) to learn hierarchical representations of INs had limited success, perhaps due to some of the issues covered above. Furthermore, in contrast to RBMs, which have exact inference, interpreting the latent structure in DBMs requires expensive Gibbs sampling procedures, heightening what was already a difficult challenge in analysis. IN analysis may be realizable with better sampling and estimation techniques, and this is left for future research.

I direct the reader's attention instead to directed graphical models, which are in many cases easier to train and evaluate, particularly with high-dimensional data. This chapter will focus on a type of directed model called a *directed belief network*, which is a generative directed graphical model consisting of a conditional density $p(\mathbf{x}|\mathbf{h})$ and a prior $p(\mathbf{h})$, such that the joint density can be expressed as a their product, $p(\mathbf{x}, \mathbf{h}) = p(\mathbf{x}|\mathbf{h})p(\mathbf{h})$. In

particular, the point density factorizes into a hierarchy of conditional densities: $p(\mathbf{x}, \mathbf{h}) = p(\mathbf{x}, \mathbf{h}_1, \dots \mathbf{h}_L) = p(\mathbf{h}_L)p(\mathbf{x}|\mathbf{h}_1)\prod_{l=1}^{L-1} p(\mathbf{h}_l|\mathbf{h}_{l+1})$, where $p(\mathbf{h}_l|\mathbf{h}_{l+1})$ is the conditional density at the l-th layer and $p(\mathbf{h}_L)$ is a prior distribution of the top layer. Sampling from the model can be done simply via ancestral-sampling, first sampling from the prior, then subsequently sampling from each layer until reaching the observation, $\mathbf{x}$. This latent variable structure can improve model capacity, but inference can still be intractable, as is the case in sigmoid belief networks (SBN, Neal, 1992), deep belief networks (DBN, Hinton et al., 2006), deep autoregressive networks (DARN, Gregor, Danihelka, Mnih, Blundell, and Wierstra, 2013), and other models in which each of the conditional distributions involves complex nonlinear functions.

4.1 Variational Lowerbound of a Directed Belief Network

The objective considered here is the likelihood function, $p(\mathbf{x}; \boldsymbol{\phi})$, where $\boldsymbol{\phi}$ represents parameters of the generative model (e.g. a directed belief network). Estimating the likelihood function given the joint distribution, $p(\mathbf{x}, \mathbf{h}; \boldsymbol{\phi})$, above is not generally possible as it requires intractable marginalization over $\mathbf{h}$. Instead, it is useful to introduce an approximate posterior, $q(\mathbf{h}|\mathbf{x})$, as a proposal distribution. In this case, the log-likelihood can be bounded from below:[1]

$$\begin{aligned}\log p(\mathbf{x}) &= \log \sum_{\mathbf{h}} p(\mathbf{x}, \mathbf{h}) \geq \sum_{\mathbf{h}} q(\mathbf{h}|\mathbf{x}) \log \frac{p(\mathbf{x}, \mathbf{h})}{q(\mathbf{h}|\mathbf{x})} \\ &= \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\left[\log \frac{p(\mathbf{x}, \mathbf{h})}{q(\mathbf{h}|\mathbf{x})}\right] := \mathcal{L}_1, \end{aligned} \tag{4.1}$$

where I introduce the subscript in the lower bound to make the connection to importance sampling in the following chapter. The bound is tight (e.g., $\mathcal{L}_1 = \log p(\mathbf{x})$) when the KL divergence between the approximate and true posterior is zero

[1]For clarity of presentation, I will often omit dependence on parameters $\boldsymbol{\phi}$ of the generative model, so that $p(\mathbf{x}, \mathbf{h}) = p(\mathbf{x}, \mathbf{h}; \boldsymbol{\phi})$

(e.g., $D_{KL}(q(\mathbf{h}|\mathbf{x})||p(\mathbf{h}|\mathbf{x})) = 0$). The gradients of the lower bound w.r.t. the generative model can be approximated using the Monte Carlo approximation of the expectation:

$$\nabla_{\boldsymbol{\phi}} \mathcal{L}_1 \approx \frac{1}{K} \sum_{k=1}^{K} \nabla_{\boldsymbol{\phi}} \log p(\mathbf{x}, \mathbf{h}^{(k)}; \boldsymbol{\phi}), \quad \mathbf{h}^{(k)} \sim q(\mathbf{h}|\mathbf{x}). \tag{4.2}$$

4.2 Helmholtz machines

The success of variational inference relies on the choice of approximate posterior, as poor choice can result in a looser variational bound. A deep feed-forward *recognition network* parameterized by $\boldsymbol{\psi}$ has become a popular choice, so that $q(\mathbf{h}|\mathbf{x}) = q(\mathbf{h}|\mathbf{x}; \boldsymbol{\psi})$, as it offers fast and flexible inference (see, e.g., Salakhutdinov & Larochelle, 2010; Kingma & Welling, 2013; Mnih and Gregor, 2014; Rezende et al., 2014). Generally known as a "Helmholtz machine" (Dayan et al., 1995), these approaches often require additional tricks to train, as the naive Monte Carlo gradient of the lower bound w.r.t. the variational parameters has high variance.

For Helmhotz machines with continuous latent variables, however, training via back-propagation is possible with the help of a re-parameterization, an approach known as variational autoencoders (VAE, Kingma & Welling, 2013). In particular, a normally distributed random variable with mean $\boldsymbol{\mu}$ and diagonal covariance vector, $\boldsymbol{\sigma}$, $\mathbf{h} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$ can be re-parameterized as:

$$\mathbf{h} = \boldsymbol{\mu} + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}; \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}). \tag{4.3}$$

This type of re-parameterization is available for a number of continuous distributions, such as Gaussian, Laplace, logistic, Poisson, and Gumbel, and can be used to effectively train directed belief networks with continuous latent variables (Rezende et al., 2014).

4.3 Variational Autoencoders for Feature Detection of Magnetic Resonance Imaging Data

4.3.1 Re-parameterization of logistic distribution

We demonstrate variational autoencoders for separating regions of interest as features from magnetic resonance imaging (MRI) data (Hjelm et al., 2016c). Let us assume that the conditional distribution is a Gaussian distribution with mean and diagonal covariance $(\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x)$, and the approximate posterior has a Logistic distribution with mean, $\boldsymbol{\mu}_h$, and scale, $\mathbf{s}_h$. The conditional parameters, $(\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x)$, and the variational parameters, $(\boldsymbol{\mu}_h, \mathbf{s}_h)$ are outputs of FFNs, f and g, with parameters $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ respectively:

$$
\begin{aligned}
(\boldsymbol{\mu}_x(\mathbf{h};\boldsymbol{\theta}), \boldsymbol{\sigma}_x(\mathbf{h};\boldsymbol{\theta})) &= g(\mathbf{h};\boldsymbol{\theta}), \\
(\boldsymbol{\mu}_h(\mathbf{x};\boldsymbol{\psi}), \mathbf{s}_h(\mathbf{x};\boldsymbol{\psi})) &= f(\mathbf{x};\boldsymbol{\psi}),
\end{aligned} \tag{4.4}
$$

Finally, assume $p(\mathbf{h})$, the prior distribution of the latent variables, is a spherical multivariate Logistic distribution. The lower bound in Equation 4.1 becomes:

$$
\begin{aligned}
\mathcal{L}(\mathbf{x}) \approx \sum_{m=1}^{M} \Bigg[& \log p(\mathbf{x}|\mathbf{h}^{(m)};\boldsymbol{\theta}) \\
& + \sum_{i=1}^{N} \left(h_i^{(m)} - 2\log\left(1+\exp(h_i^{(m)})\right) - \frac{h_i^{(m)} - \mu_h(\mathbf{x};\boldsymbol{\psi})_i}{s_h(\mathbf{x};\boldsymbol{\psi})_i} \right. \\
& \left. + \log s_h(\mathbf{x};\boldsymbol{\psi})_i + 2\log\left(1+\exp\left(\frac{h_i^{(m)} - \mu_h(\mathbf{x};\boldsymbol{\psi})_i}{s_h(\mathbf{x};\boldsymbol{\psi})_i}\right)\right)\right) \Bigg].
\end{aligned} \tag{4.5}
$$

In order to make the gradient of the first term above w.r.t the variational parameters, $\boldsymbol{\psi}$, possible, the following re-parameterization allows for training with back-propagation:

$$
\mathbf{h} = \boldsymbol{\mu}_h + \log\left(\frac{\boldsymbol{\epsilon}}{1-\boldsymbol{\epsilon}}\right) \odot \mathbf{s}_h, \ \boldsymbol{\epsilon} \sim \mathcal{U}(\mathbf{0}, \mathbf{1}),
$$

where $\mathcal{U}(\mathbf{0}, \mathbf{1})$ is a uniform distribution with range $[0, 1]$.

4.3.2 Visualizing latent variables

Visualizing a latent variable, h_i, of a directed belief network involves calculating the marginal over all other latent variables: $p(\mathbf{x}|h_i) = \sum_{\mathbf{h}_{j\neq i}} p(\mathbf{x}, \mathbf{h}_{j\neq i}|h_i)$. This is computationally intractable with most configurations. Alternatively, we can draw M samples from the approximate posterior, $\mathbf{h}_{j\neq i}^{(m)} \sim q(\mathbf{h}_{j\neq i}|\mathbf{x})$ to approximate the marginal:

$$p(\mathbf{x}|h_i) \approx \sum_{m=1}^{M} \frac{p(\mathbf{x}, \mathbf{h}_{j\neq i}|h_i)}{q(\mathbf{h}_{j\neq i}|\mathbf{x})}. \tag{4.6}$$

However, this approximation typically requires a large number of samples to be accurate (e.g. $O(100,000)$ with the MNIST dataset). In addition, this only provides a single point in the marginal, which is a continuous function of h_i. In reality, we are interested in how changes in h_i effect generation of the image. Therefore, we use the following fast approximation to determine the "projection" of the ith latent variable:

$$\Delta p(\mathbf{x}|h_i)/\Delta h_i \approx p(\mathbf{x}|h_i = \mu_i + s_i, \mathbf{h}_{j\neq i} = \boldsymbol{\mu}_{j\neq i}) - p(\mathbf{x}|\mathbf{h} = \boldsymbol{\mu}), \tag{4.7}$$

where μ_i and s_i is reserved for parameters of the prior distribution that encode first and second order statistics respectively. For instance, for a Logistic distribution, μ_i would be the center of unit i and s_i would be the scale factor. This approximation does not capture the full generative effect of the latent variables, but it is sufficient for this demonstration.

4.3.3 Experimental setup

For our medical imaging study, we used the MRI dataset from the combined schizophrenia studies in Plis et al. (2013). Whole brain MRIs were obtained on a 1.5T Signa GE scanner using identical parameters and software, and the resulting dataset was segmented into grey matter regions with 60465 voxels in each sample. For quality control, the correlation coefficient of each MRI volume was calculated and any volumes with mean coefficient of

2 standard deviations below the mean across all volumes were categorized as noisy and removed. The resulting dataset had 163 subjects and 156 healthy controls.

For our generative model, we used a logistic prior, $p(\mathbf{h}; \boldsymbol{\phi})$, with 64 units and a 2-layer "generation" feed-forward network (FFN) with a deterministic intermediate layer of 500 softplus ($\log(1 + \exp(x))$) units to parameterize a Gaussian conditional distribution, $p(\mathbf{x}|\mathbf{h}; \boldsymbol{\phi})$. Our approximate posterior, $q(\mathbf{h}|\mathbf{x}; \boldsymbol{\psi})$, was a multivariate factorized logistic which was parameterized by a 2-layer "recognition" FFN with 500 hyperbolic tangent ($\tanh$) deterministic units. We learn $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ by maximizing the variational lower bound, and trained our model with a batch size of 10 using the RMSprop algorithm (Hinton, 2012) for 1000 epochs.

4.3.4 Results

As the latent variable projections were both positive and negative and the prior distribution is symmetric with respect to our choice of positive scale factors, we reversed the sign of our projections if the mean of voxels above 2 standard deviations was negative. For each latent variable or "component", we calculate the approximate posterior for each subject, $q(h_i|\mathbf{x}_n)$, and then used logistic regression to schizophrenia using the approximate posterior means, $\boldsymbol{\mu}_h(\mathbf{x}_n)$. Each component was tested for significance by using the resulting β values from the logistic regression in a one-sample t-test.

Visual inspection of the latent variables revealed a diverse set of features that were mostly identifiable as regions of interest, with very little noisy features. There was significantly more overlap between features than is typical with ICA with PCA preprocessing or RBM with MRI data (Hjelm et al., 2014), which may or may not be beneficial depending on the research setting. Latent variables that showed high significance to schizophrenia ($p < 0.001$) are shown in Figure 4.1.

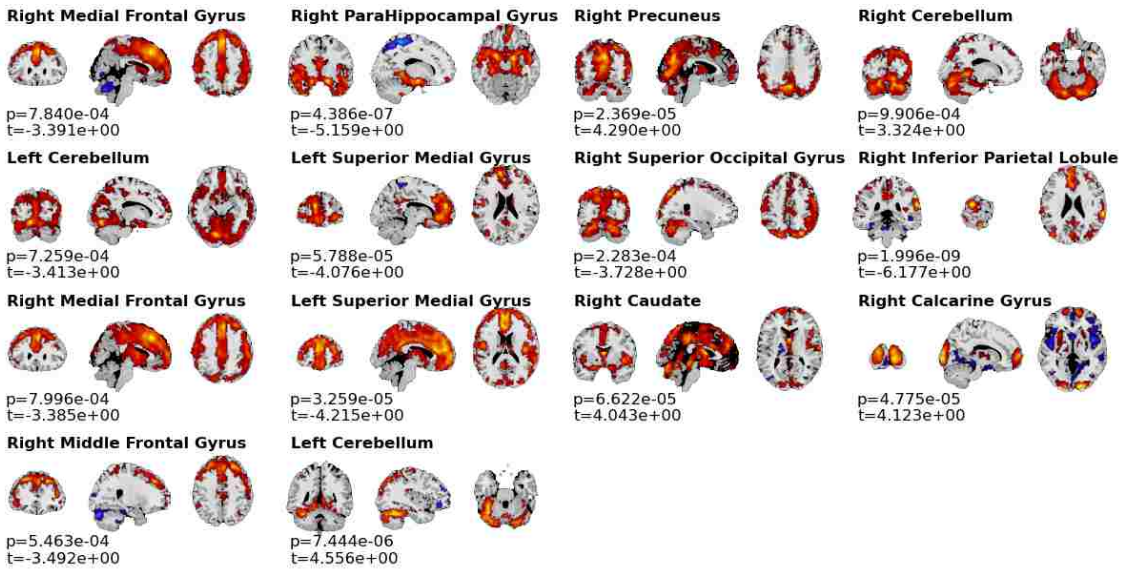

Figure 4.1: Projections of logistic latent variables for a variational autoencoder with an additional deterministic nonlinearity for the generative and recognition networks. Each projection was thresholded at 2 standard deviations, and the latent variables showed here are those that showed high significance ($p < 0.001$) from a one-sampled t-test of the β values from logistic regression to schizophrenia.

The means of the approximate posterior, $\boldsymbol{\mu}_h(\mathbf{x}_n)$, were used as input to a classification task, using simple logistic regression and 100-fold class-balanced cross validation. The resulting classification rate, 0.67, is significantly above chance. The conclusion is that, despite lacking information about the labels in the MLE objective and much lower dimensionality, the latent variables have a similar amount of information necessary to perform diagnosis. This is also apparent in the correlation matrix in Figure 4.2 using the components that showed high significance to schizophrenia. Finally, the components were grouped by calculating the correlation of approximate posterior centers across subjects. Figure 4.3 shows several groupings, as well as some inter-group relationships.

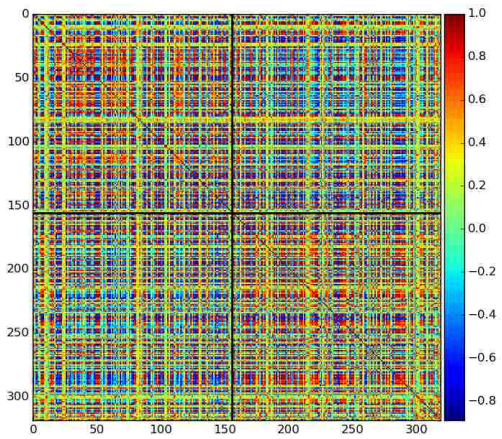

Figure 4.2: The correlation matrix of the logistic centers of the approximate posterior for each subject, $\boldsymbol{\mu}_q(\mathbf{x}_n)$ for components with high significance ($p < 0.001$) to schizophrenia. Columns and rows have been ordered to show healthy controls first, followed by patients, with higher inter-group correlation.

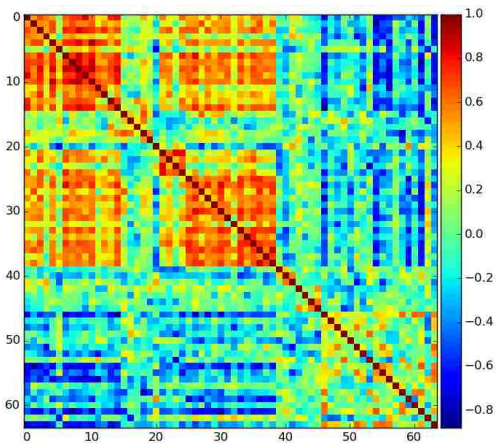

Figure 4.3: Correlation matrix between all components across subjects. Rows and columns were ordered according to grouping determined by a community multi-level analysis Blondel et al. (2008) and shows inter- and intra-group structure between components.

4.4 Recurrent Neural Networks for Spatiotemporal Dynamics of Intrinsic Networks from fMRI Data

Our work on directed graphical models with magnetic imaging data (MRI) uses only a single layer of independent latent variables inferred from and to generate a single mode (e.g. MRI) of data. VAEs with multiple layers of latent variables are realized as deep latent Gaussian models (DLGM, Rezende et al., 2014), which may reveal hierarchical structure in MRI. Furthermore, multimodal extensions of VAE should allow for in-depth analysis of the relationship between modes, such as structural and genetics data.

A notable limitation of basic VAEs is the inability to infer latent structure from sequential or time-series data, such as functional magnetic imaging (fMRI) data. It is difficult to model temporal dependencies with simple VAEs, so dynamics in time-series data are

usually extrapolated with additional analysis. Recurrent neural networks (RNNs) that incorporate stochastic latent variables along with VAE-type learning exist and are successful in language studies (Chung et al., 2015). However, models that rely on the approximate inference can exhibit high variance and may need a large number of samples to train, depending on the specific problem. This may be a challenge for applying variational methods to fMRI data, which tends to be very high-dimensional with relatively few samples. Luckily, it is not necessary to use approximate methods of inference to train or use RNNs to model temporal dynamics from latent structure.

Independent component analysis (ICA, Bell & Sejnowski, 1995), which was introduced in Section 3.1, is a shallow directed graphical model [2] with exact inference and does not require estimates of the log-likelihood function to train. ICA is trainable through one of many optimization routines that maximize non-Gaussianity or minimize mutual information (Hyvärinen and Oja, 2000). ICA is commonly used for separating intrinsic networks (INs, Biswal et al., 1995) as latent structure, which are important outcomes of fMRI studies and are central for understanding brain function and making diagnoses (Calhoun et al., 2001; Allen et al., 2012; Hjelm et al., 2014). We show here that the ICA objective is compatible with a generative RNN, so that we can learn a latent variable model of sequences that works with small high-dimensional datasets, such as fMRI.

4.4.1 Independent Component Analysis

ICA hypothesizes that the observed data is generated from a linear mixture of independent sources: $\mathbf{x}_n = \sum_i s_{n,i} M^i$, where $\mathbf{s}_n = \{s_{n,i}\}$ are K sources or *components* and M^i are the columns of a mixing matrix, $\mathbf{M}$. ICA constrains the sources to be independent, usually by specifying each component is drawn independently from separate distributions with high kurtosis. This framework presupposes any specific definition of component independence,

[2] ICA is effectively the noiseless case of a shallow directed graphical model.

and algorithms typically fall under two primary families (Hyvärinen & Oja, 2000).

In the INFOMAX algorithm (Bell & Sejnowski, 1995), the model is parameterized by an unmixing matrix $\mathbf{W} = \mathbf{M}^{-1}$, and independence is reached by minimizing the mutual information of the sources. Minimizing the mutual information of the sources is accomplished by enforcing their joint prior density factorizes, or $p_s(\mathbf{s}_n) = \prod_{i=1}^{K} p_{s_i}(s_{n,i})$. With this objective, independence is not guaranteed under a linear mixing model unless the sources follow a non-Gaussian distribution. Minimizing mutual information is the same as maximizing the log-likelihood of the marginal density, $p_x(\mathbf{x}_n)$, transformed by $f(\mathbf{x}_n) = \mathbf{W}\mathbf{x}_n$:

$$\log p_x(\mathbf{x}_n) = \log p_s(\mathbf{W}\mathbf{x}_n) + \log |\det \mathbf{W}|, \tag{4.8}$$

where $|\det \mathbf{W}| = |\det \mathbf{J}_f(\mathbf{x}_n)|$ is the Jacobian. As the transformation has a tractable inverse and Jacobian, the MLE objective does not suffer additional variance typical in methods that rely on approximate inference. This relieves some of the challenges associates with small high-dimensional datasets.

INFOMAX as applied to fMRI data assumes that the joint density of the time-series, $\mathbf{X}_n = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, factorizes into a product of marginals, or $p_x(\mathbf{X}_n) = \prod_t p_x(\mathbf{x}_{t,n}) = \prod_{t=1}^{T} p_s(\mathbf{W}\mathbf{x}_{t,n}) |\det \mathbf{W}|$. Generating a sequence can be done from an ordered set of sources using the inverse of the unmixing matrix. However, ICA, as with other popular linear methods for separating INs, is order-agnostic in time: each multivariate signal at each time step is treated as i.i.d.. While model degeneracy in time is convenient for learning, as an assumption about the data, the explicit lack of temporal dependence limits generation to unordered samples from a marginal density. In addition, ICA uses the same parameterization across subjects with fMRI data, which allows for either temporal or spatial variability, but not both. The consequence of this is that ICA cannot represent variation of shape in INs while also representing variation in the time courses. This may encourage ICA to exaggerate time course statistics, as any significant variability in shape or size must be accounted for by modulation of the sources in order to satisfy the algorithm's objective

function.

Despite these drawbacks, the benefits of using ICA for analyzing fMRI data is strongly evident in numerous studies, to the extent that has become the dominant approach for separating intrinsic networks and connectivity studies (Zuo et al., 2010; Kim et al., 2008; Allen et al., 2012; Damoiseaux et al., 2006; Calhoun et al., 2008; Smith et al., 2009). In order to overcome shortcomings in temporal dynamics and subject/temporal variability but without abandoning the fundamental strengths of ICA, we extend ICA to model sequences using RNNs. The resulting model naturally represents temporal dynamics through a sequential ICA objective, has subject and temporally-specific spatial maps, and is easily trainable using gradient descent and back-propagation.

4.4.2 RNN ICA

The RNN framework for sequence modeling we introduced in Section 3.2.1 can easily be extended to incorporate the INFOMAX objective. Define, as with ICA, a linear transformation for each observation to source configuration: $\mathbf{s}_{t,n} = \mathbf{W}\mathbf{x}_{t,n}$, and define a high-kurtosis and factorized source distribution, $p_{s_{t,n}}(\mathbf{s}_{t,n})$ (such as logistic or Laplace) for each time step, t, and each fMRI sequence, n. Let us apply the transformation to an fMRI time series: $\mathbf{s}_{1:T,n} = f(\mathbf{x}_{1:T,n}) = (\mathbf{W}\mathbf{x}_{1,n}, \mathbf{W}\mathbf{x}_{2,n}, \dots \mathbf{W}\mathbf{x}_{T,n})$. The log-likelihood function can be re-parameterized as:

$$\begin{aligned}
\log p(\mathbf{x}_{1:T,n}) &= \log p(\mathbf{x}_{1,n}) + \sum_{t=2}^{T} \log p(\mathbf{x}_{t,n}|\mathbf{x}_{1:t-1,n}) \\
&= \log p_{s_1}(\mathbf{W}\mathbf{x}_1) + \sum_{t=2}^{T} \log p_{s_{t,n}}(\mathbf{W}\mathbf{x}_{t,n}|\mathbf{x}_{1:t-1,n}) + T \log |\det \mathbf{J}_f(\mathbf{x}_{1:T,n})| \\
&= T \log |\det \mathbf{W}| + \log p_{s_{1,n}}(\mathbf{W}\mathbf{x}_{1,n}) + \sum_{t=2}^{T} \log p_{s_{t,n}}(\mathbf{W}\mathbf{x}_{t,n}|\mathbf{x}_{1:t-1,n}),
\end{aligned} \tag{4.9}$$

where $\mathbf{J}_f$ is the Jacobian and the source distribution, $p_{s_{t,n}}$, has parameters determined by the recurrent units $\mathbf{h}_{t,n}$ (see Equations 3.14 and 3.15). A high-kurtosis distribution is

desirable to ensure independence of the sources (or minimizing the conditional mutual information, e.g., the INFOMAX objective (Bell & Sejnowski, 1995)), so a reasonable choice for the outputs of the RNN at each time step are the mean, $\boldsymbol{\mu}$, and scale, $\boldsymbol{\sigma}$, for a logistic distribution:

$$\boldsymbol{\mu}_{t,n} = \mathbf{W}_{\mu}\mathbf{h}_{t,n}; \qquad \boldsymbol{\sigma}_{t,n} = \mathbf{W}_{\sigma}\mathbf{h}_{t,n}. \tag{4.10}$$

Our treatment assumes the ICA weight matrix, $\mathbf{W}$, is square, which is necessary to ensure a tractable determinant Jacobian and inverse. fMRI data is very high dimensional, so to reduce the dimensionality of the necessarily square matrix we must resort to some sort of dimensionality reduction as preprocessing. Widely used in ICA studies of fMRI is principle component analysis (PCA), which can be used to reduce the dimensionality of the data, selecting number of components to match the dimension of the sources, $\mathbf{s}_{t,n}$.

4.4.3 Time-dependent Spatial Maps

The treatment of RNN ICA above uses the same ICA weight matrix across subjects and time. The Jacobian from the likelihood function reduces to the ICA weight matrix: the cross-time terms of the determinant are all zero as the complete Jacobian is block upper triangular in time. This makes defining the objective and training straightforward, with computing the determinant of the Jacobian having similar complexity as ICA.

While the model is already novel, we can further leverage this directed dependence in the transformation to represent variable spatial maps across time and subjects. Consider a transformation, $\mathbf{s}_{t,n} = \mathbf{W}_{t,n}\mathbf{x}_{t,n}$, such that $\mathbf{W}_{t,n}$ is a subject-specific weight matrix at time t with form:

$$\mathbf{W}_{t,n} = \overline{\mathbf{W}} + \widehat{\mathbf{W}}_{t,n} = \overline{\mathbf{W}} + \mathbf{c}_{t,n}\mathbf{p}_{t,n}^{\top}, \tag{4.11}$$

where $\widehat{\mathbf{W}}_{t,n} = \mathbf{c}\mathbf{p}^{\top}$ is a rank-1 weight matrix composed of two equal-length vectors and $\overline{\mathbf{W}}$ is a weight matrix that is the same across time and subjects. To assure that the weights

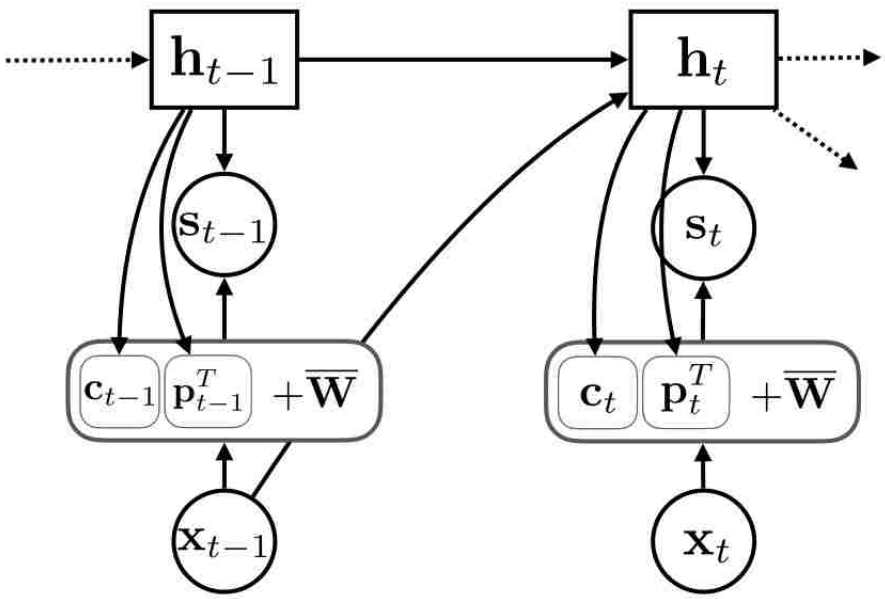

Figure 4.4: RNN ICA for a single subject. The RNN is fed observations, $\mathbf{x}_t$, and the outputs are the parameters for the sources, $\mathbf{s}_t$. The RNN is trained to maximize the likelihood of the model parameters under the data transformed by a square matrix: $\mathbf{s}_t = (\overline{\mathbf{W}} + \mathbf{c}_t\mathbf{p}_t^\top)\mathbf{x}_t$, where $\overline{\mathbf{W}} + \mathbf{c}_t\mathbf{p}_t^\top$ is square and $\mathbf{c}_t$ and $\mathbf{p}_t$ are outputs from feed forward networks with input $\mathbf{h}_t$. This allows for variable unmixing matrix with simple inverse and Jacobian computation for generation and MLE optimization.

are subject-specific and are trainable using back-propagation, the factors of the rank-1 matrix, $\mathbf{c}$ and $\mathbf{p}$, are modeled as outputs of two separate neural networks with input $\mathbf{h}_t$.

The choice of rank-1 form of the square variable weight term makes learning feasible in two important ways. First, the dimensionality of the two component vectors combined is far smaller than a full rank alternative. This avoids the problem of the variable weight matrix being underdetermined, a problem likely with fMRI data as the number of samples is relatively small. Second, both the determinant of the Jacobian and inverse of a rank-1 matrix added to a full rank square matrix are easy to compute (Sherman and Morrison,

1950):

$$\det(\mathbf{W}_{t,n}) = (1 + \mathbf{p}_{t,n}^{\top} \overline{\mathbf{W}}^{-1} \mathbf{c}_{t,n}) \det(\overline{\mathbf{W}}),$$
$$\mathbf{W}_{t,n}^{-1} = \overline{\mathbf{W}}^{-1} - \frac{\overline{\mathbf{W}}^{-1} \mathbf{c}_{t,n} \mathbf{p}_{t,n}^{\top} \overline{\mathbf{W}}^{-1}}{1 + \mathbf{p}_{t,n}^{\top} \overline{\mathbf{W}}^{-1} \mathbf{c}_{t,n}}. \tag{4.12}$$

The components of the rank-1 permutation, $\mathbf{c}_{t,n}$ and $\mathbf{p}_{t,n}$ depend on all previous input, $\mathbf{x}_{1:t-1,n}$, through a deterministic continuous function of the hidden states, $\mathbf{h}_{1:t,n}$. In general, dependence on the sequence across time would make computing the Jacobian difficult due to cross terms in time. It is easy to illustrate, however, the property that the determinant of the Jacobian reduces to one composed only of static terms. Consider the data and sources to be one-dimensional and our sequence to be of length three. The Jacobian then has the form,

$$\begin{bmatrix} \frac{\partial((\overline{\mathbf{W}}+\widehat{\mathbf{W}}_1(.))x_1)}{\partial x_1} & \frac{\partial((\overline{\mathbf{W}}+\widehat{\mathbf{W}}_2(x_1))x_2)}{\partial x_1} & \frac{\partial((\overline{\mathbf{W}}+\widehat{\mathbf{W}}_3(x_1,x_2))x_3)}{\partial x_1} \\ \frac{\partial((\overline{\mathbf{W}}+\widehat{\mathbf{W}}_1(.))x_1)}{\partial x_2} & \frac{\partial((\overline{\mathbf{W}}+\widehat{\mathbf{W}}_2(x_1))x_2)}{\partial x_2} & \frac{\partial((\overline{\mathbf{W}}+\widehat{\mathbf{W}}_3(x_1,x_2))x_3)}{\partial x_2} \\ \frac{\partial((\overline{\mathbf{W}}+\widehat{\mathbf{W}}_1(.))x_1)}{\partial x_3} & \frac{\partial((\overline{\mathbf{W}}+\widehat{\mathbf{W}}_2(x_1))x_2)}{\partial x_3} & \frac{\partial((\overline{\mathbf{W}}+\widehat{\mathbf{W}}_3(x_1,x_2))x_3)}{\partial x_3} \end{bmatrix}$$

$$= \begin{bmatrix} \overline{\mathbf{W}} + \widehat{\mathbf{W}}_1(.) & x_2 \frac{\partial \widehat{\mathbf{W}}_2(x_1)}{\partial x_1} & x_3 \frac{\partial \widehat{\mathbf{W}}_3(x_1,x_2)}{\partial x_1} \\ 0 & \overline{\mathbf{W}} + \widehat{\mathbf{W}}_2(x_1) & x_3 \frac{\partial \widehat{\mathbf{W}}_3(x_1,x_2)}{\partial x_2} \\ 0 & 0 & \overline{\mathbf{W}} + \widehat{\mathbf{W}}_3(x_1, x_2) \end{bmatrix},$$

which is upper-triangular, so that the determinant is the product of diagonal terms. For the general case, the Jacobian is block-triangular, so the determinant reduces to the product of determinants for each block on the diagonal (static terms).

4.4.4 Experiments

To demonstrate the properties and strengths of our model, we apply our method to task fMRI data as in our RBM studies in Section 3.1.

We used a simple RNN with 100 recurrent hidden units and a recurrent parameterization as in Equation 3.15, as we do not anticipate needing to model long range dependencies that necessitate gated models (Hochreiter & Schmidhuber, 1997). The initial hidden state of the RNN was a 2-layer feed forward network with 100 softplus $(\log(1 + \exp(x)))$ units using 20% dropout. Two separate 2-layer feed-forward networks with 60 softplus units were used to compute $\mathbf{c}_{t,n}$ and $\mathbf{p}_{t,n}$ from $\mathbf{h}_{t,n}$, with an additional L_2 decay cost with decay coefficient of 0.002 used in training. The model was trained using the RMSProp algorithm (Hinton, 2012) with a learning rate of 0.0001 for 1000 epochs. Training typically took on the order of an hour on an NVIDIA 980 Ti GPU, an Intel i7-$4700k$, and 32 GB of RAM.

Figure 4.5: Complete set of features found by RNN ICA with variable spatial maps, averaged across time and subject. Labels are the largest region of interest. p-values are from a one sample t-test of beta values from linear regression to the target stimulus.

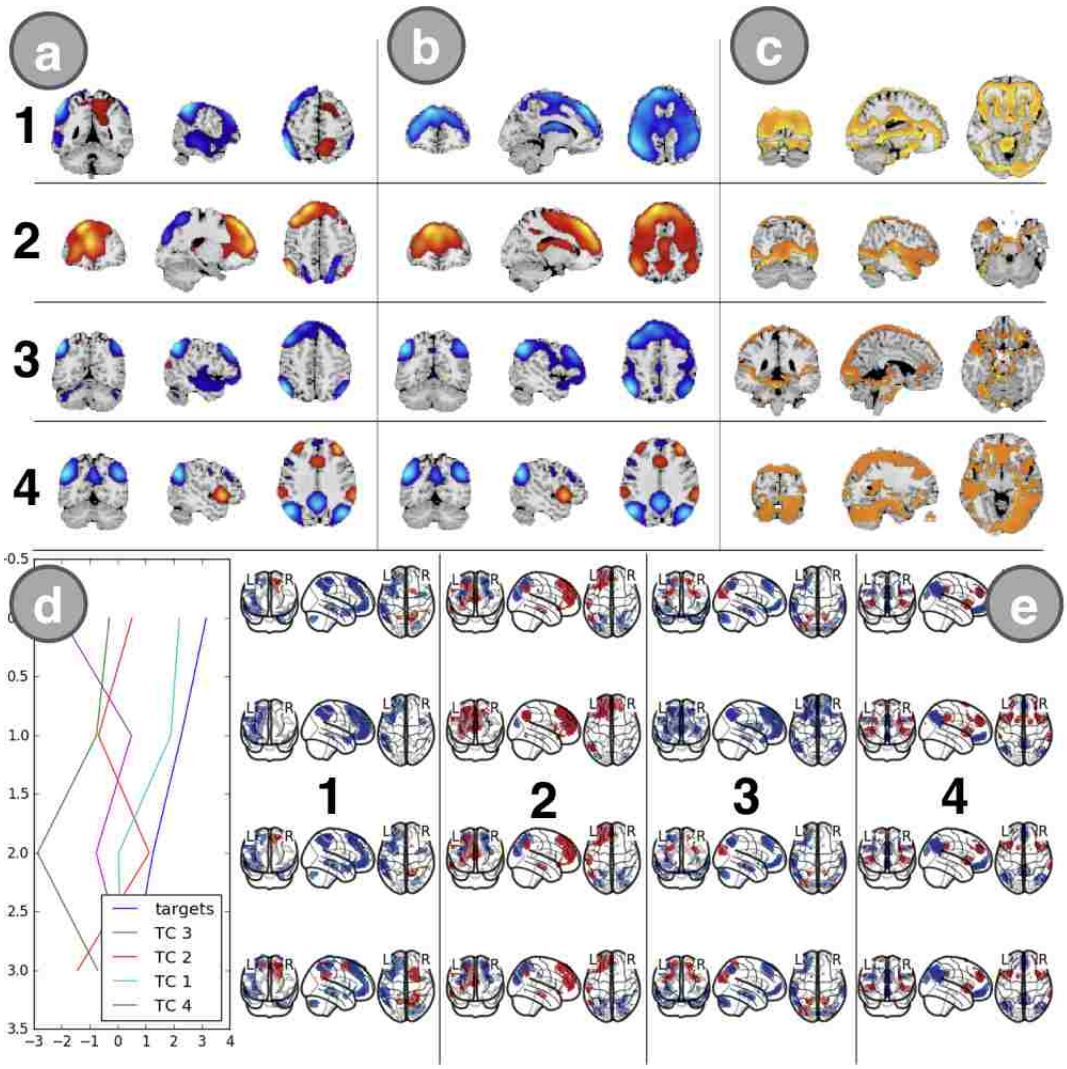

Figure 4.6: Five representative representative features with high sensitivity to the target stimulus. Labels are the largest region of interest. Above are the average spatial maps across time and subjects, back-reconstructed. p-values are from a one sample t-test of beta values from linear regression to the target stimulus. Below are four time steps in sequence of each feature for a single patient.

Our results indicate the model is able to separate interpretable and unique INs, with approximately 50 grey matter regions after filtering out white matter, ventricles, and well-

Table 4.1: Task-related features from RNN ICA with spatial variability with p values lower than 10^{-7}.

ID	Label	Targets	Novels
1	L Sup Frontal Gy	5.4e-13	2.2e-09
4	L Sup Frontal Gy	4.7e-08	
5	L Med Temporal Pole	4.6e-10	2.4e-13
6	L Sup Med Gy	2.6e-09	
8	R Postcentral Gy	5.3e-10	1.0e-09
9	R Precuneus		6.7e-07
11	R Mid Temporal Gy	6.6e-12	
15	L Insula	1.2e-07	
16	R Cingulate Gy	1.0e-08	
22	L Mid Frontal Gy	1.2e-14	
27	R Postcentral Gy	6.4e-19	
29	R Mid Temporal Gy	1.4e-10	8.8e-18
33	R Ant Cingulate	3.5e-08	
36	R Precuneus	9.5e-12	
39	R Lentiform Nucleus	7.5e-12	
41	L Cuneus	1.5e-10	
45	L Inf Parietal Lobule	1.7e-09	
48	R Precuneus		1.0e-06
51	L Angular Gy	2.1e-07	
53	L Inf Parietal Lobule	5.1e-11	
54	L Cerebellum	7.0e-11	
56	R Caudate		9.9e-09
58	L Caudate Nucleus	9.1e-09	

known motion artifacts (Figure 4.5, complete set is provided). Using the β values from ordinary least squares regression (OLS) for each subject to the target stimulus, we found that 20 of these features were highly sensitive to target according to a one-sample t-test ($p \leq 10^{-7}$, Table 4.1). Among these were motor gyrus and temporal gyrus features (Figure 4.6), which are widely found in the literature to be sensitive to task (Allen et al., 2012).

In addition, each of these features show spatial variability across time through their variable spatial maps which cannot be explained by scalar modulation of feature intensity. In order to show this, we performed linear regression on each column, $C_{t,n,m} = c_{t,n,m}\mathbf{p}_{t,n}^{\top}$

to the first principle component from PCA performed on $C_{t,n,m}$ for each independent component, m. We then used both the Bayes information criterion (BIC) and the Akaike information criterion (AIC) to evaluate fit for each component across time and subjects. The standard deviation for each component across subjects and across time was greater than 100 for both criteria, indicating poor fit. Furthermore, the spatial maps that show the worst fit (highest variance of BIC or AIC) are often those that showed sensitivity to task.

The time courses of the spatial maps, $\mathbf{c}_{t,n}\mathbf{p}_{t,n}^{\top}$, also show sensitivity to task, including significant group differences in the β values from linear regression to the target stimuli (Figure 4.7). Most surprisingly, the spatial maps show significant group differentiation ($p \leq 0.05$, FDR corrected), while none of the source time courses show significant group differences. This contrasts with findings from ICA fMRI task studies (Du, Calhoun, Li, Ma, Eichele, Kiehl, Pearlson, and Adali, 2012). This may indicate that ICA exaggerates source variation that is perhaps better accounted by IN shape, which is further supported by our finding that the spatial map variability cannot be explained by scalar modulation. However, further study is necessary to make any conclusions about spatial map variability and group differentiation.

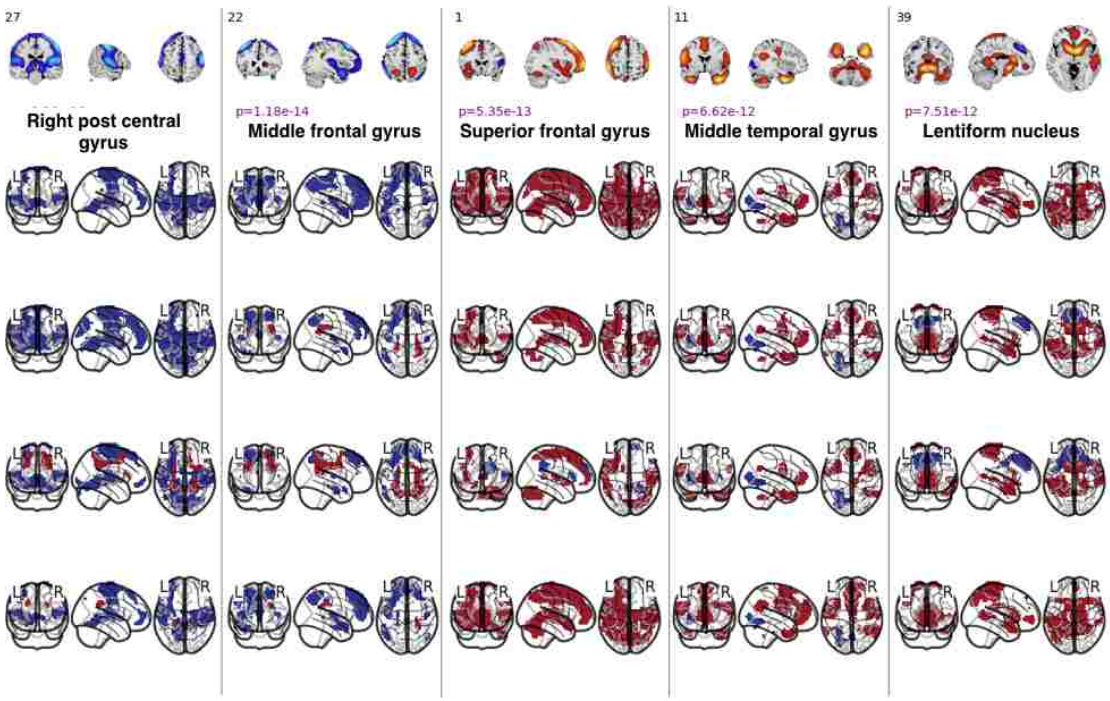

Figure 4.7: Four features whose spatial maps showed significant differences between healthy controls and groups. Differences were found from the betas from linear regression to the target stimulus. a) Average spatial maps back-reconstructed, b) The uniform spatial map, $\overline{\mathbf{W}}$, back-reconstructed, c) Mann-Whitney u values from significance test between healthy controls and patients, d) time courses and convolved target stimulus for e) four time steps of spatial maps for a single patient.

Chapter 5

Iterative Refinement Inference Techniques

In the previous chapter, we were able to show that variational autoencoders (VAE, Kingma & Welling, 2013) can be used to separate structural components from magnetic resonance imaging (MRI) data. VAEs, while extremely successful, relies on a re-parameterization of the latent variables to pass the learning signal to the recognition network. This type of parameterization, however, is not available with discrete units, and the naive Monte Carlo estimate of the gradient has too high variance to be practical (Dayan et al., 1995; Kingma & Welling, 2013).

However, good estimators are available through importance sampling (Bornschein and Bengio, 2014), input-dependent baselines (Mnih & Gregor, 2014), a combination baselines and importance sampling (Mnih and Rezende, 2016), and parametric Taylor expansions (Gu, Levine, Sutskever, and Mnih, 2015b). Each of these methods strive to be a lower-variance and unbiased gradient estimator. However, the reliance on the recognition network means that the quality of learning is bounded by the capacity of the recognition network, which in turn raises the variance. My colleagues and I demonstrate reducing the

variance of Monte Carlo based estimators by iteratively refining the approximate posterior provided by the recognition network (Hjelm et al., 2016a).

5.1 Discrete latent variables and improving the approximate posterior

We build on variational inference introduced in Section 4.1. The variational lower-bound in Eq. 4.1 is constrained by the assumptions implicit in the choice of approximate posterior, as the approximate posterior must be within the capacity of the recognition network and factorial. These assumptions can be relaxed by using an unbiased K-sampled importance-weighted estimate of the likelihood function
(see (Burda, Grosse, and Salakhutdinov, 2015) for details):

$$\mathcal{L}_1 \leq \mathcal{L}_K = \frac{1}{K}\sum_{k=1} \frac{p(\mathbf{x}, \mathbf{h}^{(k)})}{q(\mathbf{h}^{(k)}|\mathbf{x})} = \frac{1}{K}\sum_{k=1} w^{(k)} \leq p(\mathbf{x}), \tag{5.1}$$

where $\mathbf{h}^{(k)} \sim q(\mathbf{h}|\mathbf{x})$ and $w^{(k)}$ are the importance weights. This lower bound is tighter than the single-sample version provided in Eq. 4.1 and is an asymptotically unbiased estimate of the likelihood as $K \to \infty$.

The gradient of the lower bound w.r.t. the model parameters $\boldsymbol{\phi}$ is simple and can be estimated as:

$$\nabla_{\boldsymbol{\phi}} \mathcal{L}_K = \sum_{k=1}^{K} \tilde{w}^{(k)} \nabla_{\boldsymbol{\phi}} \log p(\mathbf{x}, \mathbf{h}^{(k)}; \boldsymbol{\phi}), \quad \text{where } \tilde{w}^{(k)} = \frac{w^{(k)}}{\sum_{k'=1}^{K} w^{(k')}}. \tag{5.2}$$

The estimator of the likelihood in Eq. 5.1 can reduce the variance of the gradients, $\nabla_{\boldsymbol{\psi}} \mathcal{L}_K$, but in general additional variance reduction is needed (Mnih & Rezende, 2016). Alternatively, importance sampling yields an estimate of the inclusive KL divergence, $D_{KL}(p(\mathbf{h}|\mathbf{x})||q(\mathbf{h}|\mathbf{x}))$, which can be used for training parameters $\boldsymbol{\psi}$ of the recognition network (Bornschein & Bengio, 2014). However, it is well known that importance sampling

can yield heavily-skewed distributions over the importance weights (Doucet, De Freitas, and Gordon, 2001), so that only a small number of the samples will effectively have non-zero weight. This is consequential not only in training, but also for evaluating models when using stochastic approximation of Eq. 5.1 to estimate test log-probabilities, which requires drawing a very large number of samples ($N \geq 100,000$ in the literature for models trained on MNIST (Gregor et al., 2013)).

The effective samples size, $\mathbf{n}_e$, of importance-weighted estimates increases and is optimal when the approximate posterior matches the true posterior:

$$\mathbf{n}_e = \frac{\left(\sum_{k=1}^{K} w^{(k)}\right)^2}{\sum_{k=1}^{K}(w^{(k)})^2} \leq \frac{\left(\sum_{k=1}^{K} p(\mathbf{x}, \mathbf{h}^{(k)})/p(\mathbf{h}^{(k)}|\mathbf{x})\right)^2}{\sum_{k=1}^{K}\left(p(\mathbf{x}, \mathbf{h}^{(k)})/p(\mathbf{h}^{(k)}|\mathbf{x})\right)^2} \leq \frac{(Kp(\mathbf{x}))^2}{Kp(\mathbf{x})^2} = K. \quad (5.3)$$

Conversely, importance sampling from a poorer approximate posterior will have lower effective sampling size, resulting in higher variance of the gradient estimates. In order to improve the effectiveness of importance sampling, we need a method for improving the approximate posterior from those provided by the recognition network.

Learning difficulties aside, using a recognition network for the posterior distribution couples estimates for different data points through global variational parameters, which helps the model generalize better and avoid local-minima problems common with traditional variational inference. Depending on the structure of the recognition network, this can place a global constraint on each posterior distribution, leading to poorer fit which can loosen the variational lower-bound. The posterior estimates may be inaccurate, so that the data can appear to fit the model worse than it really does. In the case of VAEs, using a non-factorized posterior can greatly improve learning and the quality of the approximate posterior (Rezende and Mohamed, 2015; Salimans, Kingma, and Welling, 2015a). Therefore, during learning we need a sufficiently complex recognition network, the limits of which are not well understood.

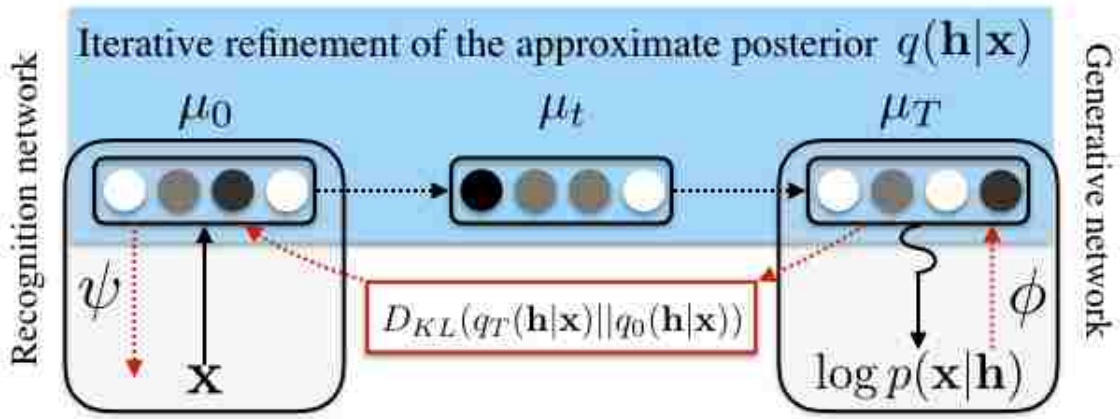

Figure 5.1: Iterative refinement for variational inference. An initial estimate of the variational parameters is made through a recognition network. The variational parameters are then updated iteratively, maximizing the lower bound. The final approximate posterior is used to train the generative model by sampling. The recognition network parameters are updated using the KL divergence between the refined posterior q_k and the output of the recognition network q_0.

5.2 Iterative refinement of the approximate posterior

To address the above issues, iterative refinement for variational inference (IRVI) uses the recognition network as a preliminary guess of the posterior, then refines the posterior through iterative updates of the variational parameters. For the refinement step, IRVI uses a stochastic transition operator, $g(.)$, that maximizes the variational lower-bound.

An overview of IRVI is available in Figure 5.1. For the expectation (E)-step, we feed the observation $\mathbf{x}$ through the recognition network to get the initial parameters, $\boldsymbol{\mu}_0$, of the approximate posterior, $q_0(\mathbf{h}|\mathbf{x};\boldsymbol{\psi})$. We then refine $\boldsymbol{\mu}_0$ by applying T updates to the variational parameters, $\boldsymbol{\mu}_{t+1} = g(\boldsymbol{\mu}_t, \mathbf{x})$, iterating through T parameterizations $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_T$ of the approximate posterior $q_t(\mathbf{h}|\mathbf{x})$.

With the final set of parameters, $\boldsymbol{\mu}_T$, the gradient estimate of the recognition parame-

ters $\boldsymbol{\psi}$ in the maximization (M)-step is taken w.r.t the negative exclusive KL divergence:

$$-\nabla_{\boldsymbol{\psi}} D_{KL}(q_T(\mathbf{h}|\mathbf{x})||q_0(\mathbf{h}|\mathbf{x};\boldsymbol{\psi})) \approx \frac{1}{K}\sum_{k=1}^{K} \nabla_{\boldsymbol{\psi}} \log q_0(\mathbf{h}^{(k)}|\mathbf{x};\boldsymbol{\psi}), \tag{5.4}$$

where $\mathbf{h}^{(k)} \sim q_T(\mathbf{h}|\mathbf{x})$. Similarly, the gradients w.r.t. the parameters of the generative model $\boldsymbol{\phi}$ follow Eqs. 4.2 or 5.2 using samples from the refined posterior $q_T(\mathbf{h}|\mathbf{x})$. As an alternative to Eq. 5.4, we can maximize the negative inclusive KL divergence using the refined approximate posterior:

$$-\nabla_{\boldsymbol{\psi}} D_{KL}(p(\mathbf{h}|\mathbf{x})||q_0(\mathbf{h}|\mathbf{x};\boldsymbol{\psi})) \approx \sum_{k=1}^{K} \tilde{w}^{(k)} \nabla_{\boldsymbol{\psi}} \log q_0(\mathbf{h}^{(k)}|\mathbf{x};\boldsymbol{\psi}). \tag{5.5}$$

The form of the IRVI transition operator, $g(\boldsymbol{\mu}_t, \mathbf{x})$, depends on the problem. In the case of continuous variables, we can make use of the VAE re-parameterization with the gradient of the lower bound in Eq. 4.1 for our refinement step (see supplementary material). However, as this is not available with discrete units, we take a different approach that relies on adaptive importance sampling.

5.2.1 Adaptive Importance Refinement (AIR)

Adaptive importance sampling (AIS, Oh and Berger, 1992) provides a general approach for iteratively refining the variational parameters. For Bernoulli distributions, we observe that the mean parameter of the true posterior, $\hat{\boldsymbol{\mu}}$, can be written as the expected value of the latent variables:

$$\hat{\boldsymbol{\mu}} = \mathbb{E}_{p(\mathbf{h}|\mathbf{x})}[\mathbf{h}] = \sum_{\mathbf{h}} \mathbf{h}\, p(\mathbf{h}|\mathbf{x}) = \frac{1}{p(\mathbf{x})} \sum_{\mathbf{h}} q(\mathbf{h}|\mathbf{x})\, \mathbf{h}\, \frac{p(\mathbf{x},\mathbf{h})}{q(\mathbf{h}|\mathbf{x})} \approx \sum_{k=1}^{K} \tilde{w}^{(k)} \mathbf{h}^{(k)}. \tag{5.6}$$

As the initial estimator typically has high variance, AIS iteratively moves $\boldsymbol{\mu}_t$ toward $\hat{\boldsymbol{\mu}}$ by applying Eq. 5.6 until a stopping criteria is met. While using the update, $g(\boldsymbol{\mu}_t, \mathbf{x}, \gamma) =$

$\sum_{k=1}^{K} \tilde{w}^{(k)}\mathbf{h}^{(k)}$ in principle works, a convex combination of importance sample estimate of the current step and the parameters from the previous step tends to work more stably:

$$\mathbf{h}^{(m)} \sim \text{Bernoulli}(\boldsymbol{\mu}_k); \quad \boldsymbol{\mu}_{t+1} = g(\boldsymbol{\mu}_t, \mathbf{x}, \gamma) = (1-\gamma)\boldsymbol{\mu}_t + \gamma \sum_{k=1}^{K} \tilde{w}^{(k)}\mathbf{h}^{(k)}. \quad (5.7)$$

Here, γ is the inference rate and $(1-\gamma)$ can be thought of as the adaptive "damping" rate.

This approach, which we call adaptive importance refinement (AIR), should work with any discrete parametric distribution. Although AIR is applicable with continuous Gaussian variables, which model second-order statistics, we leave adapting AIR to continuous latent variables for future work.

5.2.2 Algorithm and Complexity

The general AIR algorithm follows Algorithm 2 with gradient variations following Eqs. 4.2, 5.2, 5.4, and 5.5. While iterative refinement may reduce the variance of stochastic gradient estimates and speed up learning, it comes at a computational cost, as each update is T times more expensive than fixed approximations. However, in addition to potential learning benefits, AIR can also improve the approximate posterior of an already trained directed belief networks at test, independent on how the model was trained.

5.3 Related Work

Adaptive importance refinement (AIR) trades computation for expressiveness and is similar in this regard to the refinement procedure of hybrid MCMC for variational inference (HVI, Salimans et al., 2015a) and normalizing flows for VAE (NF, Rezende & Mohamed, 2015). HVI has a similar complexity as AIR, as it requires re-estimating the lower-bound at every step. While NF can be less expensive than AIR, both HVI and NF rely on the

Algorithm 2 AIR

Require: A generative model $p(\mathbf{x}, \mathbf{h}; \boldsymbol{\phi}) = p(\mathbf{x}|\mathbf{h}; \boldsymbol{\phi})p(\mathbf{h}; \boldsymbol{\phi})$ and a recognition network $\boldsymbol{\mu}_0 = f(\mathbf{x}; \boldsymbol{\psi})$

Require: A transition operator $g(\mu, \mathbf{x}, \gamma)$ and inference rate γ.

Compute $\boldsymbol{\mu}_0 = f(\mathbf{x}; \boldsymbol{\psi})$ for $q_0(\mathbf{h}|\mathbf{x}; \boldsymbol{\psi})$

for t=1:T **do**

Draw K samples $\mathbf{h}^{(k)} \sim q_t(\mathbf{h}|\mathbf{x})$ and compute normalized importance weights $\tilde{w}^{(k)}$

$\boldsymbol{\mu}_t = (1 - \gamma)\boldsymbol{\mu}_{t-1} + \gamma \sum_{k=1}^{K} \tilde{w}^{(k)}\mathbf{h}^{(k)}$

end for

if reweight **then**

$\Delta\boldsymbol{\phi} \propto \sum_{k=1}^{K} \tilde{w}^{(k)} \nabla_{\boldsymbol{\phi}} \log p(\mathbf{x}, \mathbf{h}^{(k)}; \boldsymbol{\phi})$

else

$\Delta\boldsymbol{\phi} \propto \frac{1}{K} \sum_{k=1}^{K} \nabla_{\boldsymbol{\phi}} \log p(\mathbf{x}, \mathbf{h}^{(k)}; \boldsymbol{\phi})$

end if

if inclusive KL Divergence **then**

$\Delta\boldsymbol{\psi} \propto \sum_{k=1}^{K} \tilde{w}^{(k)} \nabla_{\boldsymbol{\psi}} \log q_0(\mathbf{h}^{(k)}|\mathbf{x}; \boldsymbol{\psi})$

else

$\Delta\boldsymbol{\psi} \propto \frac{1}{K} \sum_{k=1}^{K} \nabla_{\boldsymbol{\psi}} \log q_0(\mathbf{h}^{(k)}|\mathbf{x}; \boldsymbol{\psi})$

end if

VAE re-parameterization to work, and thus cannot be applied to discrete variables. Sequential importance sampling (Doucet et al., 2001) can offer a better refinement step than AIS but typically requires resampling to control variance. While parametric versions exist that could be applicable to training directed graphical models with discrete units (Gu, Ghahramani, and Turner, 2015a; Paige and Wood, 2016), their applicability as a general refinement procedure is limited as the refinement parameters need to be learned.

Importance sampling is central to reweighted wake-sleep (RWS, Bornschein & Bengio, 2014), importance-weighted autoencoders (IWAE, Burda et al., 2015), variational inference for Monte Carlo objectives (VIMCO, Mnih & Rezende, 2016), and recent work on

stochastic feed-forward networks (SFFN, Tang & Salakhutdinov, 2013; Raiko, Berglund, Alain, and Dinh, 2014). While each of these methods are competitive, they rely on importance samples from the recognition network and do not offer the low-variance estimates available from AIR. Neural variational inference and learning (NVIL, Mnih & Gregor, 2014) is a single-sample and biased version of VIMCO, which is greatly outperformed by techniques that use importance sampling. Both NVIL and VIMCO reduce the variance of the Monte Carlo gradients with an input-dependent baseline, but this approach does not necessarily provide a better posterior and cannot be used to give better estimates of the likelihood function or expectations.

5.4 Experiments and Results

5.4.1 Settings

We evaluate iterative refinement for variational inference (IRVI) using adaptive importance refinement (AIR) for both training and evaluating directed belief networks. We train and test on the following benchmarks: the binarized MNIST handwritten digit dataset Salakhutdinov and Murray (2008) and the Caltech-101 Silhouettes dataset. We centered the MNIST and Caltech datasets by subtracting the mean-image over the training set when used as input to the recognition network. We also train additional models using the reweighted wake-sleep algorithm (RWS, Bornschein & Bengio, 2014), the state of the art for many configurations of directed belief networks with discrete variables on these datasets for comparison and to demonstrate improving the approximate posteriors with refinement. With our experiments, we show that 1) iterative refinement can train a variety of directed models as well or better than existing methods, 2) the gains from refinement improves the approximate posterior, and can be applied to models trained by other algorithms, and 3) iterative refinement can be used to improve a model with a relatively simple approximate

posterior.

Models were trained using the RMSprop algorithm (Hinton, 2012) with a batch size of 100 and early stopping by recorded best variational lower bound on the validation dataset. For AIR, 20 "inference steps" ($K = 20$), 20 adaptive samples ($M = 20$), and an adaptive damping rate, γ, of 0.9 were used during inference, chosen from validation in initial experiments. 20 posterior samples ($N = 20$) were used for model parameter updates for both AIR and RWS. All models were trained for 500 epochs and were fine-tuned for an additional 500 epochs with a decaying learning rate and the SGD learning algorithm.

We use a generative model composed of a) a factorized Bernoulli prior as with sigmoid belief networks (SBNs) or b) an autoregressive prior, as in published MNIST results with deep autoregressive networks (DARN, Gregor et al., 2013):

$$\begin{aligned} &\text{a) } p(\mathbf{h}) = \prod_i p(h_i); P(h_i = 1) = \sigma(b_i), \\ &\text{b) } P(h_i = 1) = \sigma(\sum_{j=0}^{i-1}(W_r^{i,j<i} h_{j<i}) + b_i), \end{aligned} \tag{5.8}$$

where σ is the sigmoid ($\sigma(x) = 1/(1 + \exp(-x))$) function, W_r is a lower-triangular square matrix, and $\mathbf{b}$ is the bias vector.

For our experiments, we use conditional and approximate posterior densities that follow Bernoulli distributions:

$$P(h_{i,l} = 1|\mathbf{h}_{l+1}) = \sigma(W_l^{i,:} \cdot \mathbf{h}_{l+1} + b_{i,l}), \tag{5.9}$$

where W_l is a weight matrix between the l and $l + 1$ layers. As in Gregor et al. (2013) when training on MNIST, we do not use autoregression on the observations, $\mathbf{x}$, and use a fully factorized approximate posterior.

5.4.2 Variance Reduction and Choosing the AIR Objective

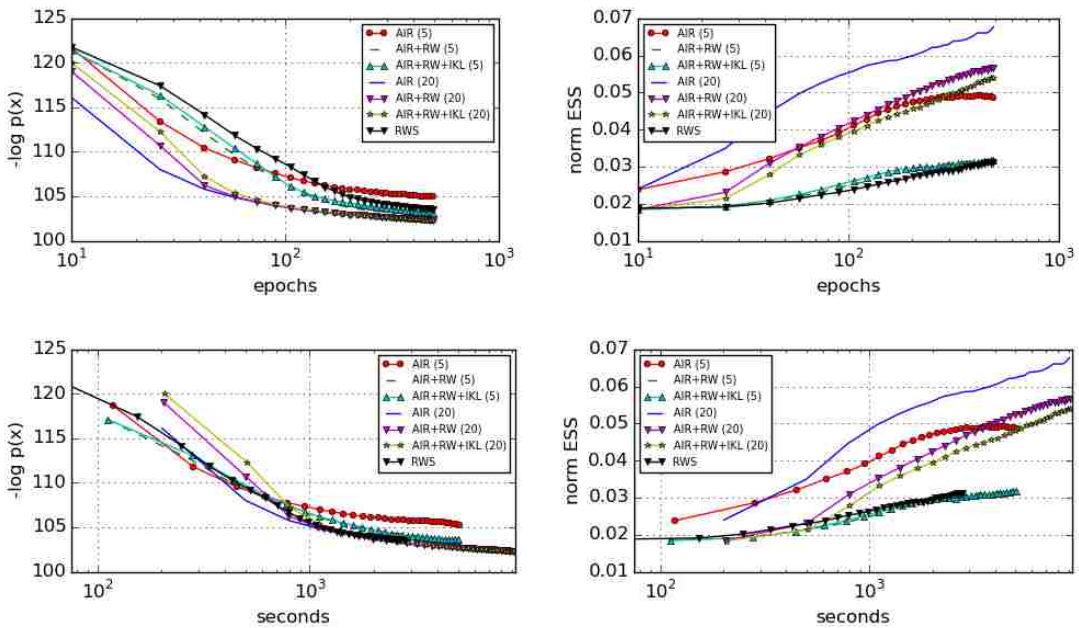

Figure 5.2: The log-likelihood (left) and normalized effective sample size (right) with epochs (top) and wall-clock times (bottom) in log-scale on the training set for AIR with 5 and 20 refinement steps (vanilla AIR), reweighted AIR with 5 and 20 refinement steps, reweighted AIR with inclusive KL objective and 5 or 20 refinement steps, and reweighted wake-sleep (RWS). All models were evaluated with 100 posterior samples, their respective number of refinement steps for the effective sample size (ESS), and with 20 refinement steps of AIR for the log-likelihood. Out of all the models, vanilla AIR had the highest ESS when trained with 20 refinement steps, and the ESS improves with the number of refinement steps. The large effective sample size for the AIR approximate posterior indirectly implies that the approximate posterior is close to the true posterior. Despite longer wall-clock time per epoch, AIR converges to lower log-likelihoods and effective sample size (ESS) than RWS. While reweighting can result in improved log-likelihood estimates, reweighting can reduce the ESS with models trained with AIR.

The effective sample size (ESS) in Eq. 5.3 is a good indicator of the variance of gradient estimate. In Fig. 5.2 (right), we observe that the ESS improves as we take more AIR steps when training a deep belief network (AIR(5) vs AIR(20)). When the approximate posterior is *not* refined (RWS), the ESS stays low throughout training, eventually resulting in a worse model. This improved ESS reveals itself as faster convergence in terms of the

exact log-likelihood in the right panel of Fig. 5.2 (see the progress of each curve until 100 epochs.)

This faster convergence does not guarantee a good final log-likelihood, as the latter depends on the tightness of the lower bound rather than the variance of its estimate. This is most apparent when comparing AIR(5), AIR+RW(5) and AIR+RW+IKL(5). AIR(5) has a low variance (high ESS) but computes the gradient of a looser lower bound from Eq. 4.2, while the other two compute the gradient of a tighter lower bound from Eq. 5.2. This results in AIR(5) converging faster than the other two, while the final log-likelihood estimates are better for the other two.

We however observe that the final log-likelihood estimates are comparable across all three variants (AIR, AIR+RW and AIR+RW+IKL) when a sufficient number of AIR steps are taken so that $\mathcal{L}_1$ is sufficiently tight. When 20 steps were taken, we observe that the AIR(20) converges faster as well as achieves a better log-likelihood compared to AIR+RW(20) and AIR+RW+IKL(20). Based on these observations, we use vanilla AIR (subsequently just "AIR") in our following experiments.

5.4.3 Training and Density Estimation

We evaluate AIR for training SBNs with one, two, and three layers of 200 hidden units and DARN with 200 and 500 hidden units, comparing against our implementation of RWS. All models were tested using $100,000$ posterior samples to estimate the lower bounds and average test log-probabilities to be consistent with the literature (Gregor et al., 2013; Bornschein & Bengio, 2014).

When training SBNs with AIR and RWS, we used a completely deterministic network for the approximate posterior. For example, for a 2-layer SBN, the approximate posterior factors into the approximate posteriors for the top and the bottom hidden layers, and the initial parameters for the top layer, $\boldsymbol{\mu}_0^{(2)}$ are a function of the initial parameters for the first

Table 5.1: Results for adaptive importance sampling iterative refinement (AIR), reweighted wake-sleep (RWS), and RWS with refinement with AIR at test (RWS+) for a variety of model configurations. Additional sigmoid belief networks (SBNs) trained with neural variational inference and learning (NVIL) from †Mnih & Gregor (2014) and variational inference for Monte Carlo objectives (VIMCO) from §Mnih & Rezende (2016). AIR is trained with 20 inference steps and adaptive samples ($K = 20, M = 20$) in training (*3 layer SBN was trained with 50 steps with a inference rate of 0.05). NVIL DARN results are from fDARN and VIMCO was trained using 50 posterior samples (as opposed to 20 with AIR and RWS).

Model	**MNIST**					**Caltech-101 Silhouettes**		
	RWS	RWS+	AIR	NVIL†	VIMCO§	RWS	RWS+	AIR
SBN 200	102.51	102.00	**100.92**	113.1	–	121.38	118.63	**116.61**
SBN 200×2	93.82	**92.83**	**92.90**	99.8	–	112.86	107.20	**106.94**
SBN 200×3	92.00	**91.02**	92.56*	96.7	**90.9**§	110.57	104.54	**104.36**
DARN 200	86.91	86.21	**85.89**	92.5†	–	113.69	**109.73**	**109.76**
DARN 500	85.40	**84.71**	85.46	90.7†	–	–	–	–

layer, $\boldsymbol{\mu}_0^{(1)}$ [1]:

$$q_0(\mathbf{h}_1, \mathbf{h}_2|\mathbf{x}; \boldsymbol{\mu}_0^{(1)}, \boldsymbol{\mu}_0^{(2)}) = q_0(\mathbf{h}_1|\mathbf{x}; \boldsymbol{\mu}_0^{(1)}) q(\mathbf{h}_2|\mathbf{x}; \boldsymbol{\mu}_0^{(2)});$$
$$\boldsymbol{\mu}_0^{(1)} = f_1(\mathbf{x}; \boldsymbol{\psi}_1); \quad \boldsymbol{\mu}_0^{(2)} = f_2(\boldsymbol{\mu}_0^{(1)}; \boldsymbol{\psi}_2). \tag{5.10}$$

For DARN, we trained two different configurations on MNIST: one with 500 stochastic units and an additional hyperbolic tangent deterministic layer with 500 units in both the generative and recognition networks, and another with 200 stochastic units with a 500 hyperbolic tangent deterministic layer in the generative network only. We only used the smaller 200 stochastic unit version of DARN when training on the Caltech-101 silhouettes dataset.

The results of our experiments with the MNIST and Caltech-101 silhouettes datasets trained with AIR, RWS, and RWS refined at test with AIR (RWS+) are in Table 5.1. Refinement at test (RWS+) always improves the results for RWS. As our unrefined results are comparable to those found in Bornschein & Bengio (2014), the improved results indicate

[1] To make the dependence on the initial *variational parameters*, $\boldsymbol{\mu}_0$, more explicit, we alter our usual parameterization of $q_0(\mathbf{h}|\mathbf{x}; \boldsymbol{\psi})$ here.

many evaluations of Helmholtz machines in the literature could benefit from refinement with AIR to improve evaluation accuracy.

For most model configurations, AIR and RWS perform comparably, though RWS appears to do better in the average test log-probability estimates for some configurations of MNIST. RWS+ performs comparably with variational inference for Monte Carlo objectives (VIMCO, Mnih & Rezende, 2016), despite the reported VIMCO results relying on more posterior samples in training. Finally, our AIR results approach SOTA with Caltech-101 silhouettes with 3-layer SBNs against neural autoregressive distribution estimator (NADE, Bornschein & Bengio, 2014).

We also tested our log-probability estimates against the exact log-probability (by marginalizing over the joint) of smaller single-layer SBNs with 20 stochastic units. The exact log-probability was -127.474 and our estimate with the unrefined approximate was -127.51 and -127.48 with 100 refinement steps. Overall, this result is consistent with those of Table 5.1, that iterative refinement improves the accuracy of log-probability estimates.

5.4.4 Posterior Improvement

In order to visualize the improvements due to refinement and to demonstrate AIR as a general means of improvement for directed models at test, we generate N samples from the approximate posterior without ($\mathbf{h} \sim q_0(\mathbf{h}|\mathbf{x}; \boldsymbol{\psi})$) and with refinement ($\mathbf{h} \sim q_T(\mathbf{h}|\mathbf{x})$), from a single-layer SBN with 20 stochastic units originally trained with RWS. We then use the samples from the approximate posterior to compute the expected conditional probability or average reconstruction: $\frac{1}{N}\sum_{n=1}^{N} p(\mathbf{x}|\mathbf{h}^{(n)})$. We used a restricted model with a lower number of stochastic units to demonstrate that refinement also works well with simple models, where the recognition network is more likely to "average" over latent configurations, giving a misleading evaluation of the model's generative capability.

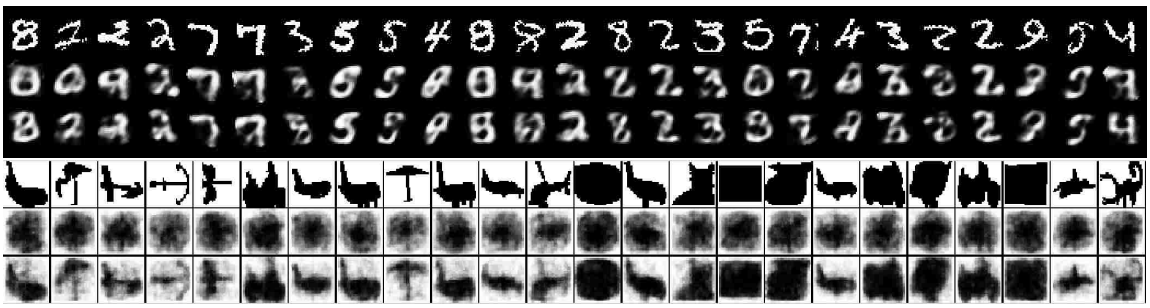

Figure 5.3: Top: Average reconstructions, $1/N \sum_{n=1}^{N} p(\mathbf{x}|\mathbf{h}^{(n)})$, for $\mathbf{h}^{(n)}$ sampled from the output of the recognition network, $q_0(\mathbf{h}|\mathbf{x})$ (middle row) against those sampled from the refined posterior, $q_T(\mathbf{h}|\mathbf{x})$ (bottom row) for $T = 20$ with a model trained on MNIST. Top row is ground truth. Among the digits whose reconstruction changes the most, many changes correctly reveal the identity of the digit. Bottom: Average reconstructions for a single-layer model with 200 trained on Caltech-101 silhouettes. Instead of using the posterior from the recognition network, we derived a simpler version, setting 80% of the variational parameters from the recognition network to 0.5, then applied iterative refinement. Despite starting at a relatively poor of the posterior, refinement is able to retrieve much structure of the ground truth.

We also refine the approximate posterior of a simplified version of the recognition network of a single-layer SBN with 200 units trained with RWS. We simplified the approximate posterior by first computing $\boldsymbol{\mu}_0 = f(\mathbf{x}; \boldsymbol{\psi})$, then randomly setting 80% of the variational parameters to 0.5.

Fig. 5.3 shows improvement from refinement for 25 digits from the MNIST test dataset, where the samples chosen were those of which the expected reconstruction error of the original test sample was the most improved. The digits generated from the refined posterior are of higher quality, and in many cases the correct digit class is revealed. This shows that, in many cases where the recognition network indicates that the generative model cannot model a test sample correctly, refinement can more accurately reveal the model's capacity. With the simplified approximate posterior, refinement is able to retrieve most of the shape of images from the Caltech-101 silhouettes, despite only starting with 20% of the original parameters from the recognition network. This indicates that the work of inference need not all be done via a complex recognition network: iterative refinement can be used to aid in inference with a relatively simple approximate posterior.

5.4.5 Continuous Latent Variables

With variational autoencoders (VAE), the back-propagated gradient of the lower bound with respect to the approximate posterior is composed of individual gradients for each factor, μ_i that can be applied simultaneously. Applying the gradient directly to the variational parameters, $\boldsymbol{\mu}$, without back-propagating to the recognition network parameters, $\boldsymbol{\psi}$, yields a simple iterative refinement operator:

$$\boldsymbol{\mu}_{t+1} = g(\boldsymbol{\mu}_t, \mathbf{x}, \gamma) = \boldsymbol{\mu}_t + \gamma \nabla_{\boldsymbol{\mu}} \mathcal{L}_1(\boldsymbol{\mu}, \mathbf{x}, \boldsymbol{\epsilon}), \tag{5.11}$$

where γ is the inference rate hyperparameter and $\boldsymbol{\epsilon}$ is auxiliary noise used in the re-parameterization.

Table 5.2: Lower bounds and NLL for various continuous latent variable models and training algorithms along with the corresponding VAE estimates. We use 200 latent Gaussian variables. †From Salimans et al. (2015a). §From Rezende & Mohamed (2015). ‡From Burda et al. (2015).

Model	$\leq$ -log p(x)	$\approx$ -log p(x)
VAE	94.48	89.31
VAE (w/ refinement)	90.57	88.53
$\text{GDIR}_{50,20}$	90.60	88.54
VAE†	94.18	88.95
HVI_1†	91.70	88.08
HVI_8†	88.30	85.51
VAE §	89.9	
DLGM+NF_{80} §	**85.1**	
VAE‡		86.35
IWAE ($K = 50$)‡		**84.78**

This gradient-descent iterative refinement (GDIR) is very straightforward with continuous latent variables as with VAE. However, GDIR with discrete units suffers the same shortcomings as when passing the gradients directly, so a better transition operator is needed (AIR). In the limit of $T = 0$, we do not arrive at VAE, as the gradients are never

passed through the approximate posterior during learning. However, as the complete computational graph involves a series of differentiable variables, $\boldsymbol{\mu}_t$, in addition to auxiliary noise, it is possible to pass gradients through GDIR to the recognition network parameters, $\boldsymbol{\psi}$, during learning, though we do not here.

For continuous latent variables, we used the same network structure as in (Kingma & Welling, 2013; Salimans et al., 2015a). Results for GDIR are presented in Table 5.2 for the MNIST dataset, and included for comparison are methods for learning non-factorial latent distributions for Gaussian variables and the corresponding result for VAE, the baseline. Though GDIR can improve the posterior in VAE, our results show that VAE is at an upper-bound for learning with a factorized posterior on the MNIST dataset. Further improvements on this dataset must be made by using a non-factorized posterior (re-weighting or sequential Monte Carlo with importance weighting). GDIR may still also provide improvement for training models with other datasets, and we leave this for future work.

5.4.6 Refinement of the Lowerbound and Effective Sample Size

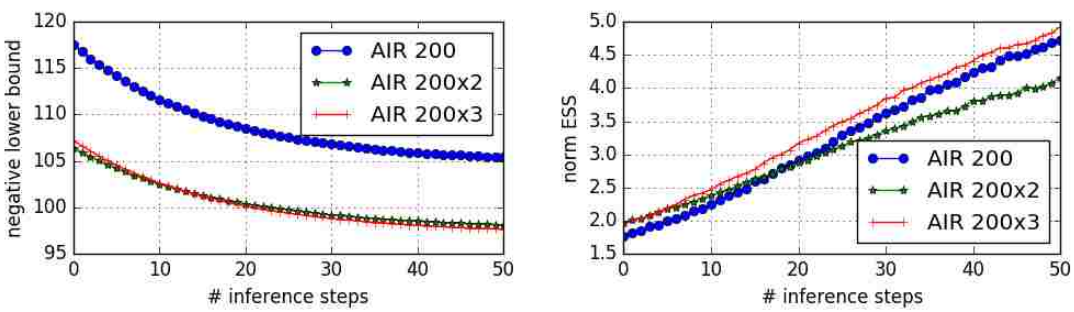

Figure 5.4: The variational lowerbound (left) and normalized effective sample size (ESS, right) the test set as the posterior is refined from the initial posterior provided by the recognition network. Models were trained with AIR with 20 refinement steps and one (AIR 200), two (AIR 200x2), and three (AIR 200x3) hidden layers. Refinement shows clear improves of both the variational lowerbound and effective sample size.

Iterative refinement via adaptive inference refinement (AIR) improves the variational lower bound and effective sample size (ESS) of the approximate posterior. To show this, we trained models with one, two, and three hidden layers with 200 binary units trained using AIR with 20 inference steps on the MNIST dataset for 500 epochs. Taking the initial approximate posterior from each model, we refined the posterior up to 50 steps (Figure 5.4), evaluating the lowerbound and ESS using 100 posterior samples. Refinement improves the posterior from models trained on AIR well beyond the number of steps used in training.

5.4.7 Updates and Wall-Clock Times

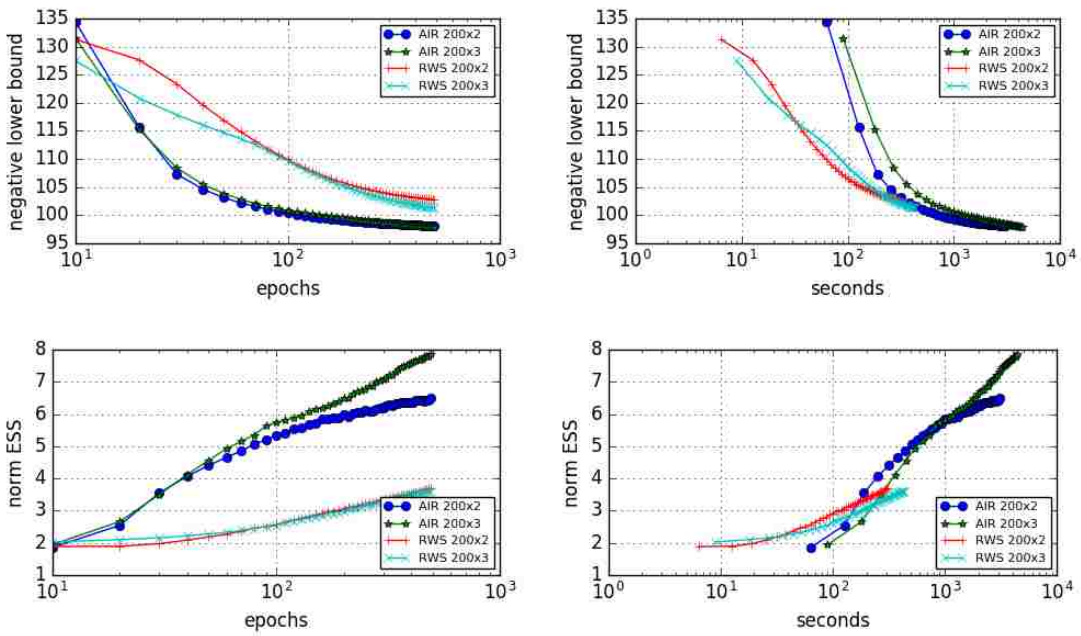

Figure 5.5: Negative lowerbound and effective samples size (ESS) across updates (epochs) and wall-clock time (seconds) for two and three layer sigmoid belief networks trained with adaptive iterative refinement (AIR) and reweighted wake-sleep (RWS). AIR was trained with 20 refinement steps with a damping rate of $\gamma = 0.9$. Each model was trained for 500 epochs and evaluated on the training dataset using 100 posterior samples. AIR takes less updates to reach equivalent variational lowerbound and ESS than RWS. While RWS can reach a higher lowerbound at earlier wall-clock times, AIR and RWS appear to converge to the same value, and AIR reaches much higher ESS.

Adaptive iterative refinement (AIR) and reweighted wake-sleep (RWS, Bornschein & Bengio, 2014) have competing convergence wall-clock times, while AIR outperforms on updates (Figures 5.2 and 5.5). AIR converges to a higher lowerbound and with far fewer updates than RWS, though RWS converges sooner to a similar value as AIR does later in training time. AIR outperforms RWS in ESS in both wall-clock time and updates. For a more accurate comparison, RWS may need to be trained at wall-clock times equal to that afforded to AIR. However, these results support the conclusion that AIR converges to similar values as RWS in less updates but similar wall-clock times.

5.4.8 Bidirectional Helmholtz Machines and AIR

As an alternative to the variational lowerbound, a lowerbound can be formulated from the geometric mean of the joint generative and approximate posterior models:

$$p^\star(\mathbf{x}, \mathbf{h}) = \frac{1}{Z}\sqrt{p(\mathbf{x}, \mathbf{h})q(\mathbf{x}, \mathbf{h})}. \tag{5.12}$$

In this procedure, known as a bidirectional Helmholtz machine (Bornschein, Shabanian, Fischer, and Bengio, 2015), the lowerbound, which minimizes the Bhattacharyya distance ($D_B(p, q) = -\log \sum_y \sqrt{p(y)q(y)}$), yields estimates of the likelihood, $p^\star(\mathbf{x})$, with importance weights,

$$w^{(k)} = \sqrt{\frac{p(\mathbf{x}, \mathbf{h}^{(k)})}{q(\mathbf{h}^{(k)}|\mathbf{x})}}. \tag{5.13}$$

Similar to with the variational lowerbound, we can refine the approximate posterior to maximize this lowerbound by simply replacing the weights in Equation 5.13.

We performed similar experiments to those in the experiments on wall-clock times above, using only a three layer SBN trained for 500 epochs with the equivalent AIR and BiHM procedures using the bidirectional lowerbound importance weights. We evaluated

these models using 10000 posterior samples on the test dataset and evaluated BiHM with (BiHM+) and without refinement.

Our results show similar negative log likelihoods for AIR (92.40 nats), BiHM (93.30 nats), and BiHM+ (92.90 nats), though AIR slightly outperforms BiHM+, and BiHM+ slightly outperforms BiHM. Further optimization is necessary for a better comparison to our experiments with the variational lowerbound. However these observations are consistent with those from our original experiments: AIR can be used to improve the posterior both in training and when evaluating models, regardless of how they were trained. Furthermore, AIR is compatible with optimizations based on alternative lowerbounds, broadening the scope in which AIR is applicable.

5.5 Iterative Refinement for Generative Adversarial Networks

The generative models used in this work so far performed inference that finds/learns the best latent configuration for given datapoint. The fitness of latent configurations is the likelihood they generate real data, and a likelihood function needs to be defined to train model parameters. Defining a likelihood function can pose a challenge in some settings as well as add constraints that may limit model capacity. Alternatively, adversarial frameworks provide a means of training generative models without directly maximizing the likelihood function or inferring latent structure. Rather, an second "adversary" model, usually trained to simply discriminate between real data and samples from the model, can be used to efficiently train the generative model.

5.5.1 Generative Adversarial Networks

Generative adversarial networks (GANs, Goodfellow et al., 2014) are a unique learning framework that use two separate models with opposing, competing, or *adversarial* objectives. Namely, a generator is pitted against a discriminatory adversary, whose objective is to discriminate between data and samples from the generator by maximizing:

$$\frac{1}{M}\left(\log D(\mathbf{x}^{(m)}) + \log(1 - D(G(\mathbf{z}^{(m)})))\right), \tag{5.14}$$

where $D(.)$ is a discriminator function with output in $[0, 1]$, $\mathbf{z}^{(m)}$ is a sample drawn from isotropic noise,[2] and $G(.)$ is a function that maps from the space of $\mathbf{z}$ to the space of data, $\mathbf{x}$. The two models play a minimax game with value function:

$$\min_G \max_D V(G, D) = \mathbb{E}[\log D(\mathbf{x})]_{\mathbf{x}\sim p(\mathbf{x})} + \mathbb{E}[\log(1 - D(G(\mathbf{z})))]_{\mathbf{z}\sim p(\mathbf{z})}. \tag{5.15}$$

The generator, rather than being trained to minimize the above value function, can be trained to maximize $\log(D(G(\mathbf{z})))$, which can alleviate some learning issues related to the discriminator loss saturating early in learning. With basic GANs, both generator and discriminator are feed forward networks, $D(.; \boldsymbol{\phi})$ and $G(.; \boldsymbol{\psi})$. Unlike directed graphical models such as those I covered in earlier sections, samples from the generator are not drawn from a distribution with parameters determined by the generator network, but the are the raw output from $G(.; \boldsymbol{\psi})$, so that the generative objective involves one continuous function with parameters from both models: $D(G(.; \boldsymbol{\psi}); \boldsymbol{\phi})$. This allows for training a generative model with a relatively simple fitness function and backpropagation, rather than a complex likelihood function.

In order for a generating function, $G(.; \boldsymbol{\psi})$, with real-valued parameters, $\boldsymbol{\psi}$, to generate discrete-valued outputs, $\mathbf{x}$, $G(.; \boldsymbol{\psi})$ cannot be fully continuous. A natural choice for the generator would be a continuous function, $\mathbf{y} = f(\mathbf{z})$ (e.g., a feed forward network),

[2] We have overloaded the sample index, m, here across the generator and the data.

followed by a step function at 0.5 so that:

$$x_i = \phi(y_i) = \begin{cases} 1, & \text{if } y_i \geq 0.5 \\ 0, & \text{otherwise,} \end{cases} \tag{5.16}$$

where $\mathbf{x} = \{x_i\}$ and $\mathbf{y} = \{y_i\}$. However, as the gradients that would otherwise be used to train the generator do not naturally back-propagate through discontinuous functions, it is not possible to train the generator from the discriminator error signal.

5.5.2 Adaptive Refinement Adversarial Networks

In order to address GANs with binary variables, first consider the optimum discriminator function, $D(\mathbf{x})^\star$, for the minimax game given a generator $G(.)$:

$$D^\star_G(\mathbf{x}) = \frac{p(\mathbf{x})}{p(\mathbf{x}) + q(\mathbf{x})}, \tag{5.17}$$

where $p(\mathbf{x})$ is the true distribution of the data, and $q(\mathbf{x})$ is the distribution as generated by $p(\mathbf{z})$ and $G(\mathbf{z})$. Given this optimal discriminator, the true density of the data can then be written as:

$$p(\mathbf{x}) = q(\mathbf{x}) \frac{D^\star_G(\mathbf{x})}{1 - D^\star_G(\mathbf{x})}. \tag{5.18}$$

One can then posit the following estimator for the true distribution, given an imperfect discriminator, $D(\mathbf{x})$:

$$\tilde{p}(\mathbf{x}) = \frac{1}{Z} q(\mathbf{x}) \frac{D(\mathbf{x})}{1 - D(\mathbf{x})}, \tag{5.19}$$

where the marginal term, $Z = \sum_{\mathbf{x}} q(\mathbf{x}) \frac{D(\mathbf{x})}{1-D(\mathbf{x})}$, guarantees that $\tilde{p}(\mathbf{x})$ is a proper probability distribution. Note that the global optimum occurs at $D(\mathbf{x}) = D^\star(\mathbf{x}) = 1/2$, so that $Z = 1$ and $\tilde{p}(\mathbf{x}) = q(\mathbf{x}) = p(\mathbf{x})$. $\tilde{p}(\mathbf{x})$ is a potentially biased estimator for the true density. However, the bias is implicit only in the quality of $D(\mathbf{x})$: the closer $D(\mathbf{x})$ is to $D^\star(\mathbf{x})$, the more guarantee that $\tilde{p}(\mathbf{x})$ is an unbiased estimator.

Inspired by the success of IRVI, we wish to train the generator to be closer to $\tilde{p}(\mathbf{x})$ via iteratively refinement. In order to accomplish this, we treat $q(\mathbf{x})$ as a the marginalization of a joint density, $q(\mathbf{x}) = \sum_{\mathbf{z}} g(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, rephrasing the generator function, $G(\mathbf{z})$ as a conditional distribution with Bernoulli centers as output. The reasoning here is that AIR relies on moving a proposal distribution toward the nearest mode, using an estimate of the mean in the neighborhood of said proposal distribution. In the case of Helmholtz machines in the previous section, the proposal distribution was the approximate posterior, while the target of refinement was the estimated mean of the latent variables over the true posterior. As the true posterior is multimodal, the effect of refinement should, on average across trials, be able locate each mode. In the case of GANs, refinement on the full distribution, $q(\mathbf{x})$, would require inference on the latent variables, which potentially reintroduces challenges in training similar to those with directed graphical models. However, if the generator outputs given the latent noise variables, $\mathbf{z}$, are the parameters for a unimodal distribution, $g(\mathbf{x}|\mathbf{z})$, then refinement can be used to simply move the parameters of said distribution toward the nearest mode of the data.

The procedure proposed here is to move the the conditional, $g(\mathbf{x}|\mathbf{z})$, toward the nearest mode as defined by the *unimodal* distribution:

$$\tilde{p}_{\mathbf{z}}(\mathbf{x}) = \frac{1}{Z_{\mathbf{z}}} g(\mathbf{x}|\mathbf{z}) \frac{D(\mathbf{x})}{1 - D(\mathbf{x})}, \tag{5.20}$$

where $\tilde{p}_{\mathbf{z}}$ is an estimate of the data in the neighborhood of the mode defined by $\mathbf{z}$, and $Z_{\mathbf{z}} = \sum_{\mathbf{x}} g(\mathbf{x}|\mathbf{z}) \frac{D(\mathbf{x})}{1-D(\mathbf{x})}$ is the marginal that ensures $\tilde{p}_{\mathbf{z}}$ is a proper probability distribution. Similar to with AIR in the previous section, I first calculate the mean over the proposal distribution defined by the conditional, $g(\mathbf{x}|\mathbf{z})$:

$$\mu_{\mathbf{z}} = \mathbb{E}[\mathbf{x}]_{\tilde{p}_{\mathbf{z}}(\mathbf{x})} = \sum_{\mathbf{x}} \mathbf{x}\, \tilde{p}_{\mathbf{z}}(\mathbf{x}) = \frac{1}{Z_{\mathbf{z}}} \sum_{\mathbf{x}} \mathbf{x}\, g(\mathbf{x}|\mathbf{z}_k) \frac{D(\mathbf{x})}{1 - D(\mathbf{x})}. \tag{5.21}$$

The mean can then be estimated using samples from the conditional:

$$\mu_{\mathbf{z}} \approx \frac{1}{\sum_{m'=1}^{M} \frac{D(\mathbf{x}^{(m')})}{1-D(\mathbf{x}^{(m')})}} \sum_{m=1}^{M} \mathbf{x}^{(m)} \frac{D(\mathbf{x}^{(m)})}{1 - D(\mathbf{x}^{(m)})} = \sum_{m=1}^{M} \mathbf{x}^{(m)}\, \tilde{w}^{(m)}, \tag{5.22}$$

where the MC estimate is performed by sampling over the conditional, $g(\mathbf{x}|\mathbf{z})$, and $\tilde{w}^{(m)}$ are the normalized weights, as with AIR. As with Helmholtz machines, $\mu_{\mathbf{z}}$ is reached iteratively via the procedure from Equation 5.7, which should reduce the variance of the estimate of $\mu_{\mathbf{z}}$ and gradients for training the generator.

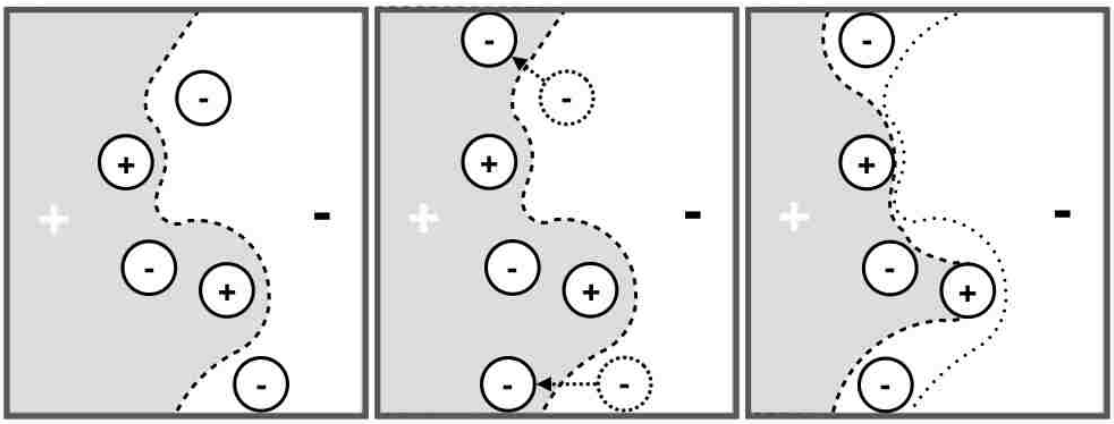

Figure 5.6: A visualization of the effect on the decision boundary defined by $D(\mathbf{x})$ with AIRGAN. First, data samples are drawn from the empirical distribution $(+)$ and the generator, $G(\mathbf{x}|\mathbf{z})$, $(-)$. These lie of different sides of the decision boundary. The refinement procedure then moves its samples over the boundary by iteratively refining the underlying mean of $G(\mathbf{x}|\mathbf{z})$. The discriminator then moves the decision boundary to effectively place the generated samples back across the decision boundary. Samples that cannot be placed back across the decision boundary are indistinguishable from real data.

An illustration of the iterative minimax procedure for adaptive iterative refinement for generative adversarial networks (AIRGAN) is provided in Figure 5.6. AIRGAN effectively uses the discriminator as a metric of the true density in the local neighborhood of samples from $g(\mathbf{x}|\mathbf{z})$. The global optima for the value function in Equation 5.15 occurs when $D^{\star}(\mathbf{x}) = 1/2$, or in other words with both real data and samples lie on the decision boundary. The refinement procedure moves the means of the generator in the neighborhood of a mode across the boundary until they are on the "positive side", corresponding to areas the discriminator reserves for real data. This increases the discriminator loss from Equation 5.14, so the discriminator tries to move the decision boundary so that the samples

from the refinement procedure are on the negative side, but with the limitation that data samples are still on the positive side. The "surviving" samples from the generator should, as defined by the capacity of the discriminator, be indistinguishable from real data.

This procedure is similar to the positive / negative phases in contrastive divergence (CD, Hinton, 2002). The data samples provide positive pressure at the decision boundary, shaping it so that all data lies on the same side. The generator samples provide negative pressure to the decision boundary, removing space on the positive side that does not correspond to data. As the generator distribution becomes arbitrarily close to the true distribution, the positive and negative samples begin to occupy the same space, which at equilibrium is at the decision boundary according to the global optimum.

Figure 5.7: Samples from the generative function, $q(\mathbf{x}) = \sum_{\mathbf{z}} g(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ trained with AIRGAN. Most samples are highly variable and data-like, with some rare noisy exceptions.

5.5.3 Results

I applied this approach on the MNIST dataset with a 2-layer discriminator with 200 and 100 softplus hidden units with a dropout rate of 0.1 and a generator with 200 and 500 softplus hidden units with isotropic Gaussian noise as input. For the inference procedure, I used an damping rate of $\gamma = 0.9$ with 20 inference steps, as this worked in our experiments with Helmholtz machines. Optimization was performed for 100 epochs using the RMSProp algorithm with a learning rate of 0.001.

Samples from the resulting model trained on AIRGAN are highly variable and data-like, with some rare exceptions that resemble in most cases to be noisy versions of digits (Figure 5.7). The true-positive rate taken over the test dataset for the discriminator was 0.55, with an average false-positive rate taken over equivalent number of samples from the generator was 0.44, indicating a good, but not optimal generative function.

Unfortunately, evaluating the likelihood of the model over the test dataset is difficult, as calculating the likelihood function requires a summation over all latent variables or a posterior. The former is intractable, though to latter may be possible via variational learning, GDIR, or adversarially-learned inference (Dumoulin, Belghazi, Poole, Lamb, Arjovsky, Mastropietro, and Courville, 2016) This, as well as further extensions (such as to continuously-valued data) and analysis we leave for future work.

Chapter 6

Conclusions

6.1 Summary of Work

I have presented our work on applying functional magnetic imaging (fMRI) data to restricted Boltzmann machines (RBMs), demonstrating comparable performance to the state-of-the art independent component analysis (ICA) in separating intrinsic networks, along with some key advantages in connectivity analysis and training. ICA is a purely linear model that requires dimension reduction to work, but benefits from a very simple learning algorithm that does not require approximations to train and has a straightforward interpretation. RBMs are defined by an energy function, potentially have higher capacity than ICA, and do not require dimensionality reduction. Despite requiring estimates for learning, RBMs appear to exhibit some advantages when applied to fMRI, indicating that fMRI analysis benefits from the more flexible representation. The lack of factorized decomposition makes the latent variables in RBMs difficult to interpret relative to ICA, though our work shows that the conditional distribution can be used to represent intrinsic networks that are competitive to or better than ICA.

I also presented an alternative to sampling from i.i.d. data or from the energy defined

by restricted Boltzmann machines (RBMs) using an auxiliary recurrent network (RNN). Despite being used primarily to model sequences, the results presented indicate that an RNN can be used to learn a transition operator such that the stationary distribution corresponds to independent and identically distributed data or is the marginal density of an RBM. This implies that RNNs can potentially be used in a variety of situations where sampling from or exploration of a distribution is necessary.

Next, I presented our work demonstrating that variational autoencoders (VAE) can be used to infer non-linear latent structure from magnetic resonance imaging (MRI) data using directed belief networks. Directed nets provide a means of inference and dimension reduction through an approximate posterior, though variance in variational inference perhaps makes learning difficult with MRI data (which contrasts with RBMs), as our results are somewhat unsatisfying. Like ICA and unlike RBMs, interpretation of latent structure is relatively straightforward, as the sources factorize, but unlike ICA, the relationship between sources and data can be highly nonlinear and can be composed of a very deep hierarchy of sources. This potentially provides a more powerful model than both ICA or RBMs, though more work is necessary to improve results.

In order to address dynamics, I presented a novel approach to learning temporal dynamics in intrinsic networks. This model successfully separating independent, task related, and group differentiating components from fMRI data with time- and subject-varying spatial maps. To our knowledge, this is a novel contribution to the study of intrinsic networks.

In the end, for static models of MRI data or fMRI data (without explicit representations of temporal dynamics), RBMs may be the best choice, as they provide equal or better features as ICA, do not require preprocessing, and provide better results that directed graphical models. Deep directed graphical models may not be appropriate, as the necessary variance from variational inference may pose challenges with high-dimensional, low sample size data, like MRI. However, if modeling dynamics are part of a research goal, RNN ICA appears to be a very promising approach for learning time course dynamics and

variable spatial maps.

Finally, we introduced a novel method for inferring latent structure, iterative refinement of the approximate posterior (IRVI), that works with directed belief networks with discrete or continuous latent variables. This was extended to generative adversarial networks (GANs), and was shown as a method for training these with discrete data.

6.2 Acknowledgements

I would like to thank my dissertation committee, Trilce Estrada, Vince Calhoun, Sergey Plis, Kyunghyun Cho, and Ruslan Salakhutdinov as well, as my various collaborators, Nebojsa Jojic, Junyoung Chung, and Elena Allen for their countless help and support, as well as for the patience toward my insufferable confusion and never-ending questions. I would also like to acknowledge the support from the Program for Interdisciplinary Biological and Biomedical Imaging and the Mind Research Network for their support, as well as Google, and Microsoft Research for given my research efforts a home.

References

Allen, Elena A, Erhardt, Erik B, Damaraju, Eswar, Gruner, William, Segall, Judith M, Silva, Rogers F, Havlicek, Martin, Rachakonda, Srinivas, Fries, Jill, Kalyanam, Ravi, et al. A baseline for the multivariate comparison of resting-state networks. *Frontiers in Systems Neuroscience*, 5(2), 2011.

Allen, Elena A, Erhardt, Erik B, Wei, Yonghua, Eichele, Tom, and Calhoun, Vince D. Capturing inter-subject variability with group independent component analysis of fMRI data: a simulation study. *Neuroimage*, 59(4):4141–4159, 2012.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Bakir, Gökhan. *Predicting structured data*. MIT press, 2007.

Beckmann, Christian F and Smith, Stephen M. Probabilistic independent component analysis for functional magnetic resonance imaging. *Medical Imaging, IEEE Transactions on*, 23(2):137–152, 2004.

Beckmann, Christian F, DeLuca, Marilena, Devlin, Joseph T, and Smith, Stephen M. Investigations into resting-state connectivity using independent component analysis. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1457): 1001–1013, 2005.

Bell, Anthony J. and Sejnowski, Terrence J. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

Bishop, Christopher M. Mixture density networks. 1994.

Biswal, Bharat, Zerrin Yetkin, F, Haughton, Victor M, and Hyde, James S. Functional connectivity in the motor cortex of resting human brain using echo-planar MRI. *Magnetic Resonance in Medicine*, 34(4):537–541, 1995.

Blondel, Vincent D, Guillaume, Jean-Loup, Lambiotte, Renaud, and Lefebvre, Etienne. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

Bornschein, Jörg and Bengio, Yoshua. Reweighted wake-sleep. *arXiv preprint arXiv:1406.2751*, 2014.

Bornschein, Jorg, Shabanian, Samira, Fischer, Asja, and Bengio, Yoshua. Bidirectional helmholtz machines. *arXiv preprint arXiv:1506.03877*, 2015.

Burda, Yuri, Grosse, Roger B, and Salakhutdinov, Ruslan. Accurate and conservative estimates of mrf log-likelihood using reverse annealing. *arXiv preprint arXiv:1412.8566*, 2014.

Burda, Yuri, Grosse, Roger, and Salakhutdinov, Ruslan. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Calhoun, Vince D., Adali, Tulay, Pearlson, Godfrey D., and Pekar, J. J. A method for making group inferences from functional MRI data using independent component analysis. *Human Brain Mapping*, 14, 2001.

Calhoun, Vince D., Kiehl, Kent A., and Pearlson, Godfrey D. Modulation of temporally coherent brain networks estimated using ICA at rest and during cognitive tasks. *Human Brain Mapping*, 29(7):828–838, 2008.

Castro, Eduardo, Hjelm, R Devon, Plis, Sergey, Dihn, Laurent, Turner, Jessica, and Calhoun, Vince. Deep independence network analysis of structural brain imaging: Application to schizophrenia. 2016.

Cho, KyungHyun, Raiko, Tapani, and Ilin, Alexander. Parallel tempering is efficient for learning restricted boltzmann machines. Citeseer.

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

Chung, Junyoung, Kastner, Kyle, Dinh, Laurent, Goel, Kratarth, Courville, Aaron C, and Bengio, Yoshua. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2962–2970, 2015.

Damoiseaux, JS, Rombouts, SARB, Barkhof, F, Scheltens, P, Stam, CJ, Smith, Stephen M, and Beckmann, CF. Consistent resting-state networks across healthy subjects. *Proceedings of the National Academy of Sciences*, 103(37):13848–13853, 2006.

Dayan, Peter, Hinton, Geoffrey E, Neal, Radford M, and Zemel, Richard S. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Desjardins, Guillaume, Courville, Aaron, Bengio, Yoshua, Vincent, Pascal, and Delalleau, Olivier. Parallel tempering for training of restricted boltzmann machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 145–152. MIT Press Cambridge, MA, 2010.

Doucet, Arnaud, De Freitas, Nando, and Gordon, Neil. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pp. 3–14. Springer, 2001.

Du, Wei, Calhoun, Vince D, Li, Hualiang, Ma, Sai, Eichele, Tom, Kiehl, Kent A, Pearlson, Godfrey D, and Adali, Tülay. High classification accuracy for schizophrenia with rest and task fmri data. *Frontiers in human neuroscience*, 6, 2012.

Dumoulin, Vincent, Belghazi, Ishmael, Poole, Ben, Lamb, Alex, Arjovsky, Martin, Mastropietro, Olivier, and Courville, Aaron. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

Erhardt, Erik, Allen, Elena A., Wei, Yonghua, and Eichele, Tom. SimTB, a simulation toolbox for fMRI data under a model of spatiotemporal separability. *Neuroimage*, 59 (4):4160–4167, 2012.

Ferguson, Thomas S. A bayesian analysis of some nonparametric problems. *The annals of statistics*, pp. 209–230, 1973.

Frey, Brendan J and Hinton, Geoffrey E. Variational learning in nonlinear gaussian belief networks. *Neural Computation*, 11(1):193–213, 1999.

Geman, Stuart and Graffigne, Christine. Markov random field image models and their applications to computer vision. In *Proceedings of the International Congress of Mathematicians*, volume 1, pp. 2, 1986.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

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

Graves, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649. IEEE, 2013.

Gregor, Karol, Danihelka, Ivo, Mnih, Andriy, Blundell, Charles, and Wierstra, Daan. Deep autoregressive networks. *arXiv preprint arXiv:1310.8499*, 2013.

Gu, Shixiang, Ghahramani, Zoubin, and Turner, Richard E. Neural adaptive sequential monte carlo. In *Advances in Neural Information Processing Systems*, pp. 2611–2619, 2015a.

Gu, Shixiang, Levine, Sergey, Sutskever, Ilya, and Mnih, Andriy. Muprop: Unbiased backpropagation for stochastic neural networks. *arXiv preprint arXiv:1511.05176*, 2015b.

Hinton, Geoffrey. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926, 2010.

Hinton, Geoffrey. Neural networks for machine learning. Coursera, video lectures, 2012.

Hinton, Geoffrey E. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Hinton, Geoffrey E, Welling, Max, Teh, Yee Whye, and Osindero, Simon. A new view of ica. In *Int. Conf. on Independent Component Analysis and Blind Source Separation*, 2001.

Hinton, Geoffrey E, Osindero, Simon, and Teh, Yee-Whye. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Hjelm, R Devon, Calhoun, Vince D, Salakhutdinov, Ruslan, Allen, Elena A, Adali, Tulay, and Plis, Sergey M. Restricted boltzmann machines for neuroimaging: an application in identifying intrinsic networks. *NeuroImage*, 96:245–260, 2014.

Hjelm, R Devon, Cho, Kyunghyun, Chung, Junyoung, Salakhutdinov, Russ, Calhoun, Vince, and Jojic, Nebojsa. Iterative refinement of approximate posterior for training directed belief networks. *Neuron Information Processing Systems, in press.*, 2016a.

Hjelm, R Devon, Plis, Sergey, and Calhoun, Vince. Recurrent neural networks for spatiotemporal dynamics of intrinsic networks from fmri data. *arXiv preprint arXiv:1611.00864*, 2016b.

Hjelm, R Devon, Plis, Sergey M, and Calhoun, Vince C. Variational autoencoders for feature detection of magnetic resonance imaging data. *arXiv preprint arXiv:1603.06624*, 2016c.

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

Hornik, Kurt, Stinchcombe, Maxwell, and White, Halbert. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Hoyer, Patrik O. Non-negative sparse coding. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pp. 557–565. IEEE, 2002.

Hyvärinen, Aapo and Oja, Erkki. Independent component analysis: algorithms and applications. *Neural networks*, 13(4):411–430, 2000.

Jordan, Michael Irwin. *Learning in graphical models*, volume 89. Springer Science & Business Media, 1998.

Kim, D, Burge, John, Lane, Terran, Pearlson, Godfrey D, Kiehl, Kent A, and Calhoun, Vincent D. Hybrid ica–bayesian network approach reveals distinct effective connectivity differences in schizophrenia. *Neuroimage*, 42(4):1560–1568, 2008.

Kim, Junghoe, Calhoun, Vince D, Shim, Eunsoo, and Lee, Jong-Hwan. Deep neural network with weight sparsity control and pre-training extracts hierarchical features and enhances classification performance: Evidence from whole-brain resting-state functional connectivity patterns of schizophrenia. *NeuroImage*, 124:127–146, 2016.

Kingma, Diederik and Welling, Max. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kiviniemi, Vesa, Starck, Tuomo, Remes, Jukka, Long, Xiangyu, Nikkinen, Juha, Haapea, Marianne, Veijola, Juha, Moilanen, Irma, Isohanni, Matti, Zang, Yu-Feng, et al. Functional segmentation of the brain cortex using high model order group PICA. *Human Brain Mapping*, 30(12):3865–3886, 2009.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

LeCun, Yann and Bengio, Yoshua. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10), 1995.

LeCun, Yann, Cortes, Corinna, and Burges, Christopher JC. The mnist database of handwritten digits, 1998.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Mnih, Andriy and Gregor, Karol. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1791–1799, 2014.

Mnih, Andriy and Rezende, Danilo J. Variational inference for monte carlo objectives. *arXiv preprint arXiv:1602.06725*, 2016.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

Neal, Radford M. Learning stochastic feedforward networks. *Department of Computer Science, University of Toronto*, 1990.

Neal, Radford M. Connectionist learning of belief networks. *Artificial intelligence*, 56(1): 71–113, 1992.

Neal, Radford M. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

Neal, Radford M et al. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2:113–162, 2011.

Oh, Man-Suk and Berger, James O. Adaptive importance sampling in monte carlo integration. *Journal of Statistical Computation and Simulation*, 41(3-4):143–168, 1992.

Oord, Aaron van den, Dieleman, Sander, Zen, Heiga, Simonyan, Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew, and Kavukcuoglu, Koray. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Paige, Brooks and Wood, Frank. Inference networks for sequential monte carlo in graphical models. *arXiv preprint arXiv:1602.06701*, 2016.

Plis, Sergey M, Hjelm, R Devon, Salakhutdinov, Ruslan, and Calhoun, Vince D. Deep learning for neuroimaging: a validation study. *arXiv preprint arXiv:1312.5847*, 2013.

Raiko, Tapani, Berglund, Mathias, Alain, Guillaume, and Dinh, Laurent. Techniques for learning binary stochastic feedforward neural networks. *arXiv preprint arXiv:1406.2989*, 2014.

Rezende, Danilo J, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1278–1286, 2014.

Rezende, Danilo Jimenez and Mohamed, Shakir. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Salakhutdinov, Ruslan and Hinton, Geoffrey E. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pp. 448–455, 2009.

Salakhutdinov, Ruslan and Larochelle, Hugo. Efficient learning of deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pp. 693–700, 2010.

Salakhutdinov, Ruslan and Murray, Iain. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pp. 872–879. ACM, 2008.

Salimans, Tim, Kingma, Diederik, and Welling, Max. Markov chain monte carlo and variational inference: Bridging the gap. In Blei, David and Bach, Francis (eds.), *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1218–1226. JMLR Workshop and Conference Proceedings, 2015a. URL `http://jmlr.org/proceedings/papers/v37/salimans15.pdf`.

Salimans, Tim, Kingma, Diederik P, Welling, Max, et al. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pp. 1218–1226, 2015b.

Schmidhuber, Jürgen. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

Sherman, Jack and Morrison, Winifred J. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21 (1):124–127, 1950.

Smith, Stephen M, Fox, Peter T, Miller, Karla L, Glahn, David C, Fox, P Mickle, Mackay, Clare E, Filippini, Nicola, Watkins, Kate E, Toro, Roberto, Laird, Angela R, et al. Correspondence of the brain's functional architecture during activation and rest. *Proceedings of the National Academy of Sciences*, 106(31):13040–13045, 2009.

Srivastava, Nitish, Hinton, Geoffrey E, Krizhevsky, Alex, Sutskever, Ilya, and Salakhut-

dinov, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Suk, Heung-Il, Lee, Seong-Whan, Shen, Dinggang, Initiative, Alzheimer's Disease Neuroimaging, et al. Hierarchical feature representation and multimodal fusion with deep learning for ad/mci diagnosis. *NeuroImage*, 101:569–582, 2014.

Sutskever, Ilya and Hinton, Geoffrey E. Deep, narrow sigmoid belief networks are universal approximators. *Neural Computation*, 20(11):2629–2636, 2008.

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

Swanson, Nathan, Eichele, Tom, Pearlson, Godfrey, Kiehl, Kent, Yu, Qingbao, and Calhoun, Vince D. Lateral differences in the default mode network in healthy controls and patients with schizophrenia. *Human Brain Mapping*, 32(4):654–664, 2011.

Tang, Yichuan and Salakhutdinov, Ruslan R. Learning stochastic feedforward neural networks. In *Advances in Neural Information Processing Systems*, pp. 530–538, 2013.

Thibodeau-Laufer, Eric, Alain, Guillaume, and Yosinski, Jason. Deep generative stochastic networks trainable by backprop. 2014.

Tieleman, Tijmen. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pp. 1064–1071. ACM, 2008.

Vincent, Pascal, Larochelle, Hugo, Lajoie, Isabelle, Bengio, Yoshua, and Manzagol, Pierre-Antoine. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

Ze, Heiga, Senior, Andrew, and Schuster, Mike. Statistical parametric speech synthesis using deep neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7962–7966. IEEE, 2013.

Zou, Hui, Hastie, Trevor, and Tibshirani, Robert. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006.

Zuo, Xi-Nian, Kelly, Clare, Adelstein, Jonathan S, Klein, Donald F, Castellanos, F Xavier, and Milham, Michael P. Reliable intrinsic connectivity networks: test–retest evaluation using ICA and dual regression approach. *Neuroimage*, 49(3):2163–2177, 2010.