Advances in deep generative modeling for clinical data

by

Rahul Gopalkrishnan

published as: Rahul G. Krishnan

BaSc., The University of Toronto (2013)

M.S., New York University (2016)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2020

Author . . . .
Department of Electrical Engineering and Computer Science
June 30, 2020

Certified by. . . . .
David A. Sontag
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee for Graduate Students

Advances in deep generative modeling for clinical data

by

Rahul Gopalkrishnan

Submitted to the Department of Electrical Engineering and Computer Science
on June 30, 2020, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in
Electrical Engineering and Computer Science

Abstract

The intelligent use of electronic health record data opens up new opportunities to improve clinical care. Such data have the potential to uncover new sub-types of a disease, approximate the effect of a drug on a patient, and create tools to find patients with similar phenotypic profiles. Motivated by such questions, this thesis develops new algorithms for unsupervised and semi-supervised learning of latent variable, deep generative models – Bayesian networks parameterized by neural networks.

To model static, high-dimensional data, we derive a new algorithm for inference in deep generative models. The algorithm, a hybrid between stochastic variational inference and amortized variational inference, improves the generalization of deep generative models on data with long-tailed distributions. We develop gradient-based approaches to interpret the parameters of deep generative models, and fine-tune such models using supervision to tackle problems that arise in few-shot learning.

To model longitudinal patient biomarkers as they vary due to treatment we propose Deep Markov Models (DMMs). We design structured inference networks for variational learning in DMMs; the inference network parameterizes a variational approximation which mimics the factorization of the true posterior distribution. We leverage insights in pharmacology to design neural architectures which improve the generalization of DMMs on clinical problems in the low-data regime. We show how to capture structure in longitudinal data using deep generative models in order to reduce the sample complexity of nonlinear classifiers thus giving us a powerful tool to build risk stratification models from complex data.

Thesis Supervisor: David A. Sontag
Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

To my graduate advisor, David Sontag, thank you for all the patience, wisdom and guidance that you has shown me for the better half of a decade. Your infinite enthusiasm for research and fearlessness in asking difficult questions, have and continue to inspire me. Interdisciplinary work is hard, but you've guided and helped me walk the tightrope that spans the daunting peaks of research questions that are technically challenging and those that can impact people's lives.

I want to thank all the members of my thesis committee for their feedback on my work and on this thesis. To Pete Szolovitz, I've enjoyed our conversations whose topics span the gamut from locations for your sabbatical to how you came to study computer science in medicine. To Matthew Hoffman, thank for showing me the breadth and width of how algorithms for probabilistic inference inference can be applied to real problems; I have learned much from your unerring eye to spot patterns that others often miss. To Uri Shalit, thank you for being an excellent office mate, and for every manner of professional and personal life advice you've given me over the years.

I am grateful to the many mentors, collaborators, and colleagues without whom many of my research ideas would not have reached fruition. Thank you to Lydia Bourouiba and Simone Cenci for introducing me to the world of epidemiological modelling; I hope to continue to explore the many relationships between statistical learning and fluid mechanics. Thank you to Simone Lacoste Julien, Dawen Liang, Rajesh Ranganth, Li-wei Lehman, Andrew Yee, Narges Razavian, Hendrik Strobelt, Nicolo Fusi and Lester Mackey for sharing your knowledge with me. Thank you to all the members of the ClinicalML lab: Yoni Halpern, Yacine Jernite, Rachel Hodos, Fredrik Johansson, Irene Chen, Michael Oberst, Monica Agrawal, Zeshan Hussain, Sanjat Kanjilal, Arjun Khandelwal and Christina Ji: your wit, intelligence and humor have made science fun. I look forward to many more collaborations with all of you.

Life doesn't press pause for a PhD, as my parents keep reminding me, and I want to thank all my friends, both in and out of school, for keeping me sane through all the ups and downs of being a PhD student. To Rachel Gubow, Siddharth Krishna, Shravas Rao, Anthony Rossi, Alex Grote and the many other graduate students at NYU, thank you for being part of many adventures in NY. Thank you to Karthik Narasimhan, Ardavan Saeedi, Zoya Bylinksii and Marzyeh Ghassemi for giving me a warm welcome when I first moved to Cambridge. To Isabel Schwarz, Jonathan Ng, and Carmen Reilly, thank you for for being excellent sounding boards for talking through any manner of road bumps in life. To Kate Finegold (and the inimitable Mocha), both of your smiles make long weeks fly by. To Firas Kamaleddine, Ben Aneesh, Anirudh Ganti, Omer Shaeldin, Aniruddha Borah, Ratika Goyal, Inmar Givoni, Mustafa El-hilo and Aniruddha Borah, Zahan Malkani, Ameya Shroff and Arka Bhattacharyya – after more than a decade since we parted ways, I'm grateful that you all remain a close part of my life.

Finally, to my family who have been a bedrock of support throughout my academic journey. To my father, Vettithuruthil Gopalkrishnan, thank you for teaching me to never stop asking questions, and for teaching me how to weather the harshest storms in life with a smile. To my mother, Beena Krishnan Nair, thank you for your steadfast belief in me, even when my belief in myself faded. And to my sister, Rasmi Gopalkrishnan, who taught me that living is believing in principles that you're willing to make sacrifices for. You fill my life with warmth and love. This is for you.

When I began my PhD I sought a framework to make sense of the myriad problems that people studied in machine learning, which comprises an abundance of ideas that span topics in statistics, deep learning, stochastic optimization, linear algebra and causality. Like others before me, in Bayesian networks, I found the technical machinery to begin to organize all these concepts within a single unifying framework. Doing so has helped me read, organize and contextualize ideas from a variety of different fields. If there is anything that I have learned over the past several years, it is that progress across multiple fields will be driven by researchers across the world speaking and understanding the same technical language. To that end, discovering, experimenting and contributing to such a unifying framework has been a liberating experience.

Contents

1 Introduction **29**

1.1 Machine learning for healthcare 29

1.2 Challenges in healthcare 31

1.3 Contributions 32

2 Background **35**

2.1 Random variables and probabilities 35

2.2 Graphical models 37

2.2.1 Structure as domain knowledge 38

2.2.2 Independence statements 39

2.3 Bayesian networks 40

2.3.1 Parameterizations of Bayesian networks 40

2.3.2 Learning 45

2.3.3 Variational learning of latent variable models 46

2.4 Learning with automatic differentiation 52

2.5 Modeling data with deep generative models 53

3 Gradient based introspection in deep generative models **55**

3.1 Introduction 55

3.2 Jacobian vectors 58

3.3 Evaluation 61

3.3.1 Text data 62

3.3.2 Electronic Health Record (EHR) data 65

3.3.3 Netflix: Embeddings for movies 68

3.4 Discussion 69

4 Representation learning for high-dimensional data 73

4.1 Introduction 73

4.2 Setup 75

4.3 Sources of error in variational learning 77

4.3.1 Limitations of joint parameter updates 79

4.4 Improving estimates of variational parameters 80

4.4.1 Between stochastic and amortized variational inference 80

4.4.2 Representations for inference networks 80

4.4.3 Spectral analysis of the Jacobian matrix 81

4.5 Related 2ork 82

4.6 Evaluation 84

4.6.1 Setup 84

4.6.2 Bag-of-words text data 85

4.6.3 Collaborative filtering 93

4.7 Discussion 96

5 Supervised fine-tuning of deep generative models 97

5.1 Introduction 98

5.2 From representation learning to reasoning 100

5.2.1 Data model 100

5.2.2 Reasoning model 102

5.2.3 Bayes factor 102

5.3 Hierarchical models with compound priors 103

5.4 Latent Reasoning Networks 105

5.5 Learning 107

5.6 Evaluation 108

5.6.1 Learning $p(z|\mathcal{Q})$ 109

5.6.2 Changing inductive biases at test-time 110

5.6.3 Modeling high-dimensional data 111

5.6.4 Few-shot learning with the Bayes factor 111

5.7 Related work 113

5.8 Discussion 115

6 Deep Markov Models **117**

6.1 Introduction 118

6.2 Setup 120

6.3 A factorized variational lower bound 121

6.3.1 Simplifying the lower bounds 123

6.3.2 Analytic forms of the KL divergence 125

6.3.3 Learning with gradient ascent 126

6.4 Structured Inference Networks 127

6.5 Deep Markov Models 130

6.6 Evaluation 131

6.6.1 Synthetic data 132

6.6.2 Polyphonic music 134

6.6.3 EHR Patient Data 138

6.7 Discussion 142

7 Inductive biases for clinical data 145

7.1 Introduction 145

7.2 Setup 147

7.2.1 First Order Markov Models (FOMMs) 148

7.2.2 Gated Recurrent Neural Network (GRUs) 149

7.2.3 State Space Models (SSMs) 150

7.2.4 Missing data 151

7.2.5 Pharmacokinetic-Pharmacodynamic (PK-PD) models 151

7.3 Intervention Effect Functions for clinical data 153

7.3.1 Capturing lines of therapy with local and global clocks 154

7.3.2 Domain expert IEF modules for clinical data 155

7.3.3 PK-PD Intervention Effect Function 157

7.4 Datasets 158

7.4.1 Synthetic data 158

7.4.2 Multiple Myleoma - ML-MMRF 160

7.5 Evaluation 162

7.5.1 Quantitative analysis 163

7.5.2 Qualitative Analysis 170

7.6 Discussion 175

8 Latent Representations of Privileged Information 177

8.1 Introduction 177

8.2 Setup 179

8.3 Privileged Information Variational Autoencoder 180

8.4 Evaluation 183

8.4.1 Synthetic Data 183

8.4.2 Evaluation 184

8.5 Related work 187

8.6 Discussion 188

9 Conclusion 191

9.1 Future directions for deep generative modeling 192

9.2 Future directions for machine learning in healthcare 194

A Model configurations 197

A.0.1 Pinwheel Dataset 197

A.0.2 MiniImagenet Dataset 198

A.0.3 MNIST Dataset 199

B Model configurations 201

List of Figures

1-1 **Patient data:** Left (clinical observations), Middle (centers of care), Right (treatments provided to patients) 30

1-2 **Patient data across scales of the human body:** From bottom to top, we depict patient data as manifested in the various scales of the human body, from micro scale to macro scale. 30

1-3 **Sequential patient data:** When tracking the progression of diseases, doctors characterize progression of disease as a function of how the patient's clinical observations vary with time. 31

2-1 **Undirected graphical models:** Nodes shaded in grey are observed random variables, while those with a white background denote unobserved or latent random variables 37

2-2 **Directed graphical nodels:** Nodes shaded in grey are observed random variables, while those with a white background denote unobserved or latent random variables 37

2-3 **Bayesian networks for supervised and unsupervised Learning:** Nodes shaded in grey are observed random variables, while those with a white background denote unobserved or latent random variables. On the left is a Bayesian network for supervised learning where x denote the inputs and y denote the random variables corresponding to the labels. On the right is a Bayesian network that characterizes a large class of latent factor models used in unsupervised learning where x is the data being modeled and z are the latent factors (or causes) that influence the data. Under the manifold hypothesis(Fefferman *et al.* , 2016), z is posited to have a lower-dimensionality than x, i.e. the domain of the latent variable z is lower-dimensional but suffices to explain variation in the higher-dimensional x. 41

2-4 **Convolutional neural networks**: On the left is an input image X that is transformed via parameteric, nonlinear functions (such as convolutional operations) to yield the vector on the right, a set of class probabilities corresponding to a distribution over probabilities of each label. 42

2-5 **Stochastic Variational Inference (SVI)** (Hoffman *et al.* , 2013) ϕ denote the variational parameters which are optimized prior to deriving gradients with respect to the model parameters θ 48

2-6 **Amortized Variational Inference (AVI)** (Rezende *et al.* , 2014; Kingma & Welling, 2014) ϕ denotes the parameters of an inference network which is used to predict the variational parameters that are subsequently used to evaluate the variational lower bound. 49

2-7 **Nonlinear factor analysis:** The model comprises a single latent variable z with the conditional probability $p(x|z)$ defined by a deep neural network with parameter θ. On the right, $q_\phi(z|x)$, the inference network, parameterized by ϕ, is used to predict variational parameters used at train and test time inference. When paired with an inference network, the resulting coupled model is known as a variational autoencoder. 50

4-1 **Learning nonlinear factor analysis with an inference network: [Left]** The generative model contains a single latent variable z. The conditional probability $p(x|z;\theta)$ parameterized by a deep neural network. **[Right]** The inference network $q_\phi(z|x)$ is used for inference at train and test time. . . . 74

4-2 **From patient history to a bag of diagnosis codes:** On the left is a depiction of a patient's history (outpatient in green and inpatient in red). On the right is how such a history would appear to machine learning models; as collections of diagnosis codes. 74

4-3 **Lower bounds in variational learning:** To estimate θ, we maximize a lower bound on $\log p(x;\theta)$. $\mathcal{L}(x;\theta,\psi(x))$ denotes the standard training objective used by VAEs. The tightness of this bound (relative to $\mathcal{L}(x;\theta,\psi^*)$) depends on the inference network. The x-axis is θ. 78

4-4 **Parameter estimation in NFA with a hybrid inference algorithm** . 80

4-5 **Mechanics of learning:** Best viewed in color. **(Left and Middle)** For the Wikipedia dataset, we visualize upper bounds on training and held-out perplexity (*evaluated* with ψ^*) viewed as a function of epochs. Items in the legend corresponds to choices of training method. **(Right)** Sorted log-singular values of $\nabla_z \log p(x|z)$ on Wikipedia (left) on RCV1 (right) for different training methods. The x-axis is latent dimension. The legend is identical to that in Fig. 4-5a. 85

4-6 **Decrease in perplexity versus sparsity:** We plot the relative drop in perplexity obtained by training with ψ^* instead of $\psi(x)$ against varying levels of sparsity in the Wikipedia data. On the y-axis, we plot $\frac{P_{[3-\psi(x)]}-P_{[3-\psi^*]}}{P_{[3-\psi(x)]}}$; P denotes the bound on perplexity (evaluated with ψ^*) and the subscript denotes the model and method used during training. Each point on the x-axis is a restriction of the dataset to the top L most frequently occurring words (number of features). 87

4-7 **Late versus early optimization of $\psi(x)$:** Fig. 4-7a (4-7b) denote the train (held-out) perplexity for three-layered models trained on the Wikipedia data in the following scenarios: ψ^* is used for training for the first ten epochs following which $\psi(x)$ is used (denoted "ψ^* then $\psi(x)$") and vice versa (denoted "$\psi(x)$ then ψ^*"). Fig. 4-7c (Left) depicts the number of singular values of the Jacobian matrix $\nabla_z \log p(x|z)$ with value greater than 1 as a function of training epochs for each of the two aforementioned methodologies. Fig. 4-7c (Right) plots the sorted log-singular values of the Jacobian matrix corresponding to the final model under each training strategy. 88

4-8 **KL divergence and rare word counts:** We plot the values of $\mathrm{KL}(\psi(x)\|\psi^*)$ versus the number of rare words. We zoom into the plot and reduce the opacity of the train points to better see the held-out points. The Spearman ρ correlation coefficient is computed between the two across $20,000$ points. We find a positive correlation. 89

4-9 **Normalized KL and Rare Word Counts:** Fig. 4-9a depicts percentage of times words appear in the Wikipedia dataset (sorted by frequency). The dotted line in blue denotes the marker for a word that has a 5% occurrence in documents. In Fig. 4-9b, 4-9c, we superimpose (1) the normalized (to be between 0 and 1) values of $\mathrm{KL}(\psi(x)\|\psi^*)$ and (2) the normalized number of rare words (sorted by value of the KL-divergence) for $20,000$ points (on the x-axis) randomly sampled from the train and held-out data. 90

4-10 **20Newsgroups - training and held-out bounds:** Fig. 4-10a, 4-10b denotes the train (held-out) perplexity for different models. Fig. 4-10c depicts the log-singular values of the Jacobian matrix for the trained models. 90

4-11 **RCV1 - training and held-out bounds:** Fig. 4-11a, 4-11b denotes the train (held-out) perplexity for different models. Fig. 4-11c depicts the log-singular values of the Jacobian matrix for the trained models. 91

4-12 **KL annealing vs learning with ψ^*** Fig. 4-12a, 4-12b denotes the train (held-out) perplexity for different training methods. The suffix at the end of the model configuration denotes the number of parameter updates that it took for the KL divergence in Equation 4.2 to be annealed from 0 to 1. 3-ψ^*-50k denotes that it took 50000 parameter updates before $-\mathcal{L}(x;\theta,\psi(x))$ was used as the loss function. Fig. 4-10c depicts the log-singular values of the Jacobian matrix for the trained models. 92

4-13 **Varying the depth of $q_\phi(z|x)$:** Fig. 4-12a (4-12b) denotes the train (held-out) perplexity for a three-layer generative model learned with inference networks of varying depth. The notation q3-ψ^* denotes that the inference network contained a two-layer intermediate hidden layer $h(x) = \mathrm{MLP}(x;\phi_0)$ followed by $\mu(x) = W_\mu h(x), \log\Sigma(x) = W_{\log\Sigma}h(x)$. 93

5-1 **Comparing objects in representational space:** On the left is a target set that will be ranked based on similarity to the query Q (right). The colour of each object is matched to a distribution in representation space. In orange is the output of the *latent reasoning network* – it represents the common factor of variation shared by $\mathcal{Q}$. The black chair should rank higher than the black table; here its distribution (in representation space) overlaps more with the output of the latent reasoning network. 98

5-2 **Hypothesis testing with deep generative models:** (a) The **Reasoning Model**, here, depicting the hypothesis that the set $\{x_t, \mathcal{Q} = \{x_1, x_2\}\}$ was generated jointly; (b) the two figures represent the hypothesis that x_t and $\mathcal{Q}$ were generated independently under different realizations of w (the random variable that captures the property shared across datapoints). . . 100

5-3 **Latent Reasoning Networks (LRN) and loss function:** On the left is a diagrammatic representation of $p_{\mathbf{rm}}(z_t|\mathcal{Q})$. On the right is a depiction of Monte-Carlo sampling (with samples from the LRN) to evaluate Bayes factor. x_i is a point similar to those in the query $\mathcal{Q} = \{x_1, x_2, x_3\}$, while x_{ns} is not. We suppress subscripts in the figure. 105

5-4 **Qualitative evaluation on pinwheel data:** Studying how the latent space of the data changes over the course of fine-tuning on the synthetic, pinwheel dataset. 109

5-5 **Qualitative evaluation on MNIST:** Studying the effect of fine-tuning the latent space of the data model on MNIST. 112

6-1 **Generative Models of Sequential Data:** (**Top Left**) Hidden Markov Model (HMM), (**Top Right**) Deep Markov Model (DMM) ■ denotes the neural networks used in DMMs for the emission and transition functions. (**Bottom**) Recurrent Neural Network (RNN), $\Diamond$ denotes a deterministic intermediate representation. Code for learning DMMs and reproducing our results may be found at: `github.com/clinicalml/structuredinference` 119

6-2 **Structured Inference Networks: MF-LR** and **ST-LR** variational approximations for a sequence of length 3, using a bi-directional recurrent neural net (BRNN). The BRNN takes as input the sequence $(x_1, \ldots x_3)$, and through a series of non-linearities denoted by the blue arrows it forms a sequence of hidden states summarizing information from the left and right (h_t^{left} and h_t^{right}) respectively. Then through a further sequence of non-linearities which we call the "combiner function" (marked (a) above), and denoted by the red arrows, it outputs two vectors μ and Σ, parameterizing the mean and diagonal covariance of $q_\phi(z_t|z_{t-1}, \vec{x})$ of Eq. 6.5. Samples $\hat{z}_t$ are drawn from $q_\phi(z_t|z_{t-1}, \vec{x})$, as indicated by the black dashed arrows. For the structured variational models **ST-LR**, the samples $\hat{z}_t$ are fed into the computation of μ_{t+1} and Σ_{t+1}, as indicated by the red arrows with the label (a). The mean-field model does *not* have these arrows, and therefore computes $q_\phi(z_t|\vec{x})$. We use $\hat{z}_0 = \vec{0}$. The inference network for **DKS** (ST-R) is structured like that of ST-LR except without the RNN from the past. 130

6-3 **Synthetic evaluation:** (**Top & Bottom**) Compiled inference for a *fixed* linear GSSM: $z_t \sim \mathcal{N}(z_{t-1} + 0.05, 10)$, $x_t \sim \mathcal{N}(0.5 z_t, 20)$. The training set comprised $N = 5000$ one-dimensional observations of sequence length $T = 25$. **(Top left)** RMSE with respect to true z^* that generated the data. **(Top right)** Variational bound during training. The results on held-out data are very similar (see supplementary material). **(Bottom four plots)** Visualizing inference in two sequences (denoted (1) and (2)); Left panels show the Latent Space of variables z, right panels show the Observations x. Observations are generated by the application of the emission function to the posterior shown in Latent Space. Shading denotes standard deviations. 133

6-4 **Parameter estimation:** Learning parameters α, β in a two-dimensional non-linear GSSM. $N = 5000, T = 25$ $\vec{z}_t \sim \mathcal{N}([0.2 z_{t-1}^0 + \tanh(\alpha z_{t-1}^1); 0.2 z_{t-1}^1 + \sin(\beta z_{t-1}^0)], 1.0)$ $\vec{x}_t \sim \mathcal{N}(0.5 \vec{z}_t, 0.1)$ where $\vec{z}$ denotes a vector, [] denotes concatenation and superscript denotes indexing. 133

6-5 **Inference in a linear SSM on held-out data:** Performance of inference networks on held-out data using a generative model with Linear Emission and Linear Transition 134

6-6 **Inference in a don-linear SSM:** Performance of inference networks trained with data from a Linear Emission and Non-linear Transition SSM135

6-7 **Inference on non-linear synthetic data:** Visualizing inference on training data. Generative Models: (a) Linear Emission and Non-linear Transition z^* denotes the latent variable that generated the observation. x denotes the true data. We compare against the results obtained by a smoothed Unscented Kalman Filter (UKF) (Wan & Van Der Merwe, 2000). The column denoted “Observations" denotes the result of applying the emission function of the respective generative model on the posterior estimates shown in the column “Latent Space". The shaded areas surrounding each curve μ denotes $\mu \pm \sigma$ for each plot. 136

6-8 Two samples from the DMM trained on JSB Chorales 137

6-9 **DMM for medical data:** The DMM (from Fig. 6-1) is augmented with external actions u_t representing medications presented to the patient. z_t is the latent state of the patient. x_t are the observations that we model. Since both u_t and x_t are always assumed observed, the conditional distribution $p(u_t|x_1, \dots, x_{t-1})$ may be ignored during learning. 140

6-10 **Left two plots; Estimating counterfactuals with DMM:** The x-axis denotes the number of 3-month intervals after prescription of Metformin. The y-axis denotes the proportion of patients (out of a test set size of 800) who, after their first prescription of Metformin, experienced a high level of A1C. In each tuple of bar plots at every time step, the left aligned bar plots (green) represent the population that received diabetes medication while the right aligned bar plots (red) represent the population that did not receive diabetes medication. **(Rightmost plot)** Upper bound on negative-log likelihood for different DMMs trained on the medical data. (T) denotes “transition”, (E) denotes “emission”, (L) denotes “linear” and (NL) denotes “non-linear”. . . 141

6-11 **Patient data generated by a DMM** Samples of a patient generated by the model. The x-axis denotes time and the y-axis denotes the observations. The intensity of the color denotes its value between zero and one 144

7-1 **Patient Data (Left):** Illustration of data from a multiple myeloma patient. Baseline (static) data typically consists of genomics, demographics, and initial labs. Longitudinal data typically includes laboratory values (e.g. serum IgG) and treatments. Baseline data is usually complete, but longitudinal measurements are frequently missing at various time points. The data tells a rich story of a patient's disease trajectory and the resulting treatment decisions. For example, a deviation of a lab value from a healthy range (e.g. spike in serum IgG) might prompt a move to the next line of therapy. Missing data (e.g. points in red) in this case are forward filled. **Unsupervised Models of Sequential Data (Right):** We show a State Space Model (SSM) of $\mathbf{X}$ (the longitudinal biomarkers) conditioned on B (genetics, denographics) and $\mathbf{U}$ (binary indicators of treatment and line of therapy). The rectangle depicts the $\mathbb{IEF}(S_{t-1}, U_{t-1}, B)$, where $S_{t-1} = Z_{t-1}$. 148

7-2 **Pharmacodynamic-Pharmacokinetic Treatment Effect Functions:** Visualizing PK/PD treatment response models. Curves denote the scalar biomarker being modeled and vertical lines denote treatment. *Left:* Log Cell Kill. The various curves (green, yellow, red) represent different parameterizations of the function. Here, (for visualization purposes) a single treatment is always present throughout time, but may be under a different line of therapy based on the shaded region. For each line, a sharp decline is followed by a rise in tumor volume, prompting a change in therapy line. Each curve corresponds to distinct rates of biomarker growth, parametrized by ρ. *Right:* Biomarker value under the Treatment Exponential model. After maintaining the response with treatments, a regression towards baseline (in blue; depicting what would have happened had no treatment been prescribed) occurs when treatment is stopped. 152

7-3 **Visualization of synthetic data:** Left: A visualization of "patient"'s baseline data (colored and marked by patient subtype). Right four plots: Examples of patient's longitudinal trajectories along with treatment response. The blue and green longitudinal data denote two diffrent patient biomarkers. Gray-dotted line represents intervention. The subtypes may, optionally, be correlated with patient outcomes as highlighted using the values of y. We do not use the outcomes in this chapter, but do so later in the thesis. 159

7-4 **Visualizations of learned SSM models**: (a) *Synthetic*: Forward samples (conditioned only on B) from $\mathbf{SSM}_{\text{PK-PD}}$ (o), $\mathbf{SSM}_{\text{Linear}}$ (x), $\mathbf{SSM}_{\text{PK-PD}}$ without local clocks (△), for a single patient. Blue circles (o) denote ground truth. The markers above the trajectories represent treatments prescribed across time. (b) *ML-MMRF*: We visualize the TSNE representations of each held-out patient's α_1 parameter (in the TE module) at the start of treatment and three years in. (c) *ML-MMRF*: For $\mathbf{SSM}_{\text{PK-PD}}$, we visualize weights, $\boldsymbol{\delta}$, on each domain expert module (LIN, LC, TE) across state space dimensions. (d) *ML-MMRF*: Each column is a different biomarker containing forward samples (conditioned only on B) from $\mathbf{SSM}_{\text{PK-PD}}$ (o) and $\mathbf{SSM}_{\text{linear}}$ (x) of a single patient. As in the synthetic samples, blue circles denote ground truth, and the markers above the trajectories represent treatments prescribed across time. y-axis shows biomarker levels (normalized to be between -8 and 8). 164

7-5 **a) NLL estimates via importance sampling**: We estimate the NLL of $\mathbf{SSM}_{\text{PK-PD}}$ and $\mathbf{SSM}_{\text{Linear}}$ for each feature, summed over all time points and averaged over all patients. **b) Condition on 6 months, forward sample 1 year:** We show L1 prediction error for forward samples over a 1 year time window conditioned on 6 months of patient data. At each time point, we compute the L1 error with the observed biomarker and sum these errors (excluding predictions for missing biomarker values) over the prediction window. We employ this procedure for each patient. **c) Condition on 6 months, sample forward 2 years:** We report L1 error for forward samples over a 2 year window conditioned on 6 months of patient data. **d) Condition on 2 years, sample forward 1 year:** Finally, we report L1 error for forward samples over a 1 year time window conditioned on 2 years of patient data. 169

7-6 **Samples from learned SSM models with differing conditioning strategies**: We visualize samples from $\mathbf{SSM}_{\text{PK-PD}}$ (*o*) and $\mathbf{SSM}_{\text{linear}}$ (x). Each row corresponds to a single patient, whereas each column represents a different biomarker for that patient. **a)**: We condition on 6 months of patient data and forward sample 2 years. **b)**: We condition on 1 year of patient data and forward sample 1 year. **c)**: We condition on data corresponding to the patient's first line of therapy and then forward sample the extent of their second and third line therapies. The blue circles denote ground truth, and the markers above the trajectories represent treatments prescribed across time. 173

7-7 **a),b) Heatmaps showing directional derivative of expected longitudinal values:** Here, we depict two heatmaps showing the directional derivatives of the expected longitudinal data with respect to VD (**a)**) and RVD (**b)**), two common first line therapies in multiple myeloma. Red boxes surround hemoglobin, creatinine, and platelet count, covariates that display the most differences between the two therapies over time. This analysis was done on $\mathbf{SSM}_{\text{PK-PD}}$. **c) Weights on the linear model that maps treatment to α_1:** We visualize the weight matrix of the linear function that maps the treatment signal to α_1, which varies across the state space dimension, in $\mathbf{SSM}_{\text{PK-PD}}$. 174

7-8 α_{1t} **Visualizations:** We visualize the TSNE representations of each held-out patient's α_1 parameter (in TE module used in trained $\mathbf{SSM}_{\text{PK-PD}}$) over multiple time points. 174

8-1 **Learning with post-treatment information:** (a) prediction of outcomes from baseline data only. (b) the Privileged Information Variational Autoencoder (PIVAE) (c) the PIVAE's inference network. 181

8-2 **Visualizing synthetic data:** Left: A visualization of patient's baseline data (coloured and marked by patient subtype). Each quadrant is annotated with [subtype] (time-to-death). Right four plots: For patients from each of the subtypes, an example of their longitudinal trajectories. The solid lines are trajectories had there been no treatment, while the dotted lines over time represent trajectories with treatment response. The dashed vertical line represents the therapy given at a particular point in time. 185

8-3 **Synthetic (held-out) data:** (a) depicts the delta distribution implied by Z under a supervised PIVAE while (b) $\sum_n q(Z_n|X, U)$, (c) $\sum_n p(Z_n|B, X_0, U_0)$ visualize the corresponding distributions from an unsupervised PIVAE. (d), (e) visualize different accuracy metrics comparing the PIVAE to various baselines. 185

8-4 **Visualizing the learned model** (a) Visualization of h_{global} (each row corresponds to the averaged, binarized h_{global} of a patient within a subtype); (b, c, d, e) for a single patient from each subtype, we sample the patient's biomarker from the generative model (conditioned on their baseline data), where we see a good fit to the ground truth 187

List of Tables

3.1 **Jacobian vectors:** The functional form of the Jacobian vectors for feature i as defined in Eq. 3.3 when $p(x_i = 1|z)$ is defined as in Eq. 3.1. 60

3.2 **Word embeddings (nearest neighbors):** We visualize nearest neighbors of word embeddings (excluding plurals of the query) 62

3.3 **Word embeddings (polysemy):** We visualize the nearest neighbors under the Jacobian vector induced by the posterior distribution of a document created based on the context word. 63

3.4 **Semantic similarity on text data:** A higher number is better. In Table 3.4a, 3.4b, the baseline results are taken from Huang *et al.* (2012). C&W uses embeddings from the language model of Collobert & Weston (2008). Glove corresponds to embeddings by Pennington *et al.* (2014). ρ corresponds to Spearman rho-correlation. 63

3.5 **Discriminative ability of Jacobian vectors:** Glove corresponds to embeddings by (Pennington *et al.* , 2014). (Stanford Sentiment Treebank) SST-fine corresponds to the fine grained classification task of predicting one of eight different sentiments while SST-binary corresponds to predicting a positive or negative sentiment for the sentence. 65

3.6 **Medical embeddings (nearest neighbors):** Nearest neighbors of some diagnosis codes (ignoring duplicates). Metformin (and it's neighbors) are diabetic drugs. A contour meter measures blood glucose. Spiriva and it's neighbors are drugs used for treating chronic obstructive pulmonary disease (COPD). 66

3.7 **Medical analogies:** We perform analogical reasoning with embeddings of medical codes. If we know a drug used to treat a disease, we can use their relationship in vector space to find unknown drugs associated with a different disease. Queries take the form Code 1$\rightarrow$Code 2 $\implies$ Code 3$\rightarrow$?. Sicca syndrome or Sjogren's disease is an immune disease treated with Evoxac and Methotrexate is commonly used to treat Rheumatoid Arthiritis. "Leg Varicosity" denotes the presence of swollen veins under the skin. "Ligation of angioaccess arteriovenous fistula" denotes the tying of a passage between an artery and a vein. 66

3.8 **Medical embeddings (clustering):** We visualize some topical clusters of diagnosis codes. 66

3.9 **Medical embeddings: Medical Relatedness Measure (MRM)** We evaluating embeddings using medical (NDF-RT and CCS) ontologies. SCUIs result from the method developed by Choi *et al.* (2016c) applied to data in Finlayson *et al.* (2014). 68

3.10 **Qualitative evaluation of movie embeddings:** We evaluate $\mathcal{J}^{\text{log}}_{\textbf{mean}}$ using 100 Monte-Carlo samples to perform the evaluation in Tables 3.10a and 3.10b. 71

4.1 **Test perplexity on RCV1: Left: Baselines** Legend: LDA (Blei *et al.* , 2003), Replicated Softmax (RSM) (Hinton & Salakhutdinov, 2009), Sigmoid Belief Networks (SBN) and Deep Autoregressive Networks (DARN) (Mnih & Gregor, 2014), Neural Variational Document Model (NVDM) (Miao *et al.* , 2016). K denotes the latent dimension in our notation. **Right:** NFA on text data with $K = 100$. We vary the features presented to the inference network $q_\phi(z|x)$ during learning between: normalized count vectors ($\frac{x}{\sum_{i=1}^{V} x_i}$, denoted "norm") and normalized TF-IDF 86

4.2 **Test perplexity on 20newsgroups: Left: Baselines** Legend: LDA (Blei *et al.* , 2003), Replicated Softmax (RSM) (Hinton & Salakhutdinov, 2009), Sigmoid Belief Networks (SBN) and Deep Autoregressive Networks (DARN) (Mnih & Gregor, 2014), Neural Variational Document Model (NVDM) (Miao *et al.* , 2016). K denotes the latent dimension in our notation. **Right:** NFA on text data with $K = 100$. We vary the features presented to the inference network $q_\phi(z|x)$ during learning between: normalized count vectors ($\frac{x}{\sum_{i=1}^{V} x_i}$, denoted "norm") and normalized TF-IDF (denoted "tfidf") features. 91

4.3 **Recall and NDCG on recommender systems:** "2-ψ^*-tfidf" denotes a two-layer (one hidden layer and one output layer) generative model. Standard errors are around 0.002 for ML-20M and 0.001 for Netflix. **Runtime:** WMF takes on the order of minutes [ML-20M & Netflix]; CDAE and NFA ($\psi(x)$) take 8 hours [ML-20M] and 32.5 hours [Netflix] for 150 epochs; NFA (ψ^*) takes takes 1.5 days [ML-20M] and 3 days [Netflix]; SLIM takes 3-4 days [ML-20M] and 2 weeks [Netflix]. 95

5.1 **5-way MiniImagenet task:** Accuracies for few-shot learning on the MiniImagenet task. The first row contains our method where higher is better. 113

6.1 **Inference networks:** BRNN refers to a Bidirectional RNN and comb.fxn is shorthand for combiner function. 128

6.2 **Comparing inference networks:** Test negative log-likelihood on polyphonic music of different inference networks trained on a DMM with a fixed structure (lower is better). The numbers inside parentheses are the variational bound. 137

6.3 **Evaluation against baselines:** Test negative log-likelihood (lower is better) on Polyphonic Music Generation dataset. **Table Legend**: RNN Boulanger-Lewandowski *et al.* (2012), LV-RNN Gu *et al.* (2015), STORN Bayer & Osendorfer (2014), TSBN, HMSBN Gan *et al.* (2015). 138

6.4 **Experiments with NADE Emission:** Test negative log-likelihood (lower is better) on Polyphonic Music Generation dataset. **Table Legend**: RNN-NADE (Boulanger-Lewandowski *et al.* , 2012) 139

7.1 **Synthetic data:** Lower is better. We report held-out negative log likelihood (or a bound on it for SSM models) with std. dev. on several model families to study generalization in the synthetic setting. 163

7.2 **ML-MMRF:** Higher is better. Each number is the fraction (with std. dev.) of held-out patients for which the model that uses $\mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}$ has a lower negative log-likelihood (or bound on it) than a model in the same family that uses a different IEF. 163

7.3 **Ablation experiments on the synthetic and ML-MMRF datasets**: **Top):** We study the effect of adding each domain expert module to $\mathbf{SSM}_{\text{PK-PD}}$. We report held-out bounds on negative log likelihood. **Bottom):** In this experiment, we study the effect of varying the tunable parameters of the domain expert modules in the SSM models vs keeping them fixed. 166

7.4 **Generalization on held-out data in ML-MMRF:** Lower is better. For the FOMM and RNN models, we report negative log-likelihood. For the SSM models, we report upper bounds on the negative log-likelihood. 167

7.5 **Pairwise comparison of models trained on ML-MMRF**: Higher is better. Each number is the fraction (with std. dev.) of held out patients for which the model that used $\mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}$ has a lower negative log-likelihood (or bound on it) than a model in the same family that uses a different IEF. We report fractions for each fold in ML-MMRF. 168

B.1 Synthetic Experiments (ReLU was used as the non-linearity) 202

B.2 Polyphonic Experiments (Tanh was used as the non-linearity). 202

B.3 Medical Experiments (Tanh was used as the non-linearity). We describe the "E:NL-T:NL" model. The observations were 48 dimensional of which there were 4 lab indicators that we treat separately to perform do-calculus. 202

Chapter 1

Introduction

1.1 Machine learning for healthcare

The ancient Egyptians, through the ritual practice of mummification, had a coarse grained but functional understanding of the taxonomy of the human body including body parts such as the brain, the heart, the blood and the role they played in keeping us alive. As civilisations evolved over the centuries, so too has our understanding of processes that govern the functioning of human bodies. We now know that the human body is among the most complex living organisms. At any point in time, there are millions of biochemical reactions happening simultaneously in the body, all of which together result in our instantaneous state of being. When one or more of these processes deviate from normalcy, we become ill.

Healthcare, broadly speaking, comprises the myriad of practices, policies and knowledge to treat our illnesses. The interventions in our present-day healthcare systems have been designed with the goal of reverting the state of our body from sick to healthy. We are constantly improving the way in which we treat diseases as we understand them better. Over the last several decades, bolstered by the ready availability of digital storage, healthcare institutions have collected, curated and organized patient data. We refer to this collection of data as Electronic Health Records (EHR). EHR data are collected by hospitals, insurance companies and clinics and record each patient's interaction with the healthcare system.

Figure 1-1, depicts the kind of data that is often collected. Clinical data may include diagnosis codes, x-ray imaging, clinical labs and occasionally patient genetics.

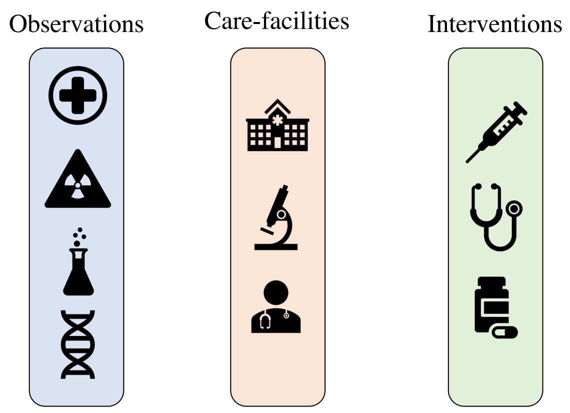


Figure 1-1: **Patient data:** Left (clinical observations), Middle (centers of care), Right (treatments provided to patients)

Depending on the source, the data may also contain information on where the data was tabulated – such as in hospitals (inpatient), external laboratories or clinics (outpatient). Finally, the data can include treatments and interventions prescribed such as surgery, check-ups or medication. Computational healthcare is concerned with the use of this data to improve our understanding of diseases and eventually improve clinical care.

This thesis lies at the intersection of computational healthcare and machine learning. The field of machine learning has seen enormous development over the last several decades. Advances in deep learning (LeCun *et al.* , 2015), powered by Graphical Processing Units (GPUs), enable practitioners to build supervised machine learning algorithms which make predictions from high-dimensional data using millions of datapoints. We have begun to see visible successes of machine learning in domains such as computer vision (Krizhevsky *et al.* , 2012), natural language processing (NLP) (Mikolov *et al.* , 2013b) and neural machine translation (Bahdanau *et al.* , 2014).

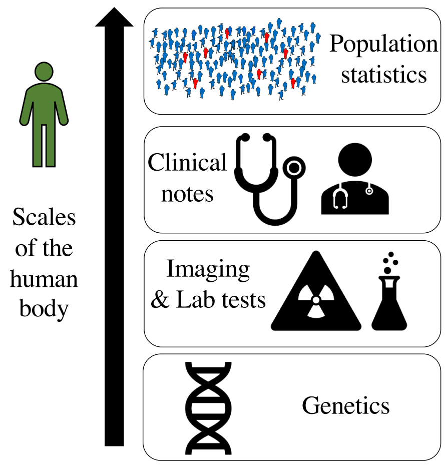


Figure 1-2: **Patient data across scales of the human body:** From bottom to top, we depict patient data as manifested in the various scales of the human body, from micro scale to macro scale.

The clarion call for *personalized medicine* has not gone unanswered. Deep learning has opened up new opportunities for improving the efficacy of clinical care. For example

(Yala *et al.* , 2019) use deep neural networks to predict pathologies from breast cancer images, while (Razavian *et al.* , 2015) build models to predict the early onset of diabetes from claims data. However, obtaining supervised data in healthcare is not feasible for every task – clinician time is valuable and labels can be expensive to obtain. This motivates the need for models that find patterns from unlabelled data, and use the underlying patterns to simplify predictive problems of interest so they may be answered even when labels are scarce.

Models that rise to such a task must, however, contend with the high-dimensionality of patient data that capture bio-chemical processes happening at multiple scales of the human body. In Figure 1-2, we provide a visual depiction of these phenomena. The dimensionality of the data at each level of granularity can span hundreds of thousands of features. We therefore turn to deep generative models, a class of statistical models that combines the representational power of deep learning with the probabilistic semantics of Bayesian networks. In contrast to discriminative models, which learn distributions of labels of interest conditioned on observations, generative models learn to model the joint distribution of all observed random variables.

This thesis presents new algorithms for unsupervised and supervised learning of deep generative models motivated by problems that arise in healthcare.

1.2 Challenges in healthcare

There are numerous challenges that practitioners face in building effective models of clinical data. Here, we highlight some of them.

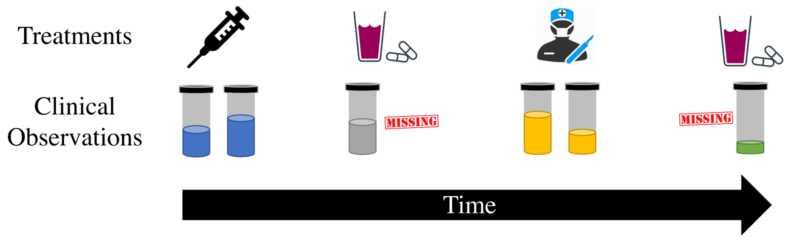


Figure 1-3: **Sequential patient data:** When tracking the progression of diseases, doctors characterize progression of disease as a function of how the patient's clinical observations vary with time.

Heterogeneity, sparsity, missingness, and high-dimensionality: Patient data is recorded in a heterogenous mix of modalities such as imaging, laboratory test results and diagnosis codes. Depending on the patient's reason for a visit to the clinic, some subset of his or her clinical data may be missing – consequently clinical data is often sparse. The sparsity may also be a consequence of missing data that can arise from a number of mechanisms in the data generating process (Mohan & Pearl, 2018).

Temporal data: Diseases change over time, these changes manifest in clinical observations and the treatments that are prescribed for them, as in Figure 1-3. To tackle predictive problems when data is dynamic, we need models capable of modeling time-varying high-dimensional clinical data.

Limited mechanistic knowledge: The human body comprises many phenomena at multiple scales – and the effects of disease over time are felt through many of them. Often, we lack fine-grained knowledge of how to characterize variation in clinical bio-markers throughout the course of disease.

Dataset sizes: While EHRs can constitute millions of patient records, to answer clinical queries for any *specific disease*, after selecting for relevant subset of patients, we are often left with only a few thousand patient records. It therefore becomes important to build data-efficient learning algorithms.

1.3 Contributions

In Chapter 2 we provide background on probabilistic inference, and parameter estimation in latent variable deep generative models. We highlights of some of the successes that deep generative models have seen and discuss why this family of models bears promise in tackling problems in healthcare. The chapters that form the bulk of this thesis are organized as follows:

Nonlinear Factor analysis: The first set of chapters studies unsupervised and supervised learning in the simplest latent variable, deep generative model : nonlinear factor analysis.

- Chapter 3: Generative models such as Latent Dirichlet Allocation (LDA) (Blei *et al.* , 2003) are inherently interpretable. The parameters that we interpret for LDA may be written as the gradient operator of the conditional likelihood of data. We make use of this idea and show how gradient operators may be used to introspect into the parameters of deep generative models. This is based on joint work with Matthew Hoffman.

- Chapter 4 studies a failure mode of the canonical learning algorithm for deep generative models when modeling high-dimensional data with long-tailed distributions. We propose a way to mitigate the underlying pathology encountered during learning. This is based on joint work with Dawen Liang and Matthew Hoffman.

- Chapter 5 depicts how the task of patient similarity may be posed as few-shot learning. To this end, we give new algorithms to fine-tune deep generative models using similarity judgements. This is based on joint work with Arjun Khandelwal, Rajesh Ranganath and David Sontag.

Deep Markov Models: The latter set of chapters studies models for unsupervised and supervised learning with high-dimensional, time-varying data.

- Chapter 6 introduces Deep Markov Models, nonlinear Gaussian state space models where the relationships between random variables are parameterized by neural networks. We propose a variational learning algorithm for the model and showcase its utility in modeling clinical data. Our work opens up new avenues for the use of deep generative models to tackle problems in clinical care. This is based on joint work with Uri Shalit and David Sontag.

- Chapter 7 proposes new neural architectures, inspired by pharmacology, which when used in Deep Markov Models, improve generalization of the model on patient data. This is based on joint work with Zeshan Hussain and David Sontag.

- Chapter 8 develops new methods for how deep generative models may be used to improve the predictive performance of classifiers by leveraging *privileged information*: information available at training time, but not at test time. This is based on joint work with Zeshan Hussain and David Sontag.

Finally, in Chapter 9, we conclude with a discussion on how the innovations made in this thesis can drive the next generation of predictive models in healthcare.

Chapter 2

Background

There are myriad ways to stratify and analyze the collective of methods used in machine learning. This thesis is best viewed from a probabilistic perspective (Murphy, 2012). This chapter is a primer on probability theory, graphical models, and deep generative models; the chapter introduces concepts and notation used throughout this thesis. For a more thorough introduction to random variables, and the statistical concepts that this thesis builds on, we refer the reader to (Wasserman, 2013).

2.1 Random variables and probabilities

Random variables are the atoms of machine learning. A random variable, as the word suggests, is a variable whose value (corresponding to an event of interest) is unknown but has the capacity to take multiple different values. The domain of a random variable may be discrete (like the side of a die), or continuous (such as how long it has been since the bus arrived). A probability is the chance of an event occurring, and a probability distribution describes the chances that a random variable takes any value in its domain. $P(X = 5)$ denotes the chances that the random variable X has of taking the assignment 5. A probability of zero denotes that the event cannot occur while a probability of one denotes the certainty of an event among all possible choices. Notationally, we will often use $P(x)$ in leiu of $P(X = x)$.

Probabilities may also be defined for multiple random variables; $P(X = x, Y = y)$ is the joint probability distribution denoting the probability that both random variables take their assigned values. Similarly, probability distributions of a random variable can also be affected by values taken on by other (typically related) random variables. A conditional probability is the probability of an event occurring given that another event has occurred. For example, the probability of a patient suffering from a heart attack increases conditional on the patient being obese. Two random variables are *independent* if conditioning on one has no consequence on the probability of the other. There are a few key rules that probability distributions follow that merit mention at this junction.

The product rule of probabilities states that the probability of two events can be written as the probability of the first event times the probability of the second event conditioned on knowing whether or not the first occurred. This rule generalizes to the chain rule of probabilities which may be written as: $P(X_1, X_2, X_3) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)$. An immediate consequence of this rule is Bayes rule, which forms the backbone of many inferential tasks. Bayes Rule states that: $P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$; i.e. given access to $P(Y|X), P(X), P(Y)$, it provides a mechanism by which we may invert conditional probabilities.

The sum rule states that $P(X \cup Y) = P(X) + P(Y) - P(X \cap Y)$ where $\cup$ denotes the union of events spanned by the random variables X, Y and $\cap$ denotes the intersection of the events. For mutually exclusive events, $P(X \cup Y) = P(X) + P(Y)$. A consequence of the sum rule is that the estimation of *marginal* probabilities $P(X)$ can be derived from the joint probability distribution $P(X, Y)$ as: $P(X) = \sum_y P(X, Y = y)$ when Y is discrete (for continuous random variables the sum would be replaced with an integral).

Finally, a probability density function, pdf for short, is a map from the assignment of a random variable onto a scalar proportional to the likelihood that the random variable takes the chosen assignment. For any event E, which constitutes values that the random variable may take: $P(X \in E) = \int_{x \in E} p(x)dx$ i.e. the probability density function characterizes how often random variable X lies in the set E.

The goal of a probabilistic treatment of machine learning is often to pose questions of interest to the practitioner using the language of probability; we refer to these questions as probabilistic queries. For example, supervised prediction corresponds to the evaluation of the conditional probability of Y, a random variable that represents the label, given covariates X: $P(Y|X)$. Similarly, the goal of unsupervised learning is to approximate $P(X)$, where X may be a vector valued random variable corresponding to high-dimensional data of interest. Queries from unsupervised models can create *new* examples of data by drawing samples via the probabilistic query $P(X)$.

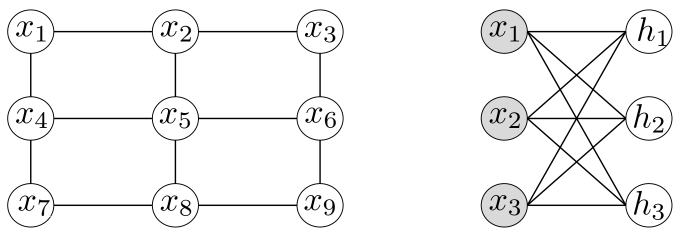


Figure 2-1: **Undirected graphical models:** Nodes shaded in grey are observed random variables, while those with a white background denote unobserved or latent random variables

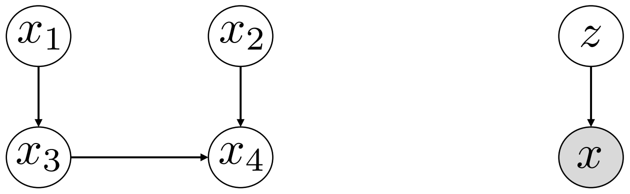


Figure 2-2: **Directed graphical nodels:** Nodes shaded in grey are observed random variables, while those with a white background denote unobserved or latent random variables

2.2 Graphical models

Most practical problems involve more than two random variables. Probability distributions over multiple random variables become unwieldy as the number of random variables grow. The relationships between random variables, such as which random variables are related, and which are not, can be difficult to track. The computation of probabilistic queries, such as conditional distributions is further complicated in the presence of a large number of random variables. To that end, graphical models, or PGMs (Koller *et al.* , 2009; Pearl, 1998), use graphs to represent probabilistic phenomena that span multiple random variables.

Graphs comprise nodes and edges. PGMs use nodes to represent random variables while edges represent probabilistic relationships that are either known or posited to exist. Random variables may be observed (i.e. the problem at hand tells us what values the observed random variables take), or latent (random variables whose values are unknown). There are two popular kinds of graphical models: undirected graphical models, also known as Markov Random Fields (MRFs) in Figure 2-1, and directed graphical models, or Bayesian networks, in Figure 2-2. But what advantages does the use of a graphical model confer upon the practitioner?

2.2.1 Structure as domain knowledge

There are several reasons why graphical models have seen tremendous success as a tool for probabilistic modelling. First, every graph structure over random variables implies a factorization on the joint distribution. For the undirected graphical model in Figure 2-1 (left), it can be shown that

$$P(X_1, \ldots, X_9) = \frac{1}{Z} \prod_{c_i \in \mathcal{C}} \phi_i(x_c),$$

where $\mathcal{C}$ are the set of all cliques in the graph (in this case, all pairs of nodes connected by an edge), where $\phi_i(x_c)$ denote clique potentials (a function that assigns a scalar to every assignment taken on by variables in the clique x_c) and Z is the normalization constant. Similarly, for the directed graphical model in Figure 2-2 (left), the joint distribution over the random variables factorizes as:

$$P(X_1, \ldots, X_4) = P(X_1)P(X_2)P(X_3|X_1)P(X_4|X_3, X_2),$$

An immediate consequence of the factorization of the joint distribution is that practioners need only track parameters associated with each of the clique potentials or conditional probabilities. For example, if all random variables were binary, then the joint distribution over random variables in Figure 2-1 (left) would naïvely be characterized by $2^9 - 1 = 511$ parameters. However, the graphical model has twelve cliques potentials, each of which can be represented via $2^2 - 1 = 3$ parameters resulting in 36 parameters: an order of magnitude in parameter savings.

Second, the exercise of creating the graphical model is often undertaken in conjunction with a domain expert. Doing so forces practitioners to think carefully about selecting the random variables in the problem, and decide how they are related. For example, the graphical model in Figure 2-1 (left) shows a grid structured model, which implies that the random variables exhibit spatial correlations, such as those among pixels in an image.

Finally, the use of a graph allows us to understand and take advantage of structural properties of the data generating distribution to simplify the computation of probabilistic queries. One concrete way in which they do so is via the simplification of independence statements.

2.2.2 Independence statements

Once a probabilistic graphical model has been created, we can borrow from the rich literature on graph theory to study the various properties that must hold among the distributions over random variables in the graph. Key among them are properties about which variables are independent from one another.

Marginal Independence: If there exists no edge between two random variables in a graph, then the random variables are said to be marginally independent. In directed graphs, there must exist no directed path between two random variables. For example, there are no random variables in Figure 2-1 (left) that are marginally independent of one another since the graph is connected. X_1 and X_2 in Figure 2-2 (left) are marginally independent of one another since one is not a parent of the other. For marginally independent random variables, we have that $P(X_1, X_2) = P(X_1)P(X_2)$.

Conditional Independence: Conditioning, or observing a random variable's value, is an important event that has ramifications about the independence properties of random variables in a graph. Conditional independence statements tell us when observing a set of random variables renders two random variables independent of one another. In Figure 2-1 (left), $X_1 \perp\!\!\!\perp X_3, X_5, X_{6,\ldots,9} | X_2, X_4$ since all the influence that X_1 has on the other random variables is via X_2, X_4. Similarly in directed graphical models, Figure 2-2 (left), $X_1 \perp\!\!\!\perp X_4 | X_3$. For conditionally independent random variables, we have that $P(X_1, X_4|X_3) = P(X_1|X_3)P(X_4|X_3)$.

In directed graphical models, there is a special form of conditioning that renders otherwise marginally independent variables dependent. This happens when conditioning on a common child. For example, $X_1 \not\perp\!\!\!\perp X_2 | X_4$ in Figure 2-2 (left) where X_4 is a common child of both X_1 and X_2. The rationale for this is as follows, consider the following: let X_1 capture whether a sprinkler is on, X_2 represent the probability of rain, and X_4 denote the grass being wet. Knowing that the grass is wet means that either the sprinkler was on rendering it less likely to have rained, or vice versa. This phenomenon is referred to as *explaining away*; in the aforementioned example, knowing the grass is wet allows rain to explain away the chance of the sprinkler being on and vice versa.

The Markov blanket of a random variable is a set of variables which if conditioned on, render a random variable independent of every other in a graph.

Definition 2.2.1. *For any random variables $X, Y \in G$, the Markov Blanket $\boldsymbol{MB}(X) \in$*

G is the minimal set of variables where

$$P(X|\boldsymbol{MB}(X), Y) = P(X|\boldsymbol{MB}(X))$$

For undirected graphical models, the Markov Blanket of a random variable are all its neighbors. For a directed graphical model, the Markov Blanket comprises a node's parents, its children, and its children's co-parents. In general, for directed graphical models, queries about whether two nodes are conditionally independent given the conditioning set and the graph can be verified in linear time (Shachter, 2013).

2.3 Bayesian networks

Thus far, our discussion has highlighted probabilistic graphical models as a means to represent probabilistic phenomena in the world by using graphs to capture known or posited relationships among random variables. We now turn to topics of practical interest and discuss how to parameterize and learn Bayesian networks from data. To ground our discussion henceforth, we will discuss two simple Bayesian networks that characterize a large swath of research done in supervised and unsupervised learning. In Figure 2-3 (left) we visualize a Bayesian network that captures many supervised models used in machine learning. In Figure 2-3 (right), we visualize a latent factor model, commonly used in unsupervised learning.

2.3.1 Parameterizations of Bayesian networks

The choices that practitioners make in selecting the parameterizations of Bayesian networks dictate the kind of model we obtain. Each choice of parameterization has an associated set of parameters that we will refer to using θ. We now discuss various choices for the conditional distributions in the Bayesian networks of Figure 2-3, and the models that result as a consequence of each choice.

Supervised Learning

In supervised learning, we are given access to a dataset $\mathcal{D} = \{(X_i, Y_i), \ldots, (X_n, Y_n)\}$ where X_i denotes a set of multi-variate covariates, and Y_i are the corresponding label

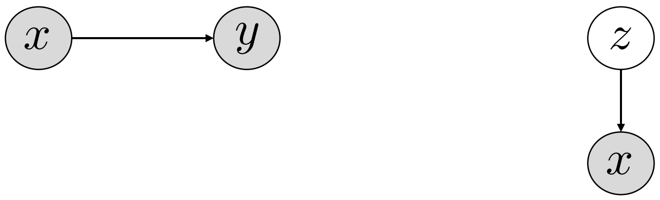


Figure 2-3: **Bayesian networks for supervised and unsupervised Learning:** Nodes shaded in grey are observed random variables, while those with a white background denote unobserved or latent random variables. On the left is a Bayesian network for supervised learning where x denote the inputs and y denote the random variables corresponding to the labels. On the right is a Bayesian network that characterizes a large class of latent factor models used in unsupervised learning where x is the data being modeled and z are the latent factors (or causes) that influence the data. Under the manifold hypothesis(Fefferman *et al.* , 2016), z is posited to have a lower-dimensionality than x, i.e. the domain of the latent variable z is lower-dimensional but suffices to explain variation in the higher-dimensional x.

which may be binary, categorical, or continuous valued. The goal of supervised learning is to obtain a model that, when given a new covariates X_k, predicts the corresponding outcome of interest Y_k. Figure 2-3(left) depicts the Bayesian network corresponding to several models commonly used for supervised learning. Here, θ denotes the parameters that dictate how the conditional distribution $P(Y|X;\theta)$ is decided.

Random forests: A classification tree is a sequence of rules corresponding to thresholds on various elements of X. For example if X was a binary, two dimensional random variable, with X^j denoting the jth dimension, and Y was a binary label, then the following represents a classification tree for this prediction problem where $\hat{Y}$ denotes the predicted label:

```
if $X^1 > 0.5$ then
    predict $\hat{Y} = 1$
else
    if $X^2 < 0.5$ then
        predict $\hat{Y} = 0$
    else
        predict $\hat{Y} = 1$
    end if
end if
```

When $P(Y|X)$ is parameterized by such a decision rule (where θ encodes both the thresholds and the dimension of X to threshold at each level of the tree), then the resulting model is a decision tree. An ensemble of decision trees is called a random

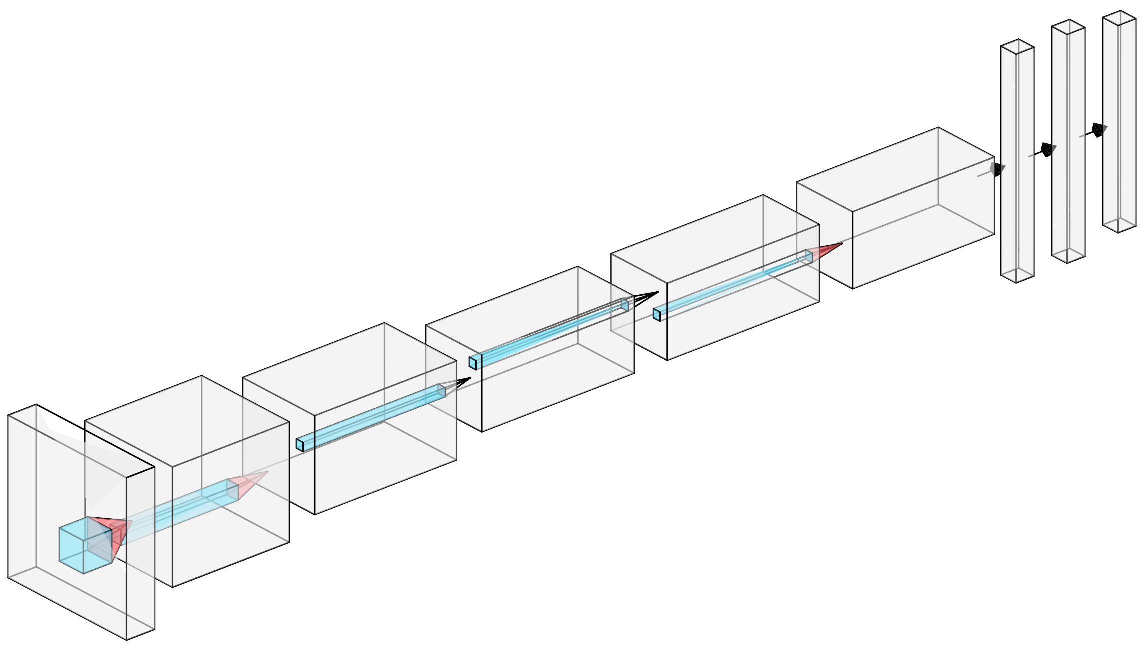


Figure 2-4: **Convolutional neural networks**: On the left is an input image X that is transformed via parameteric, nonlinear functions (such as convolutional operations) to yield the vector on the right, a set of class probabilities corresponding to a distribution over probabilities of each label.

forest.

The choice of parameterization, θ plays a large role in how well we can learn to make predictions from data. **Linear regression** is a model of continuous Y given covariates X where $P(Y|X) = \mathcal{N}(W^T X + b, \mathbf{I})$. **Logistic regression** is a model of binary Y given covariates X where $P(Y|X) = \frac{1}{1+\exp(W^T X + b)}$. In both of the aforementioned models, $\theta = \{W, b\}$.

Finally, while the above model families are *linear*, we may also parameterize $P(Y|X;\theta)$ using nonlinear functions. For example, with binary Y given covariates X we can have $P(Y|X) = \frac{1}{1+\exp(f(X;\theta))}$. There are many feasible choices for f but the one that we will discuss in detail is where $f(X;\theta)$ is a deep neural network.

Deep neural networks are a class of compositional, differentiable, parameteric functions: $f(X;\theta) = h_K(\ldots h_2(h_1(x;\theta_1);\theta_2)\ldots;\theta_K)$. Each h_k corresponds to a (potentially vector valued) function at layer k in the network, and each layer has parameters θ_k. The parameters of the model are $\theta = \{\theta_1, \theta_2, \ldots, \theta_K\}$. When h_k is the convolutional operation (LeCun *et al.* , 1998) followed by an elementwise non-linearity, $f(x)$ is a deep convolutional network. Figure 2-4 depicts a convolutional neural network. Although, two-layer neural networks can approximate any real-valued function to an arbitrary accuracy (Cybenko, 1989), practioners have found that deeper neural networks tend to yield better results. Hardware acceleration using Graphical Processing Units (GPUs) has enabled practioners to train neural networks that are hundreds of layers deep. In 2012, Krizhevsky *et al.* (2012) showed that deep convolutional neural networks, when trained on a large corpus of labelled data, were capable of detecting objects in unseen images with accuracies as high as 95%. Since then, deep neural networks have found success as powerful function approximators in diverse domains like models that play AlphaGo (Silver *et al.* , 2016) and self-driving cars (Bojarski *et al.* , 2016).

Unsupervised Learning

Unsupervised learning is the umbrella term used to describe a class of methods for modeling the likelihood of data. Given $\mathcal{D} = \{X_1, X_2, \ldots, X_n\}$ samples from some underlying distribution over data, the goal is to build a model to approximate the true data distribution $P(X)$. This task is often referred to as density estimation. The ethos of unsupervised learning is that a model which succeeds at density estimation can only do so by capturing the salient aspects of the dataset $\mathcal{D}$.

Good unsupervised models of data have several uses. They are used to generate synthetic data that appears as if it came from the true data distribution. They may be used for anomaly detection, i.e. given a parameteric model, we can use $P(X; \theta)$ to decide the likelihood that a new datapoint $\hat{X}$ could have come from the true data distribution. Finally, they may also be used to build exploratory tools of data. One way to do so is by using unsupervised models to learn low-dimensional representations of high-dimensional data.

Recall that Bayesian networks may be used to posit a data generation process for the observed data. Within that process, one or more of the variables in the network may be latent or unobserved. A common theme in many popular Bayesian networks is to use a low-dimensional, latent random variable as the parent of an observed, high-dimensional random variable. Although we do not directly observe latent variables, their values may be inferred via probabilistic inference from observed data.

There are many widely used latent variable models; here, we discuss two among them to set the stage for the work done in this thesis – both of them share the Bayesian network in Figure 2-3 (right).

Factor Analysis assumes the following generative process for high-dimensional continuous valued data:

$$z \sim \mathcal{N}(0; \mathbb{I}) \qquad x \sim \mathcal{N}(Wx + b; \Psi) \tag{2.1}$$

where $z \in \mathbb{R}^M,\ x \in \mathbb{R}^D,\ M < D$ and the parameters $\theta = \{W, b, \Psi\}$. When $\Psi = \sigma^2\mathbb{I}$, the model is known as probabilistic principal component analysis (PPCA) (Tipping & Bishop, 1999). The low-dimensional representations recovered under the model may be shown to converge to the principal components recovered by Principal Component Analysis (PCA) in the limit $\sigma \to 0$.

Nonlinear Factor Analysis generalizes factor analysis with non-linear transformations of the low-dimensional latent variable.

$$z \sim \mathcal{N}(0; \mathbb{I}) \qquad x \sim \Pi(f(z; \theta)) \tag{2.2}$$

where $z \in \mathbb{R}^M,\ x \in \mathbb{R}^D,\ M < D$. We use Π to denote an appropriate distribution depending on the kind of random variable being modelled. If x is a vector of high-dimensional binary data, then one choice for Π is a vector of probabilities, each corresponding to mean parameter of a Bernoulli distribution. There are many choices for f but of particular interest to the work done in this thesis is when f is a deep neural network with parameters θ. In this scenario, the resulting model is known as a deep generative model. When z is normally distributed, the Bayesian network is also referred to as a deep, latent Gaussian model (Rezende *et al.* , 2014).

Although this section provides a brief introduction to latent variable modeling, we emphasize that one can learn powerful generative models of data *without* the use of latent variables. One way to do so is by using the chain rule of probabilities to derive an auto-regressive decomposition of $P(X)$ over its dimensions as follows:

$$P(X; \theta) = \prod P(X^1; \theta) P(X^2 | X^1; \theta) P(X^3 | X^1, X^2; \theta) \ldots P(X^D | X^{<D}; \theta)$$

where $X^{<D} = \{X^1, \ldots, X^{D-1}\}$. By parameterizing each of the conditional distributions in the above decomposition of the joint probability, models such as PixelCNN++ (Salimans *et al.* , 2017), PixelRNN (Oord *et al.* , 2016a) and Wavenet (Oord *et al.* , 2016b) obtain impressive results when modelling high dimensional data such as images and speech.

Having discussed the various choices a practitioner has to parameterize a Bayesian network for both supervised and unsupervised learning problems, we turn to the question of estimating the parameters, or learning from data.

2.3.2 Learning

There are several guiding principles to learning the parameters of graphical models. We highlight three of them here. In each case, we will assume access to a dataset $\mathcal{D} = \{X_1, X_2, \ldots, X_n\}$ where X_i is the realization (or sample) from a parameteric distribution of random variable X driven by an unknown set of parameters θ.

The method of moments reduces the problem of estimating the parameters of a probability distribution into one of solving a system of equations. This method relies on uncovering the parameters governing a distribution via the *moments* of the distribution. The kth moment can be expressed as: $\mu_k = \mathbb{E}[X^k]$. Intuitively, moments quantify the shape of a distribution. For example, the first moment of a distribution is the mean (the average value that random variables under that distribution take), the second is the variance (the degree to which the distribution spreads about the mean), the third is the skewness (how tilted the distribution is) and the fourth moment is the kurtosis (the degree of peakiness of a distribution). For many distributions the moments may be expressed as a function of θ, the parameters of the distribution. Therefore, given (1) sufficiently many expressions of moments of the distribution using θ and (2) empirical estimates of each moment obtained using $\mathcal{D}$, we can solve for θ using k systems of equations of the form $\mu_k = \mathbb{E}[X^k]$. The complexity of the parametric distribution dictates the number of moments required to estimate θ. For samples drawn from a univariate Bernoulli random variable, a single moment suffices. More moments are necessary to estimate the parameters from distributions implied under certain classes of Bayesian networks such as Mixture Models (Anandkumar *et al.* , 2012), Noisy Or networks (Jernite *et al.* , 2013; Halpern & Sontag, 2013) and Hidden Markov Models (Hsu *et al.* , 2012).

Comparative density estimation uses comparisons between a model's prediction and observations from a dataset as a means to estimate model parameters. While the underlying goal of this methodology is to compare the distribution of data under a model with the true data distribution, in practice a variety of techniques are used to sidestep our lack of access to the latter, and in some cases the former. For example, (Dziugaite *et al.* , 2015; Li *et al.* , 2015b) derive gradient updates to θ based on how well the statistics of samples from a Bayesian network compare to the statistics from the data distribution in a Reproducing Kernel Hilbert Space (RKHS). Generative adversarial networks (Goodfellow *et al.* , 2014), derive gradients to θ using an auxiliary model (called a discriminator) to decide via classification if the samples under the generative model can be distinguished from samples in the dataset. A characteristic feature of this class of learning algorithms is that it is capable of operating in the *absence* of a parametric specification for the distribution of X. Models that one can sample from, but not necessarily evaluate the likelihood of, are often referred to as *implicit generative models*. We refer the reader to (Mohamed & Lakshminarayanan, 2016) for an overview of various techniques for learning implicit generative models and their relationship to one another.

Maximum likelihood estimation turns parameter estimation into an optimization problem. Specifically, given $\mathcal{D}$, the goal is to solve the following optimization:

$$\max_\theta \quad \underbrace{\prod_{i=1}^{N} p(X_i;\theta)}_{\text{likelihood of observing } \mathcal{D}}$$

and find model parameters θ such that the probability of observing $\mathcal{D}$ is as high as possible. In practice, we often use the logarithmic transformation of the probability density function of the dataset yielding the following optimization problem:

$$\max_\theta \log \prod_{i=1}^{N} p(X_i;\theta) = \max_\theta \sum_{i=1}^{N} \underbrace{\log p(X_i;\theta)}_{\text{log-likelihood of } \mathcal{D}}$$

2.3.3 Variational learning of latent variable models

For many classes of supervised and unsupervised models discussed above, the log-likelihood is a differentiable function of θ, the model parameters. Consequently, the optimization problem $\max_\theta \sum_{i=1}^{N} \log p(X_i;\theta)$ or $\max_\theta \sum_{i=1}^{N} \log p(Y_i|X_i;\theta)$ may be solved via stochastic gradient ascent. But this is not always the case. Learning can be challenging in latent variable models like those in Figure 2-3 (right) and will be the focus of this section where we consider learning parameters θ from a single datapoint $X = x$.

$$\log p(x;\theta) = \log \int_z p(x|z;\theta)p(z;\theta)dz \tag{2.3}$$

For models such as linear factor analysis, we can derive an analytic expression for the integral in Equation 2.3 as a function of the parameters θ.

However, when $p(x|z;\theta)$ is a non-linear function, the integral inside the logarithm is intractable. We therefore must resort to approximations to evaluate the log-likelihood. So how can we learn if the function that we use to evaluate the quality of a models' fit to data is not tractable? We use a surrogate to the likelihood function, in particular, a lower bound to it. In order to construct a lower bound to the likelihood function, we will require access to an auxillary distribution over the latent variables $q(z)$.

$$\begin{aligned}
\log p(x;\theta) &= \log \int_z p(x,z;\theta)dz = \log \int_z \frac{q(z)p(x,z;\theta)}{q(z)}dz \\
&\geq \int_z q(z) \log \frac{p(x,z;\theta)}{q(z)} \qquad (2.4) \\
&= \underbrace{\mathbb{E}_{q(z)}[\log p(x,z;\theta)] + \mathrm{H}(q(z))}_{\mathcal{L}(x,q(z);\theta)} \qquad (2.5)
\end{aligned}$$

where Equation 2.4 is due to Jensen's Inequality. The distribution $q(z)$ is known as the variational distribution and the lower bound in Equation 2.5 is called the variational lower bound or the evidence lower bound (ELBO). Note that while Equation 2.3 had an expectation *inside* the log, Equation 2.5 has the expectation outside. Consequently, as long as we can evaluate the entropy of the variational distribution and the log-probability of the joint distribution $\log p(x,z;\theta)$, we may use Monte-Carlo sampling to obtain an unbiased estimate of the lower-bound. If the resulting estimate is differentiable, then we can learn the model parameters via gradient ascent.

The practitioner is free to choose $q(z)$ and the ELBO is a valid lower bound on the log-likelihood of data for any choice of $q(z)$. However, it is easy enough to derive the best choice for $q(z)$ by studying at the difference between the log-likelihood and the variational lower bound:

$$\begin{aligned}
\log p(x) - \mathcal{L}(x,q(z)) &= \log p(x;\theta) - \int_z q(z) \log \frac{p(x,z)}{q(z)} \\
&= \int_z q(z) \log p(x) - \int_z q(z) \log \frac{p(x,z)}{q(z)} \\
&= \int_z q(z) \log \frac{p(x)q(z)}{p(x,z)} \\
&= \int_z q(z) \log \frac{p(x)q(z)}{p(z|x)p(x)} \\
&= \mathrm{KL}(q(z)||p(z|x)) \qquad (2.6)
\end{aligned}$$

Equation 2.6 tells us that gap between the log-likelihood under the model and the lower bound on it is the KL divergence between the variational distribution and the true posterior distribution $p(z|x)$. Intuitively, the posterior distribution represents the distribution over the latent variable most likely to give rise to the observed data. The practionioner therefore must choose $q(z)$ to be as close as possible to the true posterior distribution. In general this is a hard problem, and one where we must resort to approximations yet again.

Stochastic Variational Inference

Variational inference (VI) assumes that the variational distribution $q(z)$ lies within some tractable family of distributions. The desiderata that guide our choices for $q(z)$ are two-fold and stem from our desire to evaluate Equation 2.5. First, we must be able to sample from the variational distribution, and second, we must be able to evaluate its entropy. A common choice for the variational distribution is that it lies in the exponential family, for example a Gaussian distribution where $q(z; \phi) = \mathcal{N}(\mu, \Sigma)$ and $\phi = \{\mu, \Sigma\}$ are called the variational parameters.

However, even within our selection of variational distribution, there may be good and bad choices for the variational parameters for a datapoint. Stochastic variational inference (SVI) (Hoffman *et al.* , 2013) uses a gradient-based search procedure within the variational family to find the optimal variational parameters. Given the optimal variational parameters for a datapoint, we may proceed to derive gradients with respect to our model parameters θ. We visualize this two-stage procedure in Figure 2-5.

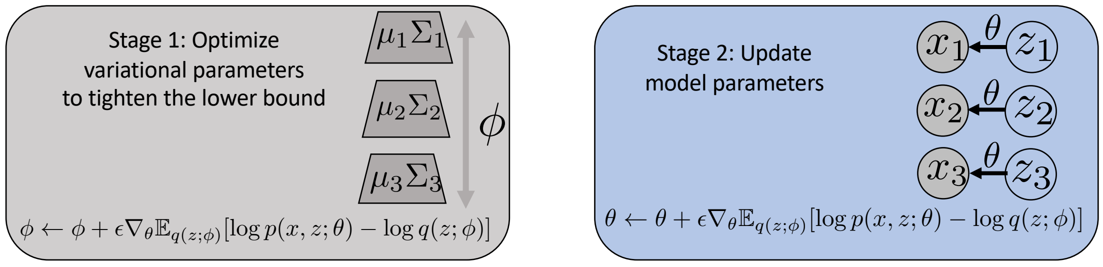

Stage 1: Optimize variational parameters to tighten the lower bound
$\mu_1 \Sigma_1$
$\mu_2 \Sigma_2$
$\mu_3 \Sigma_3$
ϕ
$\phi \leftarrow \phi + \epsilon \nabla_\theta \mathbb{E}_{q(z;\phi)}[\log p(x, z; \theta) - \log q(z; \phi)]$
Stage 2: Update model parameters
x_1 θ z_1
x_2 θ z_2
x_3 θ z_3
$\theta \leftarrow \theta + \epsilon \nabla_\theta \mathbb{E}_{q(z;\phi)}[\log p(x, z; \theta) - \log q(z; \phi)]$

Figure 2-5: **Stochastic Variational Inference (SVI)** (Hoffman *et al.* , 2013) ϕ denote the variational parameters which are optimized prior to deriving gradients with respect to the model parameters θ

Amortized Variational Inference

In SVI, each datapoint is assigned a variational parameter which is optimized during training time. In effect, the number of variational parameters that are tracked by the method scale with the number of datapoints. Furthermore, every new datapoint is assigned variational parameters that must be optimized prior to evaluating the variational bound. In 2013, (Rezende *et al.* , 2014; Kingma & Welling, 2014) derived

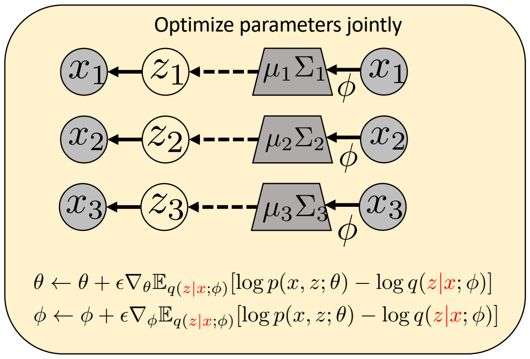

Optimize parameters jointly
x_1 z_1 $\mu_1 \Sigma_1$ ϕ x_1
x_2 z_2 $\mu_2 \Sigma_2$ ϕ x_2
x_3 z_3 $\mu_3 \Sigma_3$ ϕ x_3
$\theta \leftarrow \theta + \epsilon \nabla_\theta \mathbb{E}_{q(z|x;\phi)}[\log p(x,z;\theta) - \log q(z|x;\phi)]$
$\phi \leftarrow \phi + \epsilon \nabla_\phi \mathbb{E}_{q(z|x;\phi)}[\log p(x,z;\theta) - \log q(z|x;\phi)]$

Figure 2-6: **Amortized Variational Inference (AVI)** (Rezende *et al.* , 2014; Kingma & Welling, 2014) ϕ denotes the parameters of an inference network which is used to predict the variational parameters that are subsequently used to evaluate the variational lower bound.

a new method for probabilistic inference and learning in deep generative models reminiscent of the Wake-Sleep Algorithm (Hinton *et al.* , 1995). Rather than tracking N variational parameters, one for each datapoint, they proposed using a separate parametric function, an *inference network* with parameters ϕ, to predict the optimal variational parameters as a function of the data. i.e. they proposed the use of a conditional variational distribution $q(z|x;\phi) = \mathcal{N}(\mu(x;\phi), \Sigma(x;\phi))$ where $\mu(x;\phi), \Sigma(x;\phi)$ are functions parameterized by a neural network. Consequently, the variational bound may be derived as:

$$\log p(x;\theta) \geq \underbrace{\mathbb{E}_{q(z|x;\phi)}[\log p(x,z;\theta)] + \mathrm{H}(q(z|x;\phi))}_{\mathcal{L}(x,q(z|x;\phi);\theta)} \tag{2.7}$$

Relative to variational learning with SVI, an important difference of this approach was that rather than a two-stage approach to learning, their work resulted in a single stage approach where the model parameters θ and inference network parameters ϕ were jointly updated via gradient ascent on the variational lower bound as in Figure 2-6. The scheme of using an inference network to *predict* variational parameters was known as *Amortized Variational Inference* (AVI), since the inference network learned to amortize the solution to the optimization problem corresponding to finding the optimal variational parameters for a datapoint. The joint coupling of a model like non-linear factor analysis and its inference network is known as a variational autoencoder. Figure 2-7 depicts an example of a variational autoencoder.

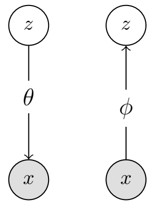


Figure 2-7: **Nonlinear factor analysis:** The model comprises a single latent variable z with the conditional probability $p(x|z)$ defined by a deep neural network with parameter θ. On the right, $q_\phi(z|x)$, the inference network, parameterized by ϕ, is used to predict variational parameters used at train and test time inference. When paired with an inference network, the resulting coupled model is known as a variational autoencoder.

Gradient based inference and learning

Both SVI and AVI require us to have a way to derive gradients with respect to the variational parameters and the model parameters. To solve the optimization problem $\max_\theta \max_\phi \mathcal{L}(x, q(z|x;\phi);\theta)$ via gradient ascent, we need access to $\nabla_\phi \mathcal{L}(x, q(z|x;\phi);\theta)$ and $\nabla_\theta \mathcal{L}(x, q(z|x;\phi);\theta)$. We may obtain gradients with respect to the model's parameter as follows:

$$\begin{aligned}\nabla_\theta \mathcal{L}(x, q(z|x;\phi);\theta) &= \nabla_\theta \mathbb{E}_{q(z|x;\phi)}[\log p(x,z;\theta)] + \mathrm{H}(q(z|x;\phi)) \\ &= \mathbb{E}_{q(z|x;\phi)}[\nabla_\theta \log p(x,z;\theta)] + \underbrace{\mathrm{H}(q(z|x;\phi))}_{\text{constant w.r.t } \theta}\end{aligned} \tag{2.8}$$

As long as we can evaluate gradients of the log-joint probability $\log p(x,z;\theta)$, we can obtain an unbiased estimate of $\nabla_\theta \mathcal{L}(x, q(z|x;\phi);\theta)$ via Monte-Carlo sampling from Equation 2.8.

Obtaining $\nabla_\phi \mathcal{L}(x, q(z|x;\phi);\theta)$ is a little harder. For example, the efficient computation of this gradient may depend on distributional assumptions made about the conditional probabilities in the generative model; e.g. Hoffman *et al.* (2013) assume the complete conditionals lie within the exponential family; consequently they can derive an analytic form for the ELBO as a function of the variational parameters. For variational inference in deep generative models, however, deriving gradients with respect to ϕ is harder. Unlike in Equation 2.8, we may not, as easily, obtain an unbiased estimate of the gradient via a Monte-Carlo approximation of an expectation since bringing the gradient operator inside the expectation leaves us with an integral over the gradient of a product of distributions.

To circumvent this issue, (Kingma & Welling, 2014; Rezende *et al.* , 2014) make use of the *reparameterization trick*:

$$\mathbb{E}_{z\sim\mathcal{N}(\mu,\Sigma)}[f(z)] = \mathbb{E}_{\epsilon\sim\mathcal{N}(0;\mathbb{I})}[f(\mu + R\epsilon)]; \quad \Sigma = RR^T \tag{2.9}$$

Crucially, the use of the trick *removes* the dependence of the expectation on the parameters ϕ.

$$\begin{aligned}\nabla_\phi\mathcal{L}(x, q(z|x;\phi);\theta) &= \nabla_\phi\mathbb{E}_{z\sim q(z|x;\phi)}[\log p(x,z;\theta)] + \nabla_\phi\mathrm{H}(q(z|x;\phi))\\ &= \nabla_\phi\mathbb{E}_{z\sim\mathcal{N}(\mu(x;\phi),R(x;\phi)R^T(x;\phi))}[\log p(x,z;\theta)] + \nabla_\phi\mathrm{H}(q(z|x;\phi))\\ &= \nabla_\phi\mathbb{E}_{\epsilon\sim\mathcal{N}(0;\mathbb{I})}[\log p(x,\mu(x;\phi) + R(x;\phi)\epsilon;\theta)] + \nabla_\phi\mathrm{H}(q(z|x;\phi))\\ &= \mathbb{E}_{\epsilon\sim\mathcal{N}(0;\mathbb{I})}[\nabla_\phi \log p(x,\mu(x;\phi) + R(x;\phi)\epsilon;\theta)] + \nabla_\phi\mathrm{H}(q(z|x;\phi))\end{aligned} \tag{2.10}$$

Where Equation 2.10 now gives us a route to approximate the gradients with respect to ϕ via Monte-Carlo sampling.

The reparameterization trick is not the only route to obtain unbiased estimates, and there is a rich literature surrounding the construction of schemes that permit the approximation of gradients via Monte-Carlo approximations. When the variational distribution is not continuous and/or reparameterizable, one may use the (more general purpose) score function estimator (Ranganath *et al.* , 2013; Mnih & Gregor, 2014) to obtain gradients. Maddison *et al.* (2016) derive a new family of reparameterizable distributions, known as the Concrete distribution, that allows for the practitioner to form variational approximations to discrete latent variables. Jankowiak & Obermeyer (2018) derive connections between the reparameterization gradients and solutions of a transport equation (as in optimal transport). Lee *et al.* (2018) derive gradients when the variational distribution used is non-differentiable. Finally, we refer the reader to Mohamed *et al.* (2019) for an overview of a variety of techniques to obtain unbiased gradients using Monte-Carlo approximations as well as their use in several sub-domains of machine learning, with variational inference, being one among them.

Overview

We instantiated variational inference as a method for probabilistic inference used within parameter estimation and highlighted Stochastic Variational Inference (SVI) and Amortized Variational Inference. In reality, both are a small part of the rich literature in variational methods for probabilistic inference and we refer the reader to (Jordan *et al.* , 1999) for a comprehensive overview of the same. Variational inference is not the only technique capable of approximating the posterior distribution. Techniques such as Markov Chain Monte Carlo are capable of drawing samples from the true posterior distribution in complex latent variable models; however, this remains outside the scope of this thesis and we refer the reader to (Neal, 1993) for a broad overview of MCMC methods for probabilistic inference in graphical models.

2.4 Learning with automatic differentiation

Until now, we have assumed access to the gradients of functions of the log-joint of the data and latent variables. But when the conditional probability distributions are parameterized by neural networks, these gradients may be complex, vector valued functions of their inputs. Manually deriving these gradients can prove tedious and can severely hinder a practitioner's ability to experiment with different choices of parameterizations for conditional probabilities in the model. Fortunately, much of the machinery behind the computation of gradients for such functions may be automated by *Automatic Differentiation* (AD).

The remarkable successes that deep learning has seen would be all but a pipe dream without automatic differentiation. Academic software such as Theano (Team *et al.* , 2016) and Torch (Collobert *et al.* , 2002) led the way for industrial scale frameworks such as PyTorch (Paszke *et al.* , 2017) and Tensorflow (Abadi *et al.* , 2015) which allow for deep learning algorithms to scale via the distribution of computation across multiple hardware devices.

At its core, AD allows users to specify a mathematical function in code. The code is silently instrumented so that when it is executed, intermediate results about the execution of the function are stored in computer memory. The calculation of derivatives then proceeds via the chain rule. A thorough review of AD is unnecessary to understand the context of this thesis and therefore out of scope. However, we point the reader to (Baydin *et al.* , 2017) for an accessible survey on the topic.

2.5 Modeling data with deep generative models

Despite the advances that machine learning has made over the decades, there is much we cannot say with certainty.

Inference or parameter estimation? A note on being Bayesian

The background we present on parameter estimation in latent variable models treat latent variables as first-class citizens whose values we must infer from data. A valid question then is: why not treat the model parameters θ as random variables too? Certainly, much of the machinery discussed in the previous sections carries forward and we can use techniques such as Markov Chain Monte Carlo and Variational Inference to pose parameter estimation as probabilistic inference. This is a valid point of view and for certain problems a desirable one, but one that this thesis does not explore in depth. We refer the reader to (MacKay & Mac Kay, 2003) for an overview of machine learning from a Bayesian lens.

What makes a good model?

One of the most important question that a modeller must answer is, how to design a probabilistic graphical model. Certainly a good starting point would be to collect all the random variables in the problem at hand – these form the nodes in the graph. Next might be engaging with a domain expert to understand which random variables are associated with each other, and what drives their association. These form the edges in the graph. However, at this junction, several important questions remain.

Are there latent variables in the problem to consider and account for? How do the latent variables relate to the observed data, is the interaction linear or non-linear, known or unknown? How can we ever know that we've gotten the right model?

There are no *right* answers to these questions. George Box, a famous statistician is credited with the saying: "*all models are wrong, but some models are useful*". In reality, model development is an iterative process. Machine learning's successes have been as much about using data judiciously as they have been about taking insights from domain experts and operationalizing them with mathematical primitives and incorporating them within graphical models.

Chapter 3

Gradient based introspection in deep generative models

Factor analysis (Spearman, 1904a) is a widely used model in the applied sciences. The generative model assumes the data is a linear function of independent latent variables. In Section 2.3.1 we introduced nonlinear factor analysis where the generative model is parameterized by deep neural networks. When paired with an inference or recognition network (Hinton *et al.* , 1995), a parametric function that predicts local variational parameters from data, such models go by the name of variational autoencoders (VAEs, Kingma *et al.* , 2014; Rezende *et al.* , 2014).

One of the reasons that factor analysis has found widespread use is that studying the matrix which maps from the latent variable onto the data characterizes the correlations that exist among features. In this chapter, we discuss ways in which we may similarly introspect into deep generative models through the use of gradient operators.

3.1 Introduction

The promise of variational autoencoders, and deep generative models, lies in the ability to model complex nonlinear data. However it is worth pausing to ask: what purpose is served by fitting more powerful generative models? Breiman (2001) argues that discriminative modeling falls into two schools of thought: the data modeling and the algorithmic modeling culture. The former advocates the use of models with interpretable, mechanistic processes while the latter espouses black box techniques with an emphasis on prediction accuracy. Breiman's arguments also apply to the divide between deep generative models with complex conditional distributions and simpler, more interpretable statistical models. Our goal in this chapter will be to bridge this divide.

Consider a classic model such as latent Dirichlet allocation (LDA) (Blei *et al.* , 2003). It is outperformed in held-out likelihood (Miao *et al.* , 2016) by deeper generative models and assumes a simple probabilistic process for data generation that is unlikely to hold in reality. Despite its simplicity, its generative semantics lend it a distinct advantage: interpretability. The word-topic matrix in the LDA allows practitioners to read off *what* the model has learned about the data. This begs the question we explore herein: is there a natural way to interpret the generative model when the conditional distributions are parameterized by a deep neural network?

Using unsupervised models of data, there are two kinds of insights one can hope to achieve:

1. **Interpreting datapoints:** Latent variable models encode our knowledge of the data generating process within the prior distributions over latent variables. By studying variation and patterns in the inferred values of (often low-dimensional) latent variables across a dataset, we may more easily observe salient structure that exists within a dataset. For example, using T-SNE (Maaten & Hinton, 2008) on the inferred latent representations from a variational autoencoder trained on MNIST (LeCun, 1998) reveals that there are ten clusters, each one corresponding to one of ten digits within the dataset.

2. **Interpreting model parameters:** Whereas the latent variables may be used to simplify patterns that differentiate datapoints in a dataset, the parameters within latent variable models can also yield insights into patterns that unify data by their inclusion in a dataset. For example, a common use of factor analysis is feature exploration. After learning the generative model in Equation 2.1, we may inspect the factor loading matrix W to study patterns by which features of the observation vary according to a latent dimension. By plotting these *feature representations* in latent space we may inspect and discover ambient structure among features that are shared across all datapoints. This is the premise underpinning the literature of exploratory factor analysis (Norris & Lecavalier, 2010) and the form of interpretability that we study in this chapter.

Our contribution towards this vision is a simple, easy to implement method to interpret the parameters learned by nonlinear factor analysis. We use the Jacobian of the conditional distribution with respect to latent variables to create embeddings (or Jacobian vectors) of the observational features. Intuitively the Jacobian captures variation for each feature of the observations along directions in the vector dimensional latent space.

Introspection into what the model has learned through the use of embeddings has a long history. Landauer *et al.* (1998) proposed latent semantic analysis, one of the earliest works to create vector space representations of documents. Bengio *et al.* (2003) and Mikolov *et al.* (2013a) propose log-linear models to create word-representations from document corpora. Rudolph *et al.* (2016) describe a family of models to create conditional embeddings where the data are parameterized to lie within the exponential family. In the context of discriminative modeling, Erhan *et al.* (2009) use gradient information to study the patterns with which neurons are activated in a deep neural network. At the crux of each of the above methods lies the ethos that the inspection of a model's parameters via embeddings can yield insights into what the model has learned about the dataset on hand.

We make use of gradients in the log densities of a statistical model relative to per-datapoint latent variables to capture patterns of interest to practitioners. However, gradient information may also be of use to other downstream models. For example, the derivative of the log-probability of data with respect to the globally shared parameters of a generative model encodes the variability in the input under the generative process. Jaakkola & Haussler (2007) exploit this variability to form a kernel function for a discriminative classifier using Fisher Score features.[1]

Generative model: In this chapter, we will consider a generative model of the form shown in Figure 2-7. We observe a set of D word-count[2] vectors $x_{1:D}$, where x_{dv} denotes the number of times that word index $v \in \{1, \ldots, V\}$ appears in document d. We assume we are given the total number of words per document $N_d \equiv \sum_v x_{dv}$, and that x_d was generated via the following generative process:

$$\begin{aligned} & z_d \sim \mathcal{N}(0, I);\ \gamma(z_d) \equiv \text{MLP}(z_d; \theta); \\ & \mu(z_d) \equiv \frac{\exp\{\gamma(z_d)\}}{\sum_v \exp\{\gamma(z_d)_v\}};\ x_d \sim \text{Mult.}(\mu(z_d), N_d). \end{aligned} \tag{3.1}$$

[1]For some model that parameterizes $p(x;\theta)$, the Fisher score is defined as $U_x = \nabla_\theta \log p(x;\theta)$

[2]We use word-count in document for the sake of concreteness. Our methodology is generally applicable to other types of discrete high-dimensional data.

That is, we draw a Gaussian random vector, pass it through a multilayer perceptron (MLP) with parameters θ, pass the resulting vector through the softmax (a.k.a. multinomial logistic) function, and sample N_d times from the resulting multinomial distribution over the vocabulary. In keeping with common practice, we neglect the multinomial base measure term $\frac{N!}{x_1!\cdots x_V!}$, which amounts to assuming that the words are observed in a particular order.

3.2 Jacobian vectors

Linear models are inherently interpretable. Consider linear regression, factor analysis (Spearman, 1904a), and latent Dirichlet allocation (LDA; Blei *et al.* , 2003), which (standardizing notation) assume the following relationships:

$$\begin{aligned}
\text{Regression: } \mathbb{E}[y|x] &= Wx + b; \\
\text{Factor Analysis: } x \sim \mathcal{N}(0, I); \quad \mathbb{E}[y|x] &= Wx + b; \\
\text{LDA: } x \sim \text{Dirichlet}(\alpha); \quad \mathbb{E}[y|x] &= Wx.
\end{aligned} \tag{3.2}$$

In each case, we need only inspect the parameter matrix W to answer the question "what happens to y if we increase x_k a little?" The answer is clear—y moves in the direction of the kth row of W. We can ask this question differently and get the same answer: "what is the derivative $\frac{\partial \mathbb{E}[y|x]}{\partial x}$?" The answer is simply the parameter matrix W.

For latent variable models like nonlinear factor analysis (NFA), the variability in the training data is assumed to be due to the single latent state z. The relationship between latent variables z and observations x cannot be quickly read off the parameters θ. But we can still ask what happens if we perturb z by some small dz—this is simply the directional derivative $\frac{\partial \mathbb{E}[x|z]}{\partial z} dz$. We can interpret this Jacobian matrix in much the same way we would a factor loading matrix, with two main differences.

1. The Jacobian matrix $\frac{\partial \mathbb{E}[x|z]}{\partial z}$ varies with z—the interpretation of z may change significantly depending on context.

2. NFAs exhibit rotational symmetry—the prior on z is rotationally symmetric, and the MLP can apply arbitrary rotations to z before applying any nonlinearities, so a priori there is no "natural" set of basis vectors for z. For a given Jacobian

matrix, however, we can find the most significant directions via a singular value decomposition (SVD).

For the generative model described in Eq. 3.1, we consider three variants of Jacobian embedding vectors based on the unnormalized potentials from the MLP, logarithmic probabilities, and linear probabilities respectively:

$$\mathcal{J}(z)^{\text{pot}} = \frac{\partial \gamma(z)}{\partial z}; \mathcal{J}(z)^{\text{log}} = \frac{\partial \log \mu(z)}{\partial z}; \mathcal{J}(z)^{\text{prob}} = \frac{\partial \mu(z)}{\partial z} \tag{3.3}$$

For any z, $\{\mathcal{J}(z)^{\text{log}}, \mathcal{J}(z)^{\text{pot}}, \mathcal{J}(z)^{\text{prob}}\} \in \mathbb{R}^{V\times K}$ where K is the latent dimension and V is the dimensionality of the observations. We use this matrix to form embeddings.

When not referring to a particular variant, we use $\mathcal{J}(z)$ to denote the Jacobian matrix. $\mathcal{J}(z)$ is a function of z leaving open the choice of where to evaluate this function. The semantics of our generative model suggest a natural choice: $\mathcal{J}_{\textbf{mean}} := \mathbb{E}_{p(z)}[\mathcal{J}(z)]$. This set of embeddings captures the variation in the output distribution with respect to the latent state across the prior distribution of the generative model. One may also evaluate the Jacobian at the approximate posterior corresponding to an observation x. In Section 3.3, we show how this may be used to obtain contextual feature vectors. In automatic-differentiation frameworks (Theano Development Team, 2016; Paszke *et al.* , 2019), $\mathcal{J}_{\textbf{mean}}$ is easily estimated via Monte Carlo sampling from the prior.

For the choice of likelihood (i.e., multinomial) of the data, we depict the functional form of the Jacobian vectors for linear and nonlinear factor analysis in Table 3.1. In linear models $\gamma(z_d) = Wz_d$ (c.f Eq 3.1) and in nonlinear models $\gamma(z_d) = f(z_d; \theta)$ for some smooth, differentiable function f. We denote by $\nu(z)_i = \nabla_z \gamma_i(z)$, the ith row of the matrix $\nabla_z \gamma(z) \in \mathbb{R}^{V\times K}$.

The three kinds of Jacobian vectors realize different ways to form low-dimensional representations of features. At the core of each is $\nu(z)_i$, the gradient of the unnormalized potential with respect to the latent state. $\mathcal{J}(z)^{\text{pot}}$ uses $\nu(z)_i$ directly as an embedding for each feature while $\mathcal{J}(z)^{\text{log}}$ uniquely represents a feature within the convex hull of pairwise differences between $\nu(z)_i$ and $\nu(z)_j$ for all other features in the vocabulary j. Which of the three is most sensible and works best may depend on the choice of parameterization for the conditional probability $p_\theta(x|z)$. Eq. 3.2 presents examples where $\mathcal{J}(z)^{\text{prob}}$ is a sensible choice and $\mathcal{J}(z)_i^{\text{pot}}$ in a linear model recovers the practice (Mikolov *et al.* , 2013a) of using the final weight matrix as embeddings for features. To the best of our knowledge, Jacobian Vectors and their properties have not been studied in the context of deep generative models.

Table 3.1: **Jacobian vectors:** The functional form of the Jacobian vectors for feature i as defined in Eq. 3.3 when $p(x_i = 1|z)$ is defined as in Eq. 3.1.

	Linear Model
$\mathcal{J}(z)_i^{\text{log}}$	$\sum_j p(x_j = 1\|z)(w_i - w_j)$
$\mathcal{J}(z)_i^{\text{prob}}$	$p(x_i = 1\|z) \sum_j p(x_j = 1\|z)(w_i - w_j)$
$\mathcal{J}(z)_i^{\text{pot}}$	w_i
	Nonlinear Model
$\mathcal{J}(z)_i^{\text{log}}$	$\sum_j p(x_j = 1\|z)(\nabla_z \nu(z)_i - \nu(z)_j)$
$\mathcal{J}(z)_i^{\text{prob}}$	$p(x_i = 1\|z) \sum_j p(x_j = 1\|z)(\nu(z)_i - \nu(z)_j)$
$\mathcal{J}(z)_i^{\text{pot}}$	$\nabla_z \nu(z)_i$

Deriving Jacobian vectors Here, we derive the function form of Jacobian vectors. For clarity we provide the generative model under consideration:

$$z_d \sim \mathcal{N}(0, I);\ \gamma(z_d) \equiv \text{MLP}(z_d; \theta);$$
$$\mu(z_d) \equiv \frac{\exp\{\gamma(z_d)\}}{\sum_v \exp\{\gamma(z_d)_v\}};\ x_d \sim \text{Multinomial}(\mu(z_d), N_d). \tag{3.4}$$

For simplicity, we derive the functional form of the Jacobian in a linear model, i.e., where $\gamma(z_d) = W z_d$ (c.f Eq 3.4). We drop the subscript d and denote by $\gamma_i(z)$, the ith element of the vector $\gamma(z)$. Then, we can write the probability of an element as:

$$p(x_i = 1|z) = \frac{\exp(\gamma_i(z))}{\sum_j \exp(\gamma_j(z))} \qquad \text{and} \quad \gamma_i(z) = w_i^T z$$

For linear models, $\nabla_z \gamma_i(z) = w_i$ directly corresponds to $\mathcal{J}(z)^{\text{pot}}$. Noting that $\nabla_z \exp(\gamma_i(z)) = \exp(\gamma_i(z)) \nabla_z \gamma_i(z)$ and $\nabla_z \sum_j \exp(\gamma_j(z)) = \sum_j \exp(\gamma_j(z)) \nabla_z \gamma_j(z)$, we estimate $\mathcal{J}(z)^{\text{prob}}$ as:

$$
\begin{aligned}
\nabla_z p(x_i = 1|z) &= \nabla_z \frac{\exp(\gamma_i(z))}{\sum_j \exp(\gamma_j(z))} \\
&= \frac{\sum_j \exp(\gamma_j(z)) \nabla_z \exp(\gamma_i(z)) - \exp(\gamma_i(z)) \nabla_z \sum_j \exp(\gamma_j(z))}{(\sum_j \exp(\gamma_j(z)))^2} \\
&= \frac{\sum_j \exp(\gamma_j(z)) \exp(\gamma_i(z)) w_i - \exp(\gamma_i(z)) \sum_j \exp(\gamma_j(z)) w_j}{(\sum_j \exp(\gamma_j(z)))^2} \\
&= p(x_i = 1|z) w_i - p(x_i = 1|z) \sum_j p(x_j = 1|z) w_j \\
&= p(x_i = 1|z)(w_i - \sum_j p(x_j = 1|z) w_j)
\end{aligned}
$$

Similarly, we may compute $\mathcal{J}(z)^{\text{log}}$:

$$
\nabla_z \log p(x_i = 1|z) = w_i - \sum_j p(x_j = 1|z) w_j = \sum_j p(x_j = 1|z)(w_i - w_j) \tag{3.5}
$$

We use $w_i - w_j$ to denote a word-pair vector, where w_i, w_j are columns of the matrix W. If we define the set of all word-pair vectors as $\mathcal{S}$, then Eq 3.5 captures the idea that the vector representation for a word i lies in the convex hull of $\mathcal{S}$. Furthermore, the word vector's location in CONV($\mathcal{S}$) is determined by the likelihood of the pairing word (x_j) under the model $p(x_j = 1|z)$. When we use a non-linear conditional probability distribution $\mathcal{J}(z)^{\text{log}}$ becomes: $\nabla_z \log p(x_i = 1|z) = \sum_j p(x_j = 1|z)(\nabla_z \gamma_i(z) - \nabla_z \gamma_j(z))$ where $\nabla_z \gamma_i(z)$ is a non-linear function of z.

3.3 Evaluation

We study the various properties of $\mathcal{J}^{\text{log}}_{\textbf{mean}}$ (unless otherwise specified) derived from a nonlinear factor analysis model trained on diverse sets of data with different kinds of structure. We form a Monte Carlo estimate of $\mathcal{J}^{\text{log}}_{\textbf{mean}}$ using 400 samples. Cosine distance is used to define neighbors of words in the embedding space of the Jacobian. We evaluate embeddings qualitatively and quantitatively.

3.3.1 Text data

There is much prior work in the construction of embeddings in Natural Language Processing (Almeida & Xexéo, 2019). Many techniques such as GLoVE embeddings (Mikolov *et al.* , 2013a; Pennington *et al.* , 2014) make use of structure within sentences to learn models whose parameters give rise to the embeddings. For example, Word2Vec (Mikolov *et al.* , 2013a) builds a predictive model of the next word given its surrounding context. This implicitly makes use of the fact that semantically correlated words often arise within the same context and this structure makes its way into the model's parameters from which embeddings are extracted. An important caveat to our results herein is that the use of a latent variable model is *sub-optimal*, in the sense that it does not make use of the structure that correlated words co-occur close to each other in sentences. For any given document, the model assumes that all words are conditionally independent given the latent variable. We do not anticipate that the embeddings for words thus obtained outperform those from Word2Vec; rather our interest lies in examining their relative merits on a variety of tasks.

Dataset: We train nonlinear factor analysis models on the large Wikipedia corpus used in Huang *et al.* (2012). The resulting dataset is of size train/valid/test: 1,212,781/2,000/10,000 and vocabulary V:20,253.

Table 3.2: **Word embeddings (nearest neighbors):** We visualize nearest neighbors of word embeddings (excluding plurals of the query)

Query	**Neighborhood**
intelligence	espionage, colleagues, cia
zen	dharma, buddhism, buddha, meditation
book	author, republished, written, paperback
medicine	physicians, medical, pathology, vascular

Preprocessing: We strip all special characters, remove hyphens between words, ignore numbers and include (a) the top 20000 words in the vocabulary and (b) the words needed to complete the evaluation for the Stanford Contextual Word Similarity (SWCS) (Huang *et al.* , 2012) and WordSim353 (Finkelstein *et al.* , 2001) benchmarks. This results in a total vocabulary size of 20253. When performing the evaluation for contextual word embeddings in Table 3.3, we use Wikipedia to obtain our context document, which we perform inference with. The context document comprises bag of words from the first 5000 characters of the word's Wikipedia page. For example, if the context word is "construction", then the URL to extract text from Wikipedia would be `https://en.wikipedia.org/w/api.php?format=json&action=`

`query&prop=extracts&exchars=5000&explaintext=&titles=construction`. We build a bag-of-words representation from the text thus extracted and the vocabulary of our dataset.

Evaluation: For text data, direct quantitative comparison is difficult since (1) our vocabulary is on the order of tens of thousands of words compared to most work (Pennington *et al.* , 2014) where V ranges from 0.4 to 2 million and (2) many of the models we compare to use local context during learning, which yields a more precise signal about the meanings of particular words. Nonetheless, we study where Jacobian vectors stand (albeit with a significantly smaller vocabulary and a global training objective). Using an inference network we train (1) 1-L multinomial PCA (Collins *et al.* , 2001), corresponding to a single linear layer in $p(x|z;\theta)$ and (2) 3-L a deep generative model with a three layer neural network that parameterizes $p(x|z;\theta)$.

Table 3.3: **Word embeddings (polysemy):** We visualize the nearest neighbors under the Jacobian vector induced by the posterior distribution of a document created based on the context word.

Word	**Context**	**Neighboring words**
crane	construction	lifting, usaaf, spanned, crushed, lift
	bird	erected, parkland, locally, farmland, causeway
bank	river	watershed, footpath, confluence, drains, tributary
	money	banking, government, bankers, comptroller, fiscal

Table 3.4: **Semantic similarity on text data:** A higher number is better. In Table 3.4a, 3.4b, the baseline results are taken from Huang *et al.* (2012). C&W uses embeddings from the language model of Collobert & Weston (2008). Glove corresponds to embeddings by Pennington *et al.* (2014). ρ corresponds to Spearman rho-correlation.

(a) **WordSim353**

Models	$\rho \times 100$
Huang	71.3
Glove	75.9
C&W	55.3
ESA	75
Huang (G)	22.8
1-L $\mathcal{J}^{\text{prob}}_{\textbf{mean}}$	69.7
3-L $\mathcal{J}^{\text{prob}}_{\textbf{mean}}$	59.6

(b) **SCWS**

Models	$\rho \times 100$
Huang	65.7
C&W	57
tf-idf-S	26.3
Pruned tf-idf-S	62.5
1-L $\mathcal{J}^{\text{prob}}_{\textbf{mean}}$	61.7
3-L $\mathcal{J}^{\text{prob}}_{\textbf{mean}}$	59.5

Qualitative analysis In Table 3.2, we visualize some of the nearest neighbors of words using $\mathcal{J}^{\text{log}}_{\textbf{mean}}$ obtained from models trained on Wikipedia and find that the neighbors are semantically sensible. Next, we consider contextual embeddings. Rather than evaluating the Jacobian at L points $z_{1:L} \sim p(z)$, we instead evaluate it at $z_{1:L} \sim q(z|x)$ for some x. In Table 3.3, we select three polysemous query words alongside "context words" that disambiguate the query's meaning. For each word-context pair, we create a document comprising a subset of words in the the context's Wikipedia page. Then, we use the inference network to perform posterior inference to evaluate $\mathcal{J}^{\text{log}}_{\textbf{mean}}$ at the corresponding $q(z|x)$. This yields a set of contextual Jacobian vectors. We display the nearest neighbors for each word under different contextual Jacobian vectors and find that, while not always perfect, they capture *different* contextually relevant semantics. Note that other approaches to obtain context specific representations (Chen *et al.* , 2014) explicitly use local context during training – our method does not. Rather the contextual nature of the representation arises due to the sensitivity of the nonlinear Jacobian vector to the choice of z. By combining posterior inference in NFA with our methodology of introspecting the model, one obtains *different* context-specific representations.

Quantitative analysis To quantify the amount of semantic content in Jacobian vectors, we evaluate the vector space representations on WordSim353 (Finkelstein *et al.* , 2001) and SCWS (Huang *et al.* , 2012). Each benchmark contains human annotated measures of similarity between words. The evaluation on the WordSim and SCWS datasets are done by computing the Spearman rank correlation between human annotated rankings between 1 and 10 and an algorithmically derived measures of word-pair similarity. We first compute the distances between all word pairs. Our measure of similarity is obtained by subtracting the distances from the maximal distance across all word pairs. Closest to us in learning procedure is (Huang (G), Table 3.4a), whose model we outperform. On a discriminative task of predicting sentiment on the Stanford Sentiment Treebank Dataset (Socher *et al.* , 2013), we find that Jacobian vectors perform only slightly worse than Glove embeddings, despite being trained with a much smaller vocabulary. For this task, we do not find much improvements in the quality of the embeddings relative to those obtained from a simpler multinomial-PCA model.

To test the discriminative ability of the learned Jacobian vectors, we use the Jacobian vectors as representations for sentiment classification. We evaluate our method on classifying the sentiment of movie ratings from the Rotten Tomatoes (RT) dataset

Table 3.5: **Discriminative ability of Jacobian vectors:** Glove corresponds to embeddings by (Pennington *et al.* , 2014). (Stanford Sentiment Treebank) SST-fine corresponds to the fine grained classification task of predicting one of eight different sentiments while SST-binary corresponds to predicting a positive or negative sentiment for the sentence.

Models	Rotten Tomatoes	SST-fine	SST-binary
Glove	75.2	41.5	77.7
1-L $\mathcal{J}^{\text{log}}_{\textbf{mean}}$	72.6	42.6	76.4
3-L $\mathcal{J}^{\text{log}}_{\textbf{mean}}$	70.3	40.2	74.1

(Pang & Lee, 2005) and from the Stanford Sentiment Treebank (SST) Dataset (Socher *et al.* , 2013). We follow the procedure in (Iyyer *et al.* , 2015), who average word embeddings from Glove and use the resulting averaged embedding as a sentence representation. They use the resulting representation as input to an MLP to predict the sentiment of the sentence. The 300 dimensional Glove vectors are created from a bigger dataset (Common Crawl) with a much larger vocabulary (2 million). In our experiment we created 300 dimensional Jacobian vectors on a variant of the Wikipedia dataset with 40000 features. To partially even the playing field, we restrict our comparison to Glove using only the words vectors that lie in our own vocabulary. While we do not outperform Glove vectors on any of the datasets, we do perform comparably on the Stanford Sentiment Treebank datasets in both the fine grained and coarse grained prediction task.

3.3.2 Electronic Health Record (EHR) data

Next, we study Jacobian vectors deep generative models learned on electronic health record data. We construct a dataset using EHR data provided by an insurance company. There are 185,000 patients and patient's data across time was aggregated to create a bag-of-diagnosis-codes representation of the patient. The vocabulary comprises four different kinds of medical diagnosis codes: diagnosis codes, laboratory tests, prescription medication and surgical procedures. The vocabulary is of size $V = 51,321$, though for any given patient, only a small handful of the embeddings are non-zero.

Qualitative Analysis In Table 3.6, similar to the setup in Table 3.2 but using Jacobian vectors derived from models of EHR data, we visualize the nearest neighbors of different drugs to find that they capture interesting disease specific structure. Table

Table 3.6: **Medical embeddings (nearest neighbors):** Nearest neighbors of some diagnosis codes (ignoring duplicates). Metformin (and it's neighbors) are diabetic drugs. A contour meter measures blood glucose. Spiriva and it's neighbors are drugs used for treating chronic obstructive pulmonary disease (COPD).

Code	**Neighboring codes**
Metformin	Glimepiride, Avandia, Contour Meter
Spiriva	Advair, Albuterol, Foradil
Asbestosis	Coal Workers' Pneumoconiosis, Ct Scan Chest
Bone Marrow Transplant **[C]**	Acute Graft-Versus-Host Disease **[I9]**, Microscopic Examination (Bacterial Smear) **[I9 Proc]**, Bone Marrow Biopsy **[C]**

Table 3.7: **Medical analogies:** We perform analogical reasoning with embeddings of medical codes. If we know a drug used to treat a disease, we can use their relationship in vector space to find unknown drugs associated with a different disease. Queries take the form Code 1$\rightarrow$Code 2 $\Longrightarrow$ Code 3$\rightarrow$?. Sicca syndrome or Sjogren's disease is an immune disease treated with Evoxac and Methotrexate is commonly used to treat Rheumatoid Arthiritis. "Leg Varicosity" denotes the presence of swollen veins under the skin. "Ligation of angioaccess arteriovenous fistula" denotes the tying of a passage between an artery and a vein.

Code 1	**Code 2**	**Code 3**	**Neighbors of Result**
Evoxac	Sicca Syndrome	Methotrexate	Rheumatoid Arthritis
Biliary Atresia	Kidney Transplant	Leg Varicosity w/ Inflammation	Ligation of angioaccess arteriovenous fistula

Table 3.8: **Medical embeddings (clustering):** We visualize some topical clusters of diagnosis codes.

Label	**Diagnosis Codes**
Thrombosis	Hx Venous Thrombosis, Compression Of Vein, Renal Vein Thrombosis
Occular Atrophy	Optic Atrophy, Retina Layer Separation, Chronic Endophthalmitis
Drug Use	Opioid Dependence, Alcohol Abuse-Continuous, Hallucinogen Dep

3.7 depicts two examples of using the learned embeddings in the Jacobian matrix to answer tasks queries related to drug-disease pairs. Table 3.8 depicts clusters found in medical diagnosis codes. In all three cases, we find that the Jacobian vectors capture the semantic structure encoded in high-dimensional representations of patient data.

For EHR data in particular, the bag-of-diagnosis-codes assumption we make is a crude one since (1) we assume the temporal nature of the patient data is irrelevant, and (2) combining patient statistics over time renders it difficult for the generative model to disambiguate the correlations between codes that correspond to multiple diseases a patient may suffer from. Despite this, it is interesting that the Jacobian vectors still capture much of the meaningful structure among the diagnosis codes.

Quantitative Analysis Choi *et al.* (2016c) explore different metrics to test whether the embedding space of medical diagnosis codes captures medically related concepts well. We evaluate medical embeddings as follows. $\text{MRM}_{\text{NDF-RT}}$ (Medical Relatedness Measure under NDF-RT) leverages a medical database (NDF-RT) to evaluate how good an embedding space is at answering analogical queries between drugs and diseases. The evaluation (MRM_{CCS}) measures if the neighborhood of the diagnosis codes is medically coherent using a predefined medical ontology (CCS) as ground truth. The number is a measure of precision, where higher is better.

$\mathbf{MRM_{CCS}}(V, G)$: The Agency for Healthcare Research and Quality's clinical classification software (CCS) collapses the hierarchical ICD9 diagnosis codes into clinically meaningful categories. The evaluation on CCS checks whether the nearest neighbors of a disease include other diseases related to it (if they are in the same category in the CCS). Using the ICD9 hierarchy, the authors further split the evaluation task into predicting neighbors of fine-grained and coarse grained diagnosis codes. For a choice of granularity $G \in \{\text{fine,coarse}\}$, $V(G) \in V$ denotes the subset of ICD9 codes in the vocabulary. $\mathbb{I}_G(v(i))$ is one if the v's i'th nearest neighbor: $v(i)$ is in the same group as v according to G.

$$\text{MRM}_{\text{CCS}}(V, G) = \frac{1}{|V(G)|} \sum_{v \in V(G)} \sum_{k=1}^{40} \frac{\mathbb{I}_G(v(i))}{\log_2(i+1)} \tag{3.6}$$

$\mathbf{MRM_{NDF\text{-}RT}}(V, R)$: This uses the National Drug File Reference Terminology (NDF-RT) to evaluate analogical reasoning. The NDF-RT provides two kinds of relationships (R) between drugs and diseases: May-Treat (if the drug may be used to treat the disease) and May-Prevent. Given ϕ_A as the embedding for a code A, this test automates the evaluation of analogies such as $\underbrace{\phi_{\text{Diabetes}}}_{r} \approx \underbrace{\phi_{\text{Metformin}}}_{v} - \underbrace{(\phi_{\text{Lung Cancer}} - \phi_{\text{Tarceva}})}_{s}$. Here v is the query code and s is a representation of the relationship we seek. (Metformin is a diabetic drug and Tarceva is used in the treatment of lung cancer.) The evaluation we perform reports a number proportional to the number of times the neighborhood of $v - s$ contains r for the best value of s (computed from the set of all valid drug-disease relationships in the datasets.) Given $V^* \in V$ (concepts for which NDF-RT has at-least one substance with the given relation), $\mathbb{I}_R\left(\cup_{i=1}^{40}(v-s)(i)\right) = 1$ if any of the medical concepts in the top-40 neighborhood of the medical concept v satisfies relation R.

$$\text{MRM}_{\text{NDF-RT}}(V, R) = \frac{1}{|V^*|} \sum_{v \in V^*} \mathbb{I}_R\left(\cup_{i=1}^{40}(v-s)(i)\right) \tag{3.7}$$

In both cases the choice of 40 (in Eq. 3.7 and 3.6) was adopted to maintain consistency with (Choi *et al.* , 2016c). Both evaluations are conducted by taking the average result over all possible seeds s and the best possible seed s for a query.

Table 3.9: **Medical embeddings: Medical Relatedness Measure (MRM)** We evaluating embeddings using medical (NDF-RT and CCS) ontologies. SCUIs result from the method developed by Choi *et al.* (2016c) applied to data in Finlayson *et al.* (2014).

Models	$\textbf{MRM}_{\textbf{NDF-RT}}$	$\textbf{MRM}_{\textbf{CCS}}$
De Vine *et al.*	53.21	22.63
Choi *et al.*	59.40	44.80
SCUI	52.75	34.16
1L $\mathcal{J}_{\textbf{mean}}^{\text{pot}}$	59.63	31.58
3L $\mathcal{J}_{\textbf{mean}}^{\text{pot}}$	60.32	37.77

Table 3.9 displays the results. It is interesting that the approach we present herein outperforms baselines published in the literature even though our training procedure ignores the longitudinal aspect of EHR data (variants of Word2Vec adapted to diagnosis codes). Furthermore, we see an instance where Jacobian vectors resulting from a deeper, better-trained model outperform those from a shallow model – highlighting the importance of building nonlinear representations. We hypothesize that nonlinearity helps in representations of EHR data due to the hierarchical structure present in medical diagnosis codes (Slee, 1978).

3.3.3 Netflix: Embeddings for movies

We study the use of NFA on data from Netflix[3]. Following standard procedure: (1) we binarize the explicit rating data keeping ratings of four or higher and interpret them as implicit feedback (Hu *et al.* , 2008) and (2) we only keep users who have positively rated at least five movies. We train with users' binary implicit feedback as x_d and the

[3]`http://www.netflixprize.com/`

vocabulary comprises the set of all movies. The number of training/validation/test users is 383,435/40,000/40,000 for Netflix (V: 17,769).

The Netflix dataset comprises movie ratings of $500,000$ users. We treat each user's ratings as a document and model the numbers ascribed to each movie (from $1-5$) as counts drawn from the multinomial distribution parameterized as in Eq. 3.1. We train the three-layer deep generative model on the dataset, evaluate $\mathcal{J}_{\textbf{mean}}$ with 100 samples and consider two distinct methods of evaluating the learned embeddings. We cluster the movie embeddings (using spectral clustering with cosine distance to obtain 100 clusters) and depict some of the clusters in Table 3.10a. We find that clusters exhibit coherent themes such as documentary films, horror and James Bond movies. Other clusters (not displayed) included multiple seasons of the same show such as Friends, WWE wrestling, and Pokemon. In Table 3.10b, we visualize the neighbors of some popular films. In the examples we visualize, the nearest neighbors include sequels, movies from the same franchise or, as in the case of 12 Angry Men, other dramatic classics.

To compare the effect of using a model to create embeddings versus using the raw data from a large dataset directly, we evaluated nearest neighbors of movies using a simple baseline. For a query movie, we found all users who gave the movie a rating of 3 or above (nominally, they watched and liked the movie). Then, for all those users, we computed the mean ratings they gave to every other movie in the vocabulary and ranked them based on the mean ratings. We display the top five movies obtained using this approach in Table 3.10c. The query words are the same as in Table 3.10b. For most of the queries, the difference between the two is evident and we simply end up with *popular, well-liked* movies rather than relevant movies.

3.4 Discussion

This chapter introduced and studied Jacobian Vectors both qualitatively and quantitatively. In three different datasets of high-dimensional data, we showed how Jacobian vectors capture semantic structure among features in datasets giving practitioners a new way to perform exploratory analysis of data with deep generative models.

Beyond the construction of embeddings, there are many other uses practiotioners can find for the gradients in deep generative models. In Chapter 4, we will see how the gradient operator may be used to characterize the quality of a learned generative model and in Chapter 7, we will make use of the Jacobian to interpret what a sequential, deep generative model has learned about the data.

Table 3.10: **Qualitative evaluation of movie embeddings:** We evaluate $\mathcal{J}^{\text{log}}_{\textbf{mean}}$ using 100 Monte-Carlo samples to perform the evaluation in Tables 3.10a and 3.10b.

(a) **Clustering movie embeddings:** We display some of the clusters found from clustering the movie embeddings. The names were assigned based on salient features of movies in the cluster

Cluster Name	**Movies**
Documentary Films	Nature: Antarctica, Ken Burns' America: Empire of the Air, Travel the World by Train: Africa, Deepak Chopra: The Way of the Wizard & Alchemy, The History Channel Presents: Troy: Unearthing the Legend
Concerts	Neil Diamond: Greatest Hits Live, Meat Loaf: Bat Out of Hell, Ricky Martin: One Night Only, Beyonce: Live at Wembley, Enigma: MCMXC A.D, Sarah Brightman: In Concert
Horror Movies	Halloween 5: The Revenge of Michael Myers, Halloween: H2O, Creepshow, Children of the Corn, Poltergeist, Friday the 13th: Part 3, The Omen, Cujo
James Bond	For Your Eyes Only, Goldfinger, The Living Daylights, Thunderball, From Russia With Love, Dr. No
Hindi Movies	Seeta Aur Geeta, Gupt, Mann, Jeans, Coolie No.1, Mission Kashmir, Rangeela, Baazigar, Daud, Zakhm

(b) **Movie neighbors:** We visualize some of the closest neighbors found to movies whose title is displayed on the column on the left

Cluster Name	**Movies**
Superman II	Superman: The Movie, Superman III, Superman IV: The Quest for Peace, RoboCop, Batman Returns
Casablanca	Citizen Kane, The Treasure of the Sierra Madre, Working with Orson Welles, The Millionairess, Indiscretion of an American Wife, Doctor Zhivago
Bride of Chucky	Bride of Chucky, Leprechaun 3, Leprechaun, Wes Craven's New Nightmare, Child's Play 2: Chucky's Back
The Princess Bride	The Breakfast Club, Sixteen Candles, Groundhog Day, Beetlejuice, Stand by Me, Pretty in Pink
12 Angry Men	To Kill a Mockingbird, Rear Window, Mr. Smith Goes to Washington, Inherit the Wind, Vertigo, The Maltese Falcon

(c) **Movie neighbors [baseline]:** We visualize some of the closest neighbors to a given query movie. We using a simple baseline that rates every movie based on average scores given by all the users who liked (rating greater than three) the query movie. LOTR (Lord of the Rings), PotC (Pirates of the Caribbean)

Cluster Name	**Movies**
Superman II	LOTR: The Two Towers, PotC: The Curse of the Black Pearl, Raiders of the Lost Ark, LOTR: The Fellowship of the Ring
Casablanca	To Kill a Mockingbird, The Usual Suspects, The Shawshank Redemption, Citizen Kane, The Wizard of Oz
Bride of Chucky	The Matrix, Independence Day, The Silence of the Lambs, PotC: The Curse of the Black Pearl, The Sixth Sense
The Princess Bride	The Shawshank Redemption, Forrest Gump, LOTR: The Two Towers, LOTR: The Fellowship of the Ring, PotC: The Curse of the Black Pearl
12 Angry Men	LOTR: The Fellowship of the Ring, PotC: The Curse of the Black Pear, The Godfather, Forrest Gump, The Shawshank Redemption

Chapter 4

Representation learning for high-dimensional data

4.1 Introduction

Deep generative models, like those highlighted in Chapter 2, learn low-dimensional *representations* z, of high-dimensional random variables x from data via unsupervised learning. There are many perspectives on how, and why such models learn meaningful representations. Among them is the perspective that the task of learning a generative model is equivalent to the task of compressing high-dimensional information. Intuitively, to effectively compress high-dimensional data the model must use latent variables to capture variation in both coarse and fine-grained structure. A more formal view on this perspective can be found in (Honkela & Valpola, 2004) who characterize the relationship between the compression of information (via an information theoretic concept known as bits-back coding) and variational learning of latent variable models. However, this perspective is not a guarantee that unsupervised learning of deep generative models will always be successful in learning meaningful representations of data. In this chapter, we take a critical look at the canonical learning algorithm for deep generative models and investigate pitfalls that practioners may encounter. We focus our discussion on latent factor models.

The assumption of linearity in factor analysis (FA, Spearman, 1904b) has been relaxed in nonlinear factor analysis (NFA) (Gibson, 1960) and extended across a variety of domains such as economics (Jones, 2006), signal processing (Jutten, 2003), and

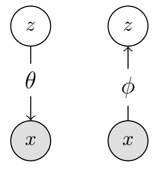
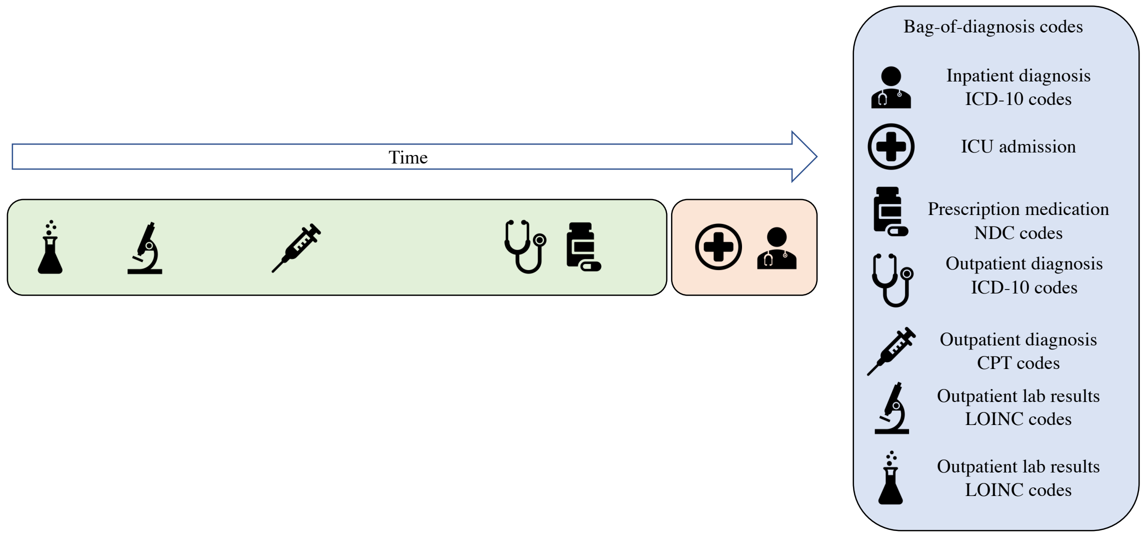


Figure 4-1: **Learning nonlinear factor analysis with an inference network: [Left]** The generative model contains a single latent variable z. The conditional probability $p(x|z;\theta)$ parameterized by a deep neural network. **[Right]** The inference network $q_\phi(z|x)$ is used for inference at train and test time.

machine learning (Valpola & Karhunen, 2002; Lawrence, 2004). NFA assumes the joint distribution factorizes as $p(x, z; \theta) = p(z)p(x|z; \theta)$ and the parameters of $p(x|z; \theta)$ in Equation 2.3 are the output of passing z through a deep neural network. Figure 4-1 depicts NFA when it is learned using an inference network (and referred to as a Variational Autoencoder (VAE)). We study VAEs for the unsupervised learning of sparse, high-dimensional categorical data.

Sparse, high-dimensional data is ubiquitous; it arises naturally in survey and demographic data, bag-of-words representations of text, mobile app usage logs, recommender systems, genomics, and finally, electronic health records. In the context of clinical data, the problem we consider in this chapter is that of learning representations of patient history as manifested in the history of diagnosis codes associated with them. Figure 4-2 depicts what this collection might look like for a single patient.



Bag-of-diagnosis codes
Inpatient diagnosis ICD-10 codes
ICU admission
Prescription medication NDC codes
Outpatient diagnosis ICD-10 codes
Outpatient diagnosis CPT codes
Outpatient lab results LOINC codes
Outpatient lab results LOINC codes
Time

Figure 4-2: **From patient history to a bag of diagnosis codes:** On the left is a depiction of a patient's history (outpatient in green and inpatient in red). On the right is how such a history would appear to machine learning models; as collections of diagnosis codes.

Why might a practitioner be interested in learning a representation of patient history? The first reason may be data analysis: a good representation may aid in finding patterns among patient cohorts that are less obvious to spot in high-dimensional data.

The second reason may be to use the representation as a proxy for the high-dimensional data in prediction tasks. High-dimensional tabular data, such as collections of patient diagnosis codes are characterized by a few frequently occurring features and a long tail of rare features. For example, during the course of a patient medical history, we may see many counts of diagnosis codes and treatments associated with common medical conditions such as hypertension but fewer codes associated with rare diseases.

When directly learning deep generative models on sparse data, a problem we run into is that the standard amortized variational learning algorithm results in *underfitting*; i.e. the learning algorithm fails to utilize the model's full capacity to model the data. This is problematic since it severely limits the applicability of this class of models to finding low-dimensional representations of sparse, high-dimensional data. To explore, understand and mitigate this phenomena, this chapter explores the following contributions to the literature (Krishnan *et al.* , 2018):

1. We identify a problem with standard VAE training when applied to sparse, high-dimensional data—underfitting. We investigate the underlying causes of this phenomenon, and propose modifications to the learning algorithm to address these causes. We combine inference networks with an iterative optimization scheme inspired by Stochastic Variational Inference (SVI) (Hoffman *et al.* , 2013).
2. We show that our proposed learning algorithm dramatically improves the quality of the estimated parameters.
3. We empirically study various factors that govern the severity of underfitting and how the techniques we propose mitigate it.
4. A practical ramification of our work is that improvements in learning NFA on recommender system data translate to more accurate predictions and better recommendations. In contrast, standard VAE training fails to outperform the simple shallow linear models that still largely dominate the collaborative filtering domain (Sedhain *et al.* , 2016).

4.2 Setup

A bag-of-words (or in the context of clinical data, a bag-of-diagnosis codes) representation is one that foregoes the ordering of observed features and instead represents collections of multivariate data as multi-sets comprising features and feature counts. For example, if the observation is a sentence: "the cat ran over the hill", then the bag-of-words representation would be $x = \{\text{the} : 2, \text{cat} : 1, \text{ran} : 1, \text{over} : 1, \text{hill} : 1\}$. Similarly for a patient's history, their bag-of-diagnosis code representation could be: $x = \{\text{ICD-10 E08.2} : 2, \text{NDC 65862-008-99} : 2, \text{LOINC 55399-0} : 3\}$. The bag-of-diagnosis codes points to a diabetic patient who had two diagnoses of diabetes mellitus (ICD-10), two prescriptions of Metformin (NDC) and three panels to track diabetes (LOINC). In what follows, we use "words" to denote the name of the item observed and "word counts" to denote their frequency. We refer to collections of words (and their counts) as documents and consider learning in generative models of the form shown in Figure 4-1. We introduce the model in the context of performing maximum likelihood estimation over a corpus of documents.

We observe a set of D word-count vectors $x_{1:D}$, where x_{dv} denotes the number of times that word index $v \in \{1, \ldots, V\}$ appears in document d. Given the total number of words per document $N_d \equiv \sum_v x_{dv}$, x_d is generated via the following generative process:

$$\begin{aligned} & z_d \sim \mathcal{N}(0, I);\ \gamma(z_d) \equiv \text{MLP}(z_d; \theta); \\ & \mu(z_d) \equiv \frac{\exp\{\gamma(z_d)\}}{\sum_v \exp\{\gamma(z_d)_v\}};\ x_d \sim \text{Mult.}(\mu(z_d), N_d). \end{aligned} \tag{4.1}$$

That is, we draw a Gaussian random vector, pass it through a multilayer perceptron (MLP) parameterized by θ, pass the resulting vector through the softmax (a.k.a. multinomial logistic) function, and sample N_d times from the resulting distribution over V.[1]

Variational Learning: For ease of exposition we drop the subscript on x_d when referring to a single data point. Jensen's inequality yields the following lower bound on the log marginal likelihood of the data:

$$\log p_\theta(x) \geq \underbrace{\mathbb{E}_{q(z;\psi)}[\log p_\theta(x \mid z)] - \text{KL}(\, q(z; \psi) \mid\mid p(z)\,)}_{\mathcal{L}(x;\theta,\psi)}. \tag{4.2}$$

$q(z; \psi)$ is a tractable "variational" distribution meant to approximate the intractable posterior distribution $p(z \mid x)$; it is controlled by some parameters ψ. For example, if

[1]In keeping with common practice, we neglect the multinomial base measure term $\frac{N!}{x_1! \cdots x_V!}$, which amounts to assuming that the words are observed in a particular order.

q is Gaussian, then we might have $\psi = \{\mu, \Sigma\}$, $q(z; \psi) = \mathcal{N}(z; \mu, \Sigma)$. We are free to choose ψ however we want, but ideally we would choose the ψ that makes the bound in equation 4.2 as tight as possible, $\psi^* \triangleq \arg\max_\psi \mathcal{L}(x; \theta, \psi)$.

Hoffman *et al.* (2013) proposed finding ψ^* using iterative optimization, starting from a random initialization. This is effective, but can be costly. More recently, Kingma & Welling (2014) and Rezende *et al.* (2014) proposed training a feedforward *inference network* (Hinton *et al.* , 1995) to find good variational parameters $\psi(x)$ for a given x, where $\psi(x)$ is the output of a neural network with parameters ϕ that are trained to maximize $\mathcal{L}(x; \theta, \psi(x))$. Often it is much cheaper to compute $\psi(x)$ than to obtain an optimal ψ^* using iterative optimization. But there is no guarantee that $\psi(x)$ produces optimal variational parameters—it may yield a much looser lower bound than ψ^* if the inference network is either not sufficiently powerful or its parameters ϕ are not well tuned.

Moving forward, we will use $\psi(x)$ to denote an inference network that implicitly depends on some parameters ϕ, and ψ^* to denote a set of variational parameters obtained by applying an iterative optimization algorithm to equation 4.2. Following common convention, we will sometimes use $q_\phi(z \mid x)$ as shorthand for $q(z; \psi(x))$.

4.3 Sources of error in variational learning

We elucidate our hypothesis on why the learning algorithm for VAEs is susceptible to underfitting. There are two sources of error in variational parameter estimation with inference networks:

1. The first is the distributional error accrued due to learning with a tractable-but-approximate family of distributions $q_\phi(z|x)$ instead of the true posterior distribution $p(z|x)$. Although difficult to compute in practice, it is easy to show that this error is exactly $\mathrm{KL}(q_\phi(z|x) \| p(z|x))$. We restrict ourselves to working with normally distributed variational approximations and do not aim to overcome this source of error.

2. The second source of error comes from the sub-optimality of the variational parameters ψ used in Eq. 4.2. We are guaranteed that $\mathcal{L}(x; \theta, \psi(x))$ is a valid lower bound on $\log p(x)$ for *any* output of $q_\phi(z|x)$ but within the same family of

variational distributions, there exists an optimal choice of variational parameters $\psi^* = \{\mu^*, \Sigma^*\}$ realizing the tightest variational bound for a data point x.

It is easy to establish the following sequence of lower bounds on the log-marginal likelihood of the data.

$$\log p(x) \geq \underbrace{\mathbb{E}_{\mathcal{N}(\mu^*;\Sigma^*)}[\log p(x|z;\theta)] - \mathrm{KL}(\mathcal{N}(\mu^*,\Sigma^*)\|p(z))}_{\mathcal{L}(x;\theta,\psi^*)} \geq \mathcal{L}(x;\theta,\psi(x)), \tag{4.3}$$

$$\text{with: } \psi^* := \{\mu^*, \Sigma^*\} = \arg\max_{\mu,\Sigma} \mathbb{E}_{\mathcal{N}(\mu,\Sigma)}[\log p(x|z;\theta)] - \mathrm{KL}(\mathcal{N}(\mu,\Sigma)\|p(z)).$$

The cartoon in Figure 4-3 illustrates this double bound.

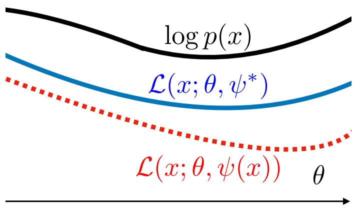

$\log p(x)$
$\mathcal{L}(x;\theta,\psi^*)$
$\mathcal{L}(x;\theta,\psi(x))$
θ

Figure 4-3: **Lower bounds in variational learning:** To estimate θ, we maximize a lower bound on $\log p(x;\theta)$. $\mathcal{L}(x;\theta,\psi(x))$ denotes the standard training objective used by VAEs. The tightness of this bound (relative to $\mathcal{L}(x;\theta,\psi^*)$ depends on the inference network. The x-axis is θ.

The canonical learning algorithm for deep generative models updates θ, ϕ jointly based on $\mathcal{L}(x;\theta,\psi(x))$. It directly uses $\psi(x)$ (as output by $q_\phi(z|x)$) to estimate Equation 4.2. See Algorithm 1 for pseudocode.

Algorithm 1 Learning with inference networks (Kingma *et al.* , 2014)

```
Inputs: $\mathcal{D} := [x_1, \ldots, x_D]$ ,
Model: $q_\phi(z|x)$, $p_\theta(x|z)$, $p(z)$;
for k = 1...K do
    Sample: $x \sim \mathcal{D}$, $\psi(x) = q_\phi(z|x)$, update $\theta, \phi$:
        $\theta^{k+1} \leftarrow \theta^k + \eta_\theta \nabla_{\theta^k} \mathcal{L}(x;\theta^k,\psi(x))$
        $\phi^{k+1} \leftarrow \phi^k + \eta_\phi \nabla_{\phi^k} \mathcal{L}(x;\theta^k,\psi(x))$
end for
```

In contrast, stochastic variational inference methods (Hoffman *et al.* , 2013) update θ based on gradients of $\mathcal{L}(x;\theta,\psi^*)$ by updating randomly initialized variational parameters for each example. ψ^* is obtained by maximizing $\mathcal{L}(x;\theta,\psi)$ with respect to ψ. This maximization is performed by M gradient ascent steps yielding $\psi_M \approx \psi^*$; see Algorithm 2 for pseudocode.

Algorithm 2 **Learning with Stochastic Variational Inference**: M: number of gradient updates to ψ.

```
**Inputs**: $\mathcal{D} := [x_1, \dots, x_D]$,
Model: $p_\theta(x|z)$, $p(z)$;
**for** k = 1...K **do**
  **1.** Sample: $x \sim \mathcal{D}$ and initialize: $\psi_0 = \mu_0, \Sigma_0$
  **2.** Approx. $\psi_M \approx \psi^* = \arg\max_\psi \mathcal{L}(x; \theta; \psi)$:
      For $m = 0, \dots, M-1$:
          $\psi_{m+1} = \psi_m + \eta_\psi \frac{\partial \mathcal{L}(x;\theta^k,\psi_m)}{\partial \psi_m}$
  **3.** Update $\theta$: $\theta^{k+1} \leftarrow \theta^k + \eta_\theta \nabla_{\theta^k} \mathcal{L}(x; \theta^k, \psi_M)$
**end for**
```

4.3.1 Limitations of joint parameter updates

Alg. (1) updates θ, ϕ jointly. During training, the inference network learns to approximate the posterior, and the generative model improves itself using local variational parameters $\psi(x)$ output by $q_\phi(z|x)$. If the variational parameters $\psi(x)$ output by the inference network are close to the optimal variational parameters ψ^* (Eq. 4.3), then the updates for θ are based on a relatively tight lower bound on $\log p(x)$. But in practice $\psi(x)$ may not be a good approximation to ψ^*.

Both the inference network and generative model are initialized randomly. At the start of learning, $\psi(x)$ is the output of a randomly initialized neural network, and will therefore be a poor approximation to the optimal parameters ψ^*. So the gradients used to update θ will be based on a very loose lower bound on $\log p(x)$. These gradients may push the generative model towards a poor local minimum – previous work has argued that deep neural networks (which form the conditional probability distributions $p_\theta(x|z)$) are often sensitive to initialization (Glorot & Bengio, 2010; Larochelle *et al.* , 2009). Even later in learning, $\psi(x)$ may yield suboptimal gradients for θ if the inference network is not powerful enough to find optimal variational parameters for all data points.

Learning in the original SVI scheme does not suffer from this problem, since the variational parameters are optimized within the inner loop of learning before updating to θ (i.e. in Alg. (2)); $\partial\theta$ is effectively derived using $\mathcal{L}(x; \theta, \psi^*)$). However, this method requires potentially an expensive iterative optimization.

This motivates blending the two methodologies for parameter estimation. Rather than rely entirely on the inference network, we use its output to "warm-start" an SVI-style optimization that yields higher-quality estimates of ψ^*, which in turn should yield more meaningful gradients for θ.

4.4 Improving estimates of variational parameters

Having highlighted ways in which sub-optimal variational parameters may affect learning, in this section, we present two improvements we propose towards improving the learning algorithm for deep generative models on sparse, high-dimensional data.

4.4.1 Between stochastic and amortized variational inference

We use the local variational parameters $\psi = \psi(x)$ predicted by the inference network to initialize an iterative optimizer. As in Alg. 2, we perform gradient ascent to maximize $\mathcal{L}(x; \theta, \psi)$ with respect to ψ. The resulting ψ_M approximates the optimal variational parameters: $\psi_M \approx \psi^*$. Since NFA is a continuous latent variable model, these updates can be achieved via the re-parameterization gradient (Kingma & Welling, 2014). We use ψ^* to derive gradients for θ under $\mathcal{L}(x; \theta, \psi^*)$. Finally, the parameters of the inference network (ϕ) are updated using stochastic backpropagation and gradient descent, holding fixed the parameters of the generative model (θ). Our procedure is detailed in Alg. 3 and depicted in Figure 4-4.

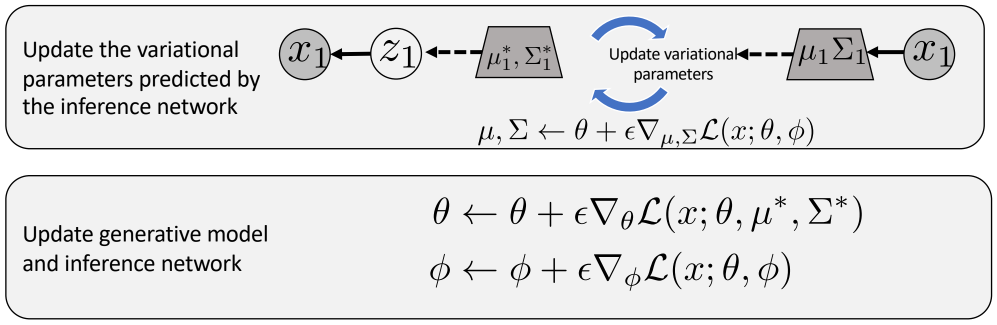

Update the variational parameters predicted by the inference network
x_1
z_1
μ_1^*, Σ_1^*
Update variational parameters
$\mu_1 \Sigma_1$
x_1
$\mu, \Sigma \leftarrow \theta + \epsilon \nabla_{\mu,\Sigma} \mathcal{L}(x; \theta, \phi)$
Update generative model and inference network
$\theta \leftarrow \theta + \epsilon \nabla_\theta \mathcal{L}(x; \theta, \mu^*, \Sigma^*)$
$\phi \leftarrow \phi + \epsilon \nabla_\phi \mathcal{L}(x; \theta, \phi)$

Figure 4-4: **Parameter estimation in NFA with a hybrid inference algorithm**

4.4.2 Representations for inference networks

The inference network must learn to regress to the optimal variational parameters for any combination of features, but in sparse datasets, many words appear only rarely.

Algorithm 3 Maximum likelihood estimation of θ with optimized local variational parameters: Expectations in $\mathcal{L}(x, \theta, \psi^*)$ (see Eq. 4.3) are evaluated with a single sample from the optimized variational distribution. M is the number of updates to the variational parameters ($M = 0$ implies no additional optimization). $\theta, \psi(x), \phi$ are updated using stochastic gradient descent with learning rates $\eta_\theta, \eta_\psi, \eta_\phi$ obtained via ADAM (Kingma & Ba, 2014). In step 4, we update ϕ separately from θ. One could alternatively, update ϕ using $\mathrm{KL}(\psi(x)_M \| q_\phi(z|x))$ as in Salakhutdinov & Larochelle (2010).

```
**Inputs**: $\mathcal{D} := [x_1, \dots, x_D]$,
Inference Model: $q_\phi(z|x)$,
Generative Model: $p_\theta(x|z), p(z)$,
**for** k = 1...K **do**
  **1.** Sample: $x \sim \mathcal{D}$ and set $\psi_0 = \psi(x)$
  **2.** Approx. $\psi_M \approx \psi^* = \arg\max_\psi \mathcal{L}(x; \theta^k; \psi)$,
       For $m = 0, \dots, M-1$:
           $\psi_{m+1} = \psi_m + \eta_\psi \frac{\partial \mathcal{L}(x;\theta^k, \psi_m)}{\partial \psi_m}$
  **3.** Update $\theta$,
       $\theta^{k+1} \leftarrow \theta^k + \eta_\theta \nabla_{\theta^k} \mathcal{L}(x; \theta^k, \psi_M)$
  **4.** Update $\phi$,
       $\phi^{k+1} \leftarrow \phi^k + \eta_\phi \nabla_{\phi^k} \mathcal{L}(x; \theta^{k+1}, \psi(x))$
**end for**
```

To provide more global context about rare words, we provide to the inference network (but not the generative network) TF-IDF (Baeza-Yates *et al.* , 1999) features instead of counts. These give the inference network a hint that rare words are likely to be highly informative. TF-IDF is a popular technique in information retrieval that re-weights features to increase the influence of rarer features while decreasing the influence of common features. The transformed feature-count vector is $\tilde{x}_{dv} \equiv x_{dv} \log \frac{D}{\sum_{d'} \min\{x_{d'v}, 1\}}$. The resulting vector $\tilde{x}$ is then normalized by its L2 norm.

4.4.3 Spectral analysis of the Jacobian matrix

One consequence of underfitting in latent variable modeling is a phenomenon known as overpruning in latent variable models, where only a small number of dimensions in the latent variable are used to model the data while the others remain *inactive* i.e. they revert to the prior distribution and have no discernible effect on the likelihood of observed data. In order to evaluate the efficacy of our proposed learning algorithm, we need a way to visualize how much of the latent space is being made use of by the model. To do this, we return to the technique developed in Chapter 3 and use of the Jacobian of the conditional likelihood.

For any vector valued function $f(x) : \mathbb{R}^K \to \mathbb{R}^V$, $\nabla_x f(x)$ is the matrix-valued function representing the sensitivity of the output to the input. When $f(x)$ is a deep neural network, Wang *et al.* (2016) use the spectra of the Jacobian matrix under various inputs x to quantify the complexity of the learned function. They find that the spectra are correlated with the complexity of the learned function. We adopt their technique for studying the utilization of the latent space in deep generative models. In the case of NFA, we seek to quantify the learned complexity of the generative model. To do so, we compute the Jacobian matrix as $\mathcal{J}(z) = \nabla_z \log p(x|z)$. This is a read-out measure of the sensitivity of the likelihood with respect to the latent dimension.

$\mathcal{J}(z)$ is a matrix valued function that can be evaluated at every point in the latent space. We evaluate it at the mode of the (unimodal) prior distribution i.e. at $z = \vec{0}$. The singular values of the resulting matrix denote how much the log-likelihood changes from the origin along the singular vectors lying in latent space. The intensity of these singular values (which we plot) is a read-out measure of how many intrinsic dimensions are utilized by the model parameters θ at the mode of the prior distribution. Our choice of evaluating $\mathcal{J}(z)$ at $z = \vec{0}$ is motivated by the fact that much of the probability mass in latent space under the NFA model will be placed at the origin. We use the utilization at the mode as an approximation for the utilization across the entire latent space. We visualized the spectral decomposition obtained under a Monte-Carlo approximation to the matrix $\mathbb{E}[\mathcal{J}(z)]$ and found it to be similar to the decomposition obtained by evaluating the Jacobian at the mode. Another possibility to measure utilization would be using the KL divergence of the prior and the output of the inference network (as in Burda *et al.* (2015)).

Unlike in Chapter 3, where we made use of the Jacobian matrix to introspect what the deep generative model had learned about data, here, we use it as a means to quantify how much information about the observations is captured by the generative model.

4.5 Related 2ork

Salakhutdinov & Larochelle (2010) optimize local mean-field parameters from an inference network in the context of learning deep Boltzmann machines. Salimans *et al.* (2015) explore warm starting MCMC with the output of an inference network.

Previous work has studied the failure modes of learning VAEs. They can be broadly categorized into two classes. The first aims to improves the utilization of latent variables using a richer posterior distribution (Burda *et al.* , 2015). However, for sparse data, the limits of learning with a normally distributed $q_\phi(z|x)$ have barely been pushed – our goal is to do so in this work. Further gains may indeed be obtained with a richer posterior distribution but the techniques herein can inform work along this vein. The second class of methods studies ways to alleviate the underutilization of latent dimensions due to an overly expressive choice of models for $p(x|z;\theta)$ such as a Recurrent Neural Network (Bowman *et al.* , 2015; Chen *et al.* , 2016). This too, is not the scenario we are in; underfitting of VAEs on sparse data occurs even when $p(x|z;\theta)$ is an MLP. Our study here exposes a third failure mode; one in which learning is challenging not just because of the objective used in learning but also because of the *characteristics* of the data.

Our work was among the first to study the effects of sub-optimal variational parameters in deep generative models; since then there have been several advances that have enriched our understanding of limitations of inference networks. Cremer *et al.* (2018) coin the term *amortization gap* to refer to the divergence between the variational distribution predicted by the inference network and the optimal variational distribution within the distributional family. They highlight that limitations in the predictions of the inference network, rather than the choice of variational family are responsible for amortization gaps observed in practice. In our work, we use SVI to update the variational parameters predicted by the inference network; however our updates to the parameters of the inference network do not make use of the optimized variational parameters. Kim *et al.* (2018) derive gradients of the inference network through the computation of optimized variational parameters – they find that doing so yields strong results in building deep generative models of sequence data where $p(x|z;\theta)$ is parameterized by a recurrent neural network. He *et al.* (2019) propose a simple, yet effective approach to improve the quality of inference networks, for each update of the generative model's parameters, conduct multiple updates to the parameters of the inference network. Finally, Lucas *et al.* (2019) perform a case study on probabilistic PCA, a linear factor model, and study how posterior collapse, or the underutilization of the model's latent variable, can arise due to the existance of multiple local optima in the log-marginal likelihood.

4.6 Evaluation

We first confirm our hypothesis empirically that underfitting is an issue when learning VAEs on high dimensional sparse datasets. We quantify the gains (at training and test time) obtained by the use of TF-IDF features and the continued optimization of $\psi(x)$ on two different types of high-dimensional sparse data—text and movie ratings. In Section 4.6.2, we learn VAEs on two large scale bag-of-words datasets. We study (1) *where* the proposed methods might have the most impact and (2) present evidence for *why* the learning algorithm (Alg. 3) works. In Section 4.6.3, we show that improved inference is crucial to building deep generative models that can tackle problems in recommender systems.

4.6.1 Setup

Notation: In all experiments, $\psi(x)$ denotes learning with Alg. 1 and ψ^* denotes the results of learning with Alg. 3. $M = 100$ (number of updates to the local variational parameters) on the bag-of-words text data and $M = 50$ on the recommender systems task. M was chosen based on the number of steps it takes for $\mathcal{L}(x;\theta,\psi_m)$ (Step 2 in Alg. 3) to converge on training data. 3-ψ^*-norm denotes a model where the MLP parameterizing $\gamma(z)$ has three layers: two hidden layers and one output layer, ψ^* is used to derive an update of θ and normalized count features are conditioned on by the inference network. In all tables, we display evaluation metrics obtained under both $\psi(x)$ (the output of the inference network) and ψ^* (the optimized variational parameters). In figures, we always display metrics obtained under ψ^* (even if the model was trained with $\psi(x)$) since $\mathcal{L}(x;\theta,\psi^*)$ always forms a tighter bound to $\log p(x)$. If left unspecified TF-IDF features are used as input to the inference network.

Training and Evaluation: We update θ using learning rates given by ADAM (Kingma & Ba, 2014) (using a batch size of 500), The inference network's intermediate hidden layer $h(x) = \text{MLP}(x;\phi_0)$ (we use a two-layer MLP in the inference network for all experiments) are used to parameterize the mean and diagonal log-variance as: $\mu(x) = W_\mu h(x), \log\Sigma(x) = W_{\log\Sigma}h(x)$ where $\phi = \{W_\mu, W_{\log\Sigma}, \phi_0\}$. Code is available at `github.com/rahulk90/vae_sparse`.

4.6.2 Bag-of-words text data

Datasets and Metrics: We study two large text datasets.

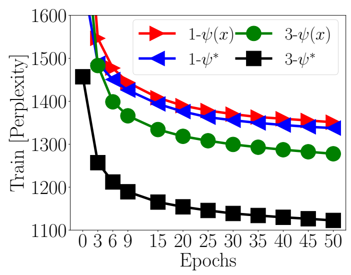
1. RCV1 (Lewis *et al.* , 2004) dataset (train/valid/test: 789,414/5,000/10,000, V: 10,000). We follow the preprocessing procedure in Miao *et al.* (2016),
2. Wikipedia corpus used in Huang *et al.* (2012) (train/test: 1,104,937/100,000 and V:20,000). We set all words to lowercase, ignore numbers and restrict the dataset to the top $20,000$ frequently occurring words.

We report an upper bound on perplexity (Mnih & Gregor, 2014) given by

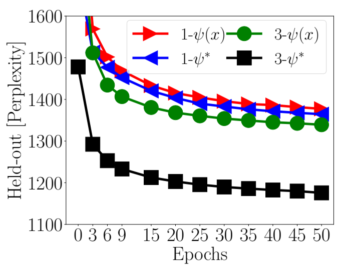
$$\exp(-\frac{1}{N}\sum_{i}\frac{1}{N_i}\log p(x_i))$$

where $\log p(x_i)$ is replaced by Eq 4.2.

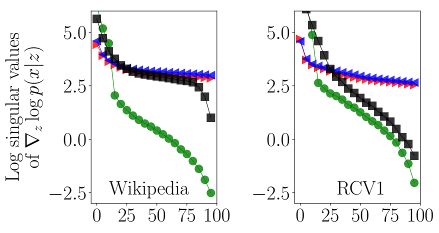
To study the utilization of the latent dimension obtained by various training methods, we compute the Jacobian $\mathcal{J}(z)$ matrix (as $\nabla_z \log p(x|z)$). The singular value spectrum of the Jacobian directly measures the *utilization* of the latent dimensions in the model.



(a) Wikipedia - Training



(b) Wikipedia - Evaluation



(c) Log-Singular Values

Figure 4-5: **Mechanics of learning:** Best viewed in color. **(Left and Middle)** For the Wikipedia dataset, we visualize upper bounds on training and held-out perplexity (*evaluated* with ψ^*) viewed as a function of epochs. Items in the legend corresponds to choices of training method. **(Right)** Sorted log-singular values of $\nabla_z \log p(x|z)$ on Wikipedia (left) on RCV1 (right) for different training methods. The x-axis is latent dimension. The legend is identical to that in Fig. 4-5a.

Reducing Underfitting: Is underfitting a problem and does optimizing $\psi(x)$ with the use of TF-IDF features help? Table 4.1 confirms both statements.

Between "norm" and "tfidf" (comparing first four rows and second four rows), we find that the use of TF-IDF features almost always improves parameter estimation.

Furthermore, optimizing $\psi(x)$ at test time (comparing column ψ^* with $\psi(x)$) always yields a tighter bound on $\log p(x)$, often by a wide margin. Even after extensive training the inference network can fail to tightly approximate $\mathcal{L}(x;\theta,\psi^*)$, suggesting that there may be limitations to the power of generic amortized inference. Optimizing $\psi(x)$ during training ameliorates under-fitting and yields significantly better generative models on the RCV1 dataset. The degree of underfitting and subsequently the improvements from training with ψ^* are significantly more pronounced on the larger and sparser Wikipedia dataset (Fig. 4-5a and 4-5b).

Effect of optimizing $\psi(x)$: How does learning with ψ^* affect the rate of convergence the learning algorithm? We plot the upper bound on perplexity versus epochs on the Wikipedia (Fig. 4-5a, 4-5b) datasets. As in Table 4.1, the additional optimization does not appear to help much when the generative model is linear. On the deeper three-layer model, learning with ψ^* dramatically improves the model allowing it to fully utilize its potential for density estimation. Models learned with ψ^* quickly converge to a better local minimum early on (as reflected in the perplexity evaluated on the training data and held-out data). We experimented with continuing to train 3-$\psi(x)$ beyond 150 epochs, where it reached a validation perplexity of approximately 1330, worse than that obtained by 3-ψ^* at epoch 10 suggesting that longer training is insufficient to overcome local minima issues afflicting VAEs.

Overpruning of latent dimensions: One cause of underfitting is due to overpruning of the latent dimensions in the model. If the variational distributions for

Table 4.1: **Test perplexity on RCV1: Left: Baselines** Legend: LDA (Blei *et al.* , 2003), Replicated Softmax (RSM) (Hinton & Salakhutdinov, 2009), Sigmoid Belief Networks (SBN) and Deep Autoregressive Networks (DARN) (Mnih & Gregor, 2014), Neural Variational Document Model (NVDM) (Miao *et al.* , 2016). K denotes the latent dimension in our notation. **Right:** NFA on text data with $K = 100$. We vary the features presented to the inference network $q_\phi(z|x)$ during learning between: normalized count vectors ($\frac{x}{\sum_{i=1}^{V} x_i}$, denoted "norm") and normalized TF-IDF

Model	K	**RCV1**	**NFA**	$\psi(x)$	ψ^*
LDA	50	1437	1-$\psi(x)$-norm	501	481
LDA	200	1142	1-ψ^*-norm	488	454
RSM	50	988	3-$\psi(x)$-norm	396	355
SBN	50	784	3-ψ^*-norm	378	**331**
fDARN	50	724	1-$\psi(x)$-tfidf	480	456
fDARN	200	598	1-ψ^*-tfidf	482	454
NVDM	50	563	3-$\psi(x)$-tfidf	384	344
NVDM	200	550	3-ψ^*-tfidf	376	**331**

a subset of the latent dimensions of z are set to the prior, this effectively reduces the model's capacity. If the KL-divergence in Eq. 4.2 encourages the approximate posterior to remain close to the prior early in training, *and* if the gradient signals from the likelihood term are weak or inconsistent, the KL may dominate and prune out latent dimensions before the model can use them. In Fig. 4-5c, we plot the log-spectrum of the Jacobian matrices for different training methods and models. For the deeper models, optimizing $\psi(x)$ is crucial to utilizing its capacity, particularly on the sparser Wikipedia data. Without it, only about ten latent dimensions are used, and the model severely underfits the data. Optimizing $\psi(x)$ iteratively likely limits overpruning since the variational parameters (ψ^*) don't solely focus on minimizing the KL-divergence but also on maximizing the likelihood of the data (the first term in Eq. 4.2).

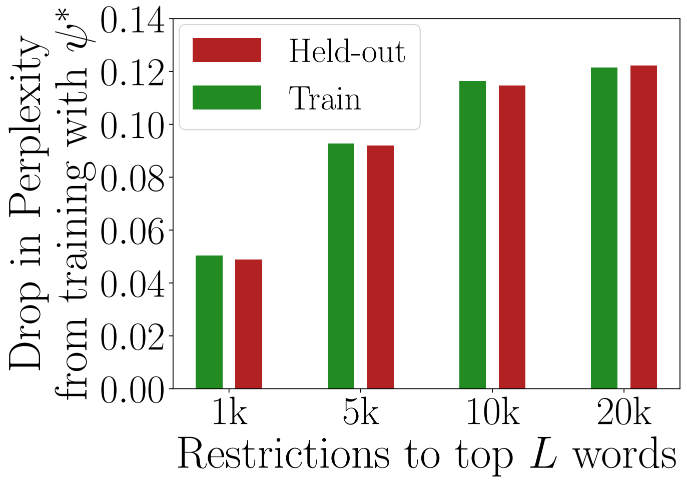
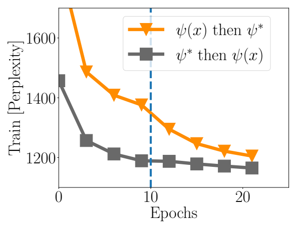


Figure 4-6: **Decrease in perplexity versus sparsity:** We plot the relative drop in perplexity obtained by training with ψ^* instead of $\psi(x)$ against varying levels of sparsity in the Wikipedia data. On the y-axis, we plot $\frac{P_{[3-\psi(x)]}-P_{[3-\psi^*]}}{P_{[3-\psi(x)]}}$; P denotes the bound on perplexity (evaluated with ψ^*) and the subscript denotes the model and method used during training. Each point on the x-axis is a restriction of the dataset to the top L most frequently occurring words (number of features).

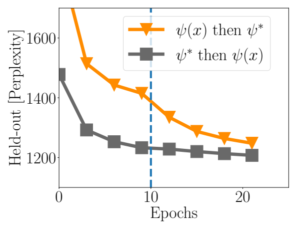
Sparse data is challenging: What is the relationship between data sparsity and how well inference networks work? We hold fixed the number of training samples and vary the sparsity of the data. We do so by restricting the Wikipedia dataset to the top L most frequently occurring words. We train three layer generative models on the different subsets. On training and held-out data, we computed the difference between the perplexity when the model is trained with (denoted $P_{[3-\psi^*]}$) and without optimization of $\psi(x)$ (denoted $P_{[3-\psi(x)]}$). We plot the relative decrease in perplexity obtained by training with ψ^* in Fig. 4-6.

Learning with ψ^* helps *more* as the data dimensionality increases. Data sparsity, therefore, poses a significant challenge to inference networks. One possible explanation is that many of the tokens in the dataset are rare, and the inference network therefore needs many sweeps over the dataset to learn to properly interpret these rare words; while the inference network is learning to interpret these rare words the generative model is receiving essentially random learning signals that drive it to a poor local optimum.

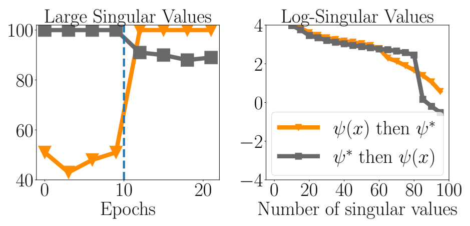
Designing new strategies that can deal with such data may be a fruitful direction for future work. This may require new architectures or algorithms—we found that simply making the inference network deeper does not solve the problem.



(a) Training Data



(b) Held-out Data



(c) Log-singular Values

Figure 4-7: **Late versus early optimization of $\psi(x)$:** Fig. 4-7a (4-7b) denote the train (held-out) perplexity for three-layered models trained on the Wikipedia data in the following scenarios: ψ^* is used for training for the first ten epochs following which $\psi(x)$ is used (denoted "ψ^* then $\psi(x)$") and vice versa (denoted "$\psi(x)$ then ψ^*"). Fig. 4-7c (Left) depicts the number of singular values of the Jacobian matrix $\nabla_z \log p(x|z)$ with value greater than 1 as a function of training epochs for each of the two aforementioned methodologies. Fig. 4-7c (Right) plots the sorted log-singular values of the Jacobian matrix corresponding to the final model under each training strategy.

When should $\psi(x)$ be optimized: *When* are the gains obtained from learning with ψ^* accrued? We learn three-layer models on Wikipedia under two settings: (a) we train for 10 epochs using ψ^* and then 10 epochs using $\psi(x)$. and (b) we do the opposite.

Fig. 4-7 depicts the results of this experiment. We find that: (1) much of the gain from optimizing $\psi(x)$ comes from the early epochs, (2) somewhat surprisingly using ψ^* instead of $\psi(x)$ later on in learning also helps (as witnessed by the sharp drop in perplexity after epoch 10 and the number of large singular values in Fig. 4-7c [Left]). This suggests that even after seeing the data for several passes, the inference network is unable to find $\psi(x)$ that explain the data well. Finally, (3) for a fixed computational budget, one is better off optimizing $\psi(x)$ sooner than later – the curve that optimizes $\psi(x)$ later on does not catch up to the one that optimizes $\psi(x)$ early in learning. This suggests that learning early with ψ^*, even for a few epochs, may alleviate underfitting.

Rare words and loose lower bounds: Fig. 4-6 suggests that data sparsity presents a problem for inference networks at an aggregate level. We now ask which *data points* benefit from the optimization of $\psi(x)$? We sample 20000 training and held-out data points; we compute $\mathrm{KL}(\psi(x)\|\psi^*)$ (both are Normal distributions and the KL is analytic) and the number of rare words in each document (where a word is classified as being rare if it occurs in less than 5% of training documents). We visualize them in Fig. 4-8. We also display the Spearman ρ correlation between the two values in Fig. 4-8. There exists a positive correlation (about 0.88 on the training data) between the two values suggesting that the gains in perplexity that we observe empirically in Table 4.1 and Fig. 4-5 are due to being able to better model the likelihood of documents with rare words in them.

We present another way to visualize the results of Fig. 4-8. We sample 20000 training and held-out data points; we compute $\mathrm{KL}(\psi(x)\|\psi^*)$ (both are Normal distributions and the KL is analytic) and the number of rare words in each document (where a word is classified as being rare if it occurs in less than 5% of training documents). We scale each value to be between 0 and 1 using: $\frac{c_i - min(c)}{max(c) - min(c)}$ where c is the vector of KL divergences or number of rare words. We sort the scaled values by the KL divergence and plot them in Fig. 4-9. As before, we observe that the documents that we move the farthest in KL divergence are those which have many rare words.

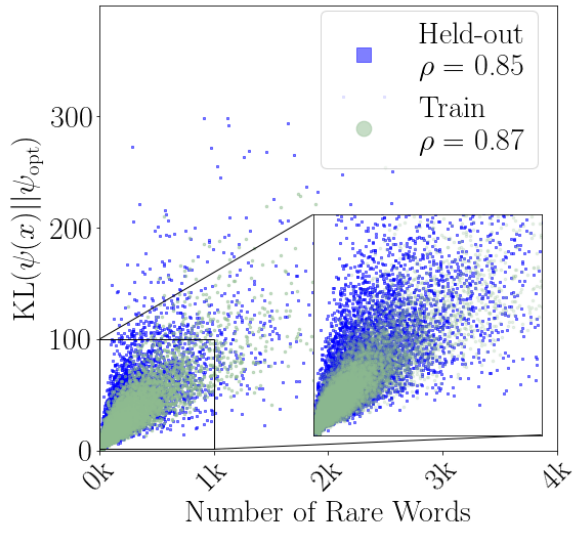


Figure 4-8: **KL divergence and rare word counts:** We plot the values of $\mathrm{KL}(\psi(x)\|\psi^*)$ versus the number of rare words. We zoom into the plot and reduce the opacity of the train points to better see the held-out points. The Spearman ρ correlation coefficient is computed between the two across $20,000$ points. We find a positive correlation.

Learning with ψ^* on small data: We study the role of learning with ψ^* in the small-data regime. Table 4.2 depicts the results obtained after training models for 200 passes through the data. We summarize our findings: (1) across the board, TF-IDF features improve learning, and (2) in the small data regime, deeper non-linear models (3-ψ^*-tfidf) overfit quickly and better results are obtained by the simpler multinomial-logistic PCA model (1-ψ^*-tfidf). Overfitting is also evident in Fig. 4-10

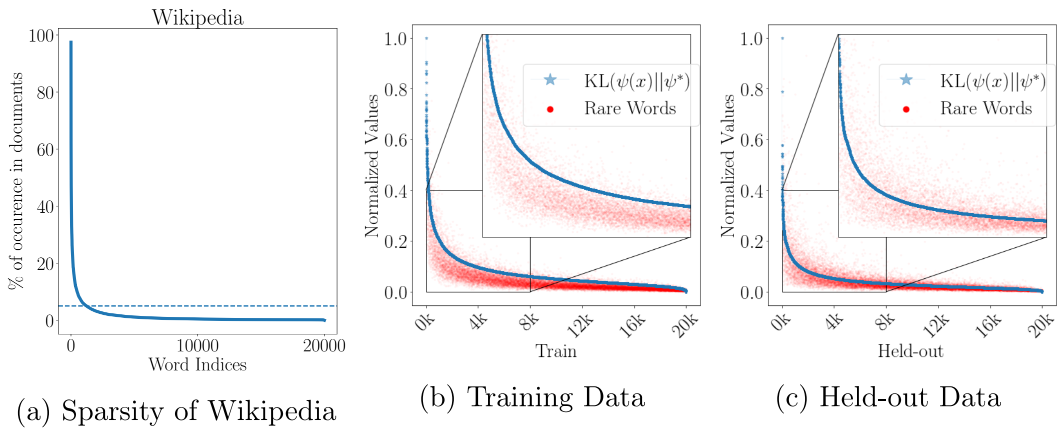


Figure 4-9: **Normalized KL and Rare Word Counts:** Fig. 4-9a depicts percentage of times words appear in the Wikipedia dataset (sorted by frequency). The dotted line in blue denotes the marker for a word that has a 5% occurrence in documents. In Fig. 4-9b, 4-9c, we superimpose (1) the normalized (to be between 0 and 1) values of $\text{KL}(\psi(x)\|\psi^*)$ and (2) the normalized number of rare words (sorted by value of the KL-divergence) for $20,000$ points (on the x-axis) randomly sampled from the train and held-out data.

from comparing curves on the validation set to those on the training set. Interestingly, in the small dataset setting, we see that learning with $\psi(x)$ has the potential to have a regularization effect in that the results obtained are not much worse than those obtained from learning with ψ^*.

For completeness, in Fig. 4-11, we also provide the training behavior for the RCV1 dataset corresponding to the results of Table 4.1. The results here, echo the convergence behavior on the Wikipedia dataset.

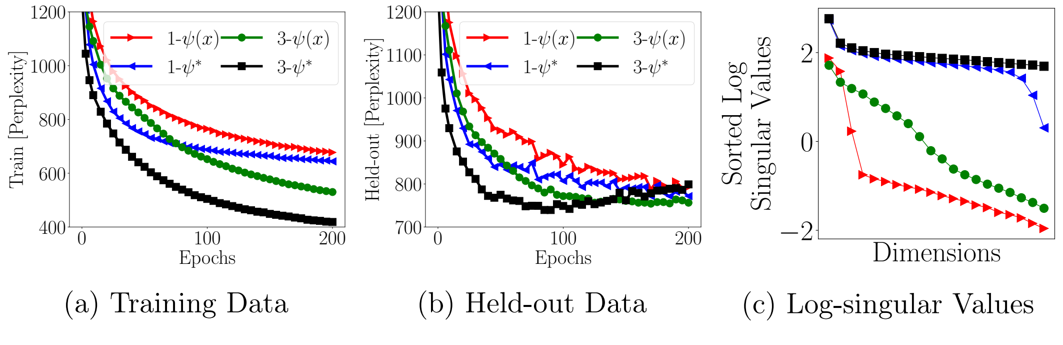


Figure 4-10: **20Newsgroups - training and held-out bounds:** Fig. 4-10a, 4-10b denotes the train (held-out) perplexity for different models. Fig. 4-10c depicts the log-singular values of the Jacobian matrix for the trained models.

Comparison with KL-annealing: An empirical observation made in previous work is that when $p(x|z;\theta)$ is complex (parameterized by a recurrent neural network or a neural autoregressive density estimator (NADE)), the generative model also must

Table 4.2: **Test perplexity on 20newsgroups: Left: Baselines** Legend: LDA (Blei *et al.* , 2003), Replicated Softmax (RSM) (Hinton & Salakhutdinov, 2009), Sigmoid Belief Networks (SBN) and Deep Autoregressive Networks (DARN) (Mnih & Gregor, 2014), Neural Variational Document Model (NVDM) (Miao *et al.* , 2016). K denotes the latent dimension in our notation. **Right:** NFA on text data with $K = 100$. We vary the features presented to the inference network $q_\phi(z|x)$ during learning between: normalized count vectors ($\frac{x}{\sum_{i=1}^{V} x_i}$, denoted "norm") and normalized TF-IDF (denoted "tfidf") features.

Model	K	**Results**
LDA	50	1091
LDA	200	1058
RSM	50	953
SBN	50	909
fDARN	50	917
fDARN	200	—
NVDM	50	836
NVDM	200	852

NFA	**Perplexity**	
	$\psi(x)$	ψ^*
1-$\psi(x)$-norm	1018	903
1-ψ^*-norm	1279	889
3-$\psi(x)$-norm	986	857
3-ψ^*-norm	1292	879
1-$\psi(x)$-tfidf	932	839
1-ψ^*-tfidf	953	828
3-$\psi(x)$-tfidf	999	842
3-ψ^*-tfidf	1063	839

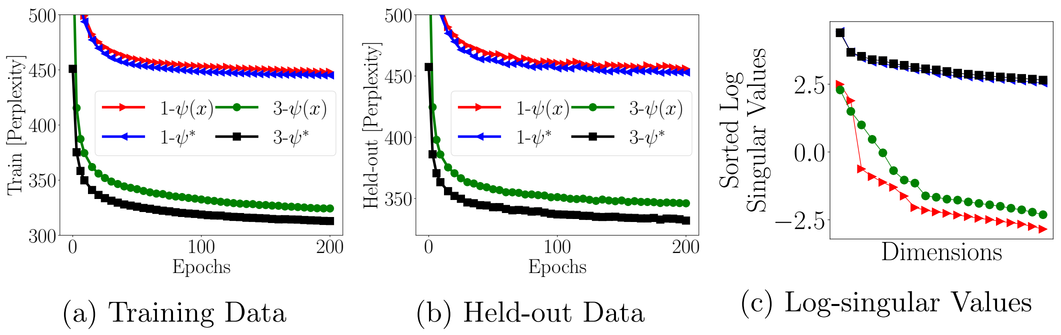
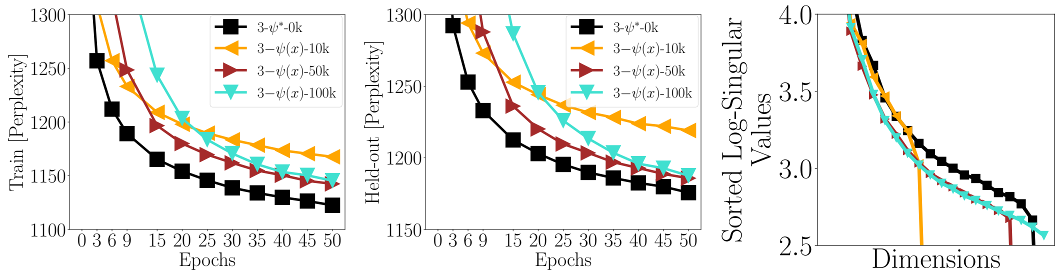


Figure 4-11: **RCV1 - training and held-out bounds:** Fig. 4-11a, 4-11b denotes the train (held-out) perplexity for different models. Fig. 4-11c depicts the log-singular values of the Jacobian matrix for the trained models.

contend with overpruning of the latent dimension. A proposed fix is the annealing of the KL divergence term in Equation 4.2 (e.g., Bowman *et al.* , 2015) as one way to overcome local minima. Although this is a different scenario to the one we present in that our decoder is a MLP – nonetheless, we apply KL annealing within our setting.

In particular, we optimized $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z))] - \eta\,\mathrm{KL}(\,q_\phi(z|x)||p(z)\,)$ where η was annealed from 0 to 1 (linearly – though we also tried exponential annealing) over the course of several parameter updates. Note that doing so does *not* give us a lower bound on the likelihood of the data anymore. There are few established guidelines about the rate of annealing the KL divergence and in general, we found it tricky to get it to work reliably. We experimented with different rates of annealing for learning a three-layer generative model on the Wikipedia data.

Our findings (visualized in Fig. 4-12) are as follows: (1) on sparse data we found annealing the KL divergence is very sensitive to the annealing rate – too small an annealing rate and we were still left with underfitting (as in annealing for 10k), too high an annealing rate (as in 100k) and this resulted in slow convergence; (2) learning with ψ^* always outperformed (in both rate of convergence and quality of final result on train and held-out data) annealing the KL divergence across various choices of annealing schedules. Said differently, on the Wikipedia dataset, we conjecture there exists a choice of annealing of the KL divergence for which the perplexity obtained *may* match those of learning with ψ^* but finding this schedule requires significant trial and error – Fig. 4-12 suggests that we did not find it. We found that learning with ψ^* required less tuning (setting values of M to be larger than 100 never hurt) and always performed at par or better than annealing the KL divergence. Furthermore, we did not find annealing the KL to work effectively for the experiments on the recommender systems task. In particular, we were unable to find an annealing schedule that reliably produced good results.

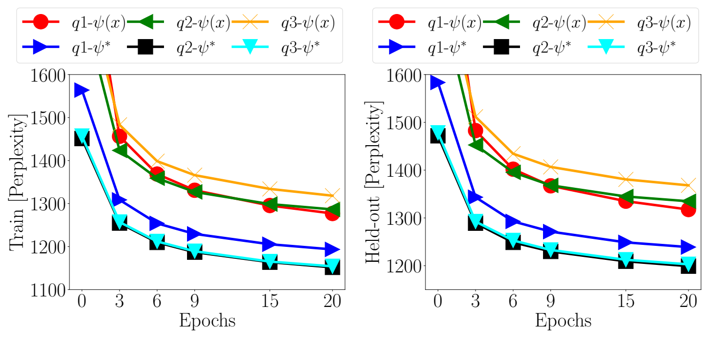


(a) Training Data (b) Held-out Data (c) Log-singular Values

Figure 4-12: **KL annealing vs learning with** ψ^* Fig. 4-12a, 4-12b denotes the train (held-out) perplexity for different training methods. The suffix at the end of the model configuration denotes the number of parameter updates that it took for the KL divergence in Equation 4.2 to be annealed from 0 to 1. 3-ψ^*-50k denotes that it took 50000 parameter updates before $-\mathcal{L}(x;\theta,\psi(x))$ was used as the loss function. Fig. 4-10c depicts the log-singular values of the Jacobian matrix for the trained models.

Depth of $q_\phi(z|x)$: Can the overall effect of the additional optimization be *learned* by the inference network at training time? The experimental evidence we observe in Fig. 4-13 suggests this is difficult.

When learning with $\psi(x)$, increasing the number of layers in the inference network slightly decreases the quality of the model learned. This is likely because the already stochastic gradients of the inference network must propagate along a longer path in a deeper inference network, slowing down learning of the parameters ϕ which in turn affects $\psi(x)$, thereby reducing the quality of the gradients used to updated θ.



(a) Training Data (b) Held-out Data

Figure 4-13: **Varying the depth of $q_\phi(z|x)$:** Fig. 4-12a (4-12b) denotes the train (held-out) perplexity for a three-layer generative model learned with inference networks of varying depth. The notation q3-ψ^* denotes that the inference network contained a two-layer intermediate hidden layer $h(x) = \text{MLP}(x; \phi_0)$ followed by $\mu(x) = W_\mu h(x), \log \Sigma(x) = W_{\log \Sigma} h(x)$.

Putting it all together: Our analysis describes a narrative of how underfitting occurs in learning VAEs on sparse data. The rare words in sparse, high-dimensional data are difficult to map into local variational parameters that model the term $\mathbb{E}[\log p(x|z)]$ well (Fig. 4-6,4-8); $q_\phi(z|x)$ therefore focuses on the less noisy (the KL is evaluated analytically) signal of minimizing $\text{KL}(q_\phi(z|x)||p(z))$. Doing so prunes out many latent dimensions early on resulting in underfitting (Fig. 4-7c [Left]). By using ψ^*, the inadequacies of the inference network are decoupled from the variational parameters used to derive gradients to θ. The tighter variational bound $\mathcal{L}(x; \theta, \psi^*)$ achieves a better tradeoff between $\mathbb{E}[\log p(x|z)]$ and $\text{KL}(q_\phi(z|x)||p(z))$ (evidenced by the number of large singular values of $\nabla_z \log p(x|z)$ when optimizing ψ^* in Fig. 4-7c). The gradient updates with respect to this tighter bound better utilize θ.

4.6.3 Collaborative filtering

Modeling rare features in sparse, high-dimensional data is necessary to achieve strong results on this task. We study the top-N recommendation performance of NFA under strong generalization (Marlin & Zemel, 2009).

Datasets: We study two large user-item rating datasets: MovieLens-20M (ML-20M)

(Harper & Konstan, 2015) and Netflix[2]. Following standard procedure: we binarize the explicit rating data, keeping ratings of four or higher and interpreting them as implicit feedback (Hu *et al.* , 2008) and keep users who have positively rated at least five movies. We train with users' binary implicit feedback as x_d; the vocabulary is the set of all movies. The number of training/validation/test users is 116,677/10,000/10,000 for ML-20M (V: 20,108) and 383,435/40,000/40,000 for Netflix (V: 17,769).

Evaluation and metrics: We train with the complete feedback history from training users, and evaluate on held-out validation/test users. We select model architecture (MLP with 0, 1, 2 hidden layers) from the held-out validation users based on NDCG@100 and report metrics on the held-out test users. For held-out users, we randomly select 80% of the feedback as the input to the inference network and see how the other 20% of the positively rated items are ranked based $\mu(z)$. We report two ranking-based metrics averaged over all held-out users: Recall@N and truncated normalized discounted cumulative gain (NDCG@N) (Järvelin & Kekäläinen, 2002). For each user, both metrics compare the predicted rank of unobserved items with their true rank. While Recall@N considers all items ranked within the first N to be equivalent, NDCG@N uses a monotonically increasing discount to emphasize the importance of higher ranks versus lower ones.

Define π as a ranking over all the items where $\pi(v)$ indicates the v-th ranked item, $\mathbb{I}\{\cdot\}$ is the indicator function, and $d(\pi(v))$ returns 1 if user d has positively rated item $\pi(v)$. Recall@N for user d is

$$\text{Recall@}N(d, \pi) := \sum_{v=1}^{N} \frac{\mathbb{I}\{d(\pi(v)) = 1\}}{\min(N, \sum_{v'}^{V} \mathbb{I}\{d(\pi(v')) = 1\})}.$$

The expression in the denominator evaluates to the minimum between N and the number of items consumed by user d. This normalizes Recall@N to have a maximum of 1, which corresponds to ranking all relevant items in the top N positions. Discounted cumulative gain (DCG@N) for user d is

$$\text{DCG@}N(d, \pi) := \sum_{v=1}^{N} \frac{2^{\mathbb{I}\{d(\pi(v))=1\}} - 1}{\log(v+1)}.$$

NDCG@N is the DCG@N normalized by ideal DCGN, where all the relevant items are ranked at the top. We have, NDCG@$N \in [0, 1]$. As baselines, we consider:

[2]`http://www.netflixprize.com/`

Weighted matrix factorization (WMF) (Hu *et al.* , 2008): a linear low-rank factor model. We train WMF with alternating least squares; this generally leads to better performance than with SGD.

SLIM (Ning & Karypis, 2011): a linear model which learns a sparse item-to-item similarity matrix by solving a constrained ℓ_1-regularized optimization problem.

Collaborative denoising autoencoder (CDAE) (Wu *et al.* , 2016): An autoencoder achitecture specifically designed for top-N recommendation. It augments a denoising autoencoder (Vincent *et al.* , 2008) by adding a per-user latent vector to the input, inspired by standard linear matrix-factorization approaches. Among the baselines, CDAE is most akin to NFA.

Table 4.3 summarizes the results of NFA under different settings. We found that optimizing $\psi(x)$ helps both at train and test time and that TF-IDF features consistently improve performance. Crucially, the standard training procedure for VAEs realizes a poorly trained model that underperforms every baseline. The improved training techniques we recommend generalize across different kinds of sparse data. With them, *the same generative model*, outperforms CDAE and WMF on both datasets, and marginally outperforms SLIM on ML-20M while achieving nearly state-of-the-art results on Netflix. In terms of runtimes, we found that learning NFA (with ψ^*) to be approximately two-three times faster than SLIM. Our results highlight the importance of inference at training time showing NFA, when properly fit, can outperform the popular linear factorization approaches.

Table 4.3: **Recall and NDCG on recommender systems:** "2-ψ^*-tfidf" denotes a two-layer (one hidden layer and one output layer) generative model. Standard errors are around 0.002 for ML-20M and 0.001 for Netflix. **Runtime:** WMF takes on the order of minutes [ML-20M & Netflix]; CDAE and NFA ($\psi(x)$) take 8 hours [ML-20M] and 32.5 hours [Netflix] for 150 epochs; NFA (ψ^*) takes takes 1.5 days [ML-20M] and 3 days [Netflix]; SLIM takes 3-4 days [ML-20M] and 2 weeks [Netflix].

ML-20M	Recall@50		NDCG@100		**Netflix**	Recall@50		NDCG@100	
NFA	$\psi(x)$	ψ^*	$\psi(x)$	ψ^*	**NFA**	$\psi(x)$	ψ^*	$\psi(x)$	ψ^*
2-$\psi(x)$-norm	0.475	0.484	0.371	0.377	2-$\psi(x)$-norm	0.388	0.393	0.333	0.337
2-ψ^*-norm	0.483	0.508	0.376	0.396	2-ψ^*-norm	0.404	0.415	0.347	0.358
2-$\psi(x)$-tfidf	0.499	0.505	0.389	0.396	2-$\psi(x)$-tfidf	0.404	0.409	0.348	0.353
2-ψ^*-tfidf	0.509	**0.515**	0.395	**0.404**	2-ψ^*-tfidf	0.417	0.424	0.359	0.367
WMF	0.498		0.386		WMF	0.404		0.351	
SLIM	0.495		0.401		SLIM	**0.427**		**0.378**	
CDAE	0.512		0.402		CDAE	0.417		0.360	

4.7 Discussion

Studying the failures of learning with inference networks is an important step to designing more robust neural architectures for inference networks. We show that avoiding gradients obtained using poor variational parameters is vital to successfully learning VAEs on sparse data. An interesting question is *why* inference networks have a harder time turning sparse data into variational parameters compared to images? One hypothesis is that the redundant correlations that exist among pixels (but occur less frequently in features found in sparse data) are more easily transformed into local variational parameters $\psi(x)$ that are, in practice, often reasonably close to ψ^* during learning.

Hjelm *et al.* explore a similar idea as ours to derive an importance-sampling-based bound for learning deep generative models with discrete latent variables. They find that learning with ψ^* does not improve results on binarized MNIST. This is consistent with our experience—we find that our secondary optimization procedure helped more when learning models of sparse data. Miao *et al.* learn log-linear models (multinomial-logistic PCA, Collins *et al.* , 2001) of documents using inference networks. We show that mitigating underfitting in deeper models yields better results on the benchmark RCV1 data. Our use of the spectra of the Jacobian matrix of $\log p(x|z)$ to inspect learned models is inspired by Wang *et al.* (2016).

Chapter 5

Supervised fine-tuning of deep generative models

Supervision in machine learning comes in many forms. Although the canonical form of supervised data is in the form of labelled annotations (for each datapoint), this may not always be the best way to guide learning algorithm. One form of supervision of interest are pairwise expressions of similarity between datapoints.

A motivating example is the task of patient similarity where a doctor may be interested in searching a hospital database to find *similar* patients. For example, a doctor may have a set of patients who they believe are similar because they respond rapidly to a treatment. The doctor may be interested in finding other patients who respond similarly. In this scenario, the doctor does not prescribe discrete labels to every patient, but rather identifies similar patients based on how much they satisfy a clinical criteria such as their response to treatment.

We build algorithms to tackle such a problem by way of analogy to few-shot learning. In few-shot learning, a learner is given access to sets of datapoints that are assumed to be similar. At test time, the learner is given a target datapoint and multiple query sets. Each query set, comprising one or more datapoints, is a candidate to be identified as being most similar to the target.

To make the analogy to healthcare precise, the target datapoint may comprise patient data from a new patient and the query sets may comprise pre-defined sets of patients, each exhibiting known phenotypic characteristics. The goal of the learner is to identify which set of phenotypic characteristic the new patient is most similar to. In this chapter, we an approach to tackle such a problem via the characterization of similarity as overlap in the latent space of a deep generative model.

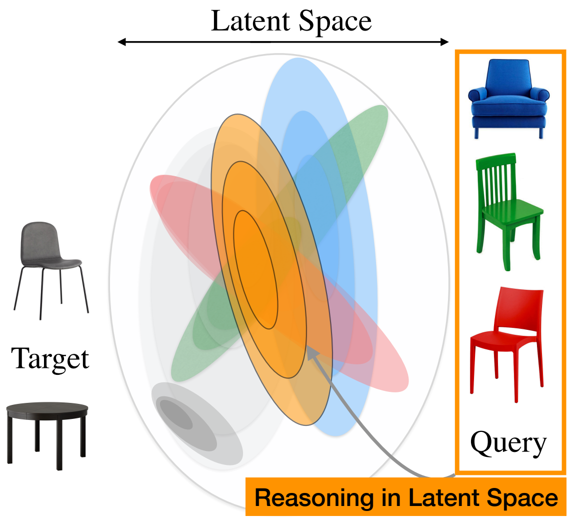


Figure 5-1: **Comparing objects in representational space:** On the left is a target set that will be ranked based on similarity to the query Q (right). The colour of each object is matched to a distribution in representation space. In orange is the output of the *latent reasoning network* – it represents the common factor of variation shared by Q. The black chair should rank higher than the black table; here its distribution (in representation space) overlaps more with the output of the latent reasoning network.

5.1 Introduction

How can we frame the problem of selecting, from a target set, an object most similar to a given query set? For example—given a red chair, a blue chair and a black chair, we would rank chairs in the target set highly. At the same time, given a red chair, a red car and a red shirt, we would rank red objects highly. Between the two tasks, our understanding of the *data* has not changed; what has changed is our understanding of the *task based on the context* given by the query. The query highlights the relevant property of the data that is needed for solving a specific task. Such tasks appear in few-shot learning, where the goal is ranking objects according to their similarity to a given query set and in healthcare where a task may be finding similar patients to a given cohort.

To answer such queries, we could train discriminative models attuned to answering set-conditional queries at test time (e.g. Vinyals *et al.* (2016)). Or we could encode class separability in the structure of a generative model (Edwards & Storkey, 2016) and use inference for prediction.

We learn a generic representation space (using unsupervised data) that is warped (using supervised data) for potentially different test-time problems. The task of scoring objects given a query is decomposed into two subtasks. The first determines the common property shared by items in the query set and represents the property as a region in representation space. In Figure 5-1, we visualize such a hypothetical space. On the right is a query comprising chairs of different colors and (in orange) a region of space that characterizes the property (in this case, a likeness to a chair) common to items in the query. The second task is to score a target item based on how much it expresses the region of representation space shared by items in the query. For the two candidate target points in Figure 5-1 (left), the black chair would rank rank highly since its representation has more in common with the property encapsulated by the query.

Here, we will use the latent space of deep generative models (Rezende *et al.* , 2014; Kingma & Welling, 2014) as our representation space. In such models, one can use the inference network to do posterior inference and map from raw data onto a distribution in latent space. However, to find commonalities among a set of multiple query items, we need a way to aggregate the information contained in multiple posterior distributions. Therefore, we introduce a *latent reasoning network* (LRN). The LRN takes a query as input and constructs a probability distribution over the latent space that *summarizes* the representations of the query points into a single distribution. Figure 5-1 (orange) depicts what the output of the LRN might look like. We design the neural architecture for the LRN to be permutation invariant, based on Zaheer *et al.* (2017), so that it does not depend on the size of the query set. To identify whether a target point is similar to a query, we assign a *score* to the latent space of a target item. We propose using the logarithm of the Bayes Factor (Jeffreys, 1998) which measures how conditioning on the query alters the likelihood of a target point. Our approach is inspired by Bayesian Sets Ghahramani & Heller (2005) wherein data was assumed to be modeled by a hierarchical exponential family distribution and the likelihood ratio of the joint distribution and product of marginals was shown to be a useful measure of similarity.

The latent (representation) space of a deep generative model learned with unsupervised data is typically non-identifiable. i.e. there will exist multiple good (from the perspective of log-likelihood) representation spaces. Each space corresponds to a different notion of similarity. To reduce this non-identifiability, we make use of

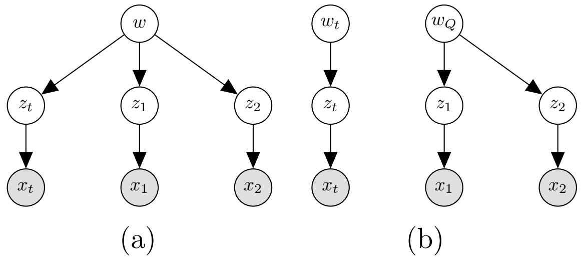


Figure 5-2: **Hypothesis testing with deep generative models:** (a) The **Reasoning Model**, here, depicting the hypothesis that the set $\{x_t, \mathcal{Q} = \{x_1, x_2\}\}$ was generated jointly; (b) the two figures represent the hypothesis that x_t and $\mathcal{Q}$ were generated independently under different realizations of w (the random variable that captures the property shared across datapoints).

supervision. Queries provide extra information in that they reveal which points should be expected to be close together in latent space. We take advantage of this and propose a supervised max-margin learning algorithm for the LRN such that scores given to items in the query are larger than scores unrelated to the query.

We obtain a coupled set of models: in which one model is a deep generative model of the data whilst the other reshapes the latent space of the first and serves to answer queries about similarity judgements between datapoints. We study how the proposed approach can tune the latent space of deep generative models and be used to build new types of models for few-shot learning. We begin in Section 5.2 by motivating the Bayes Factor as a viable tool for computing similarity.

5.2 From representation learning to reasoning

Here, we consider the problem of scoring elements in a set based on how similar they are to a given query. Suppose we are given a dataset $\mathcal{D} = \{x_1, \ldots, x_N\},\ x_i \in \mathbb{R}^n, x_i \in \mathcal{D}$. Then for a query $\mathcal{Q} = \{x_1, \ldots, x_Q\};\ |\mathcal{Q}| = Q$, we wish to assign to each $x_t \in \mathcal{D}$ a score$(x_t, \mathcal{Q})$ that denotes how *similar* x_t is to elements of the query $\mathcal{Q}$.

5.2.1 Data model

A simple way to quantify how similar objects are (here, between $\mathcal{Q}$ and x_t) might be to take the pairwise Euclidian distance between them. For complex, high dimensional data that do not lie on a Euclidian manifold, such a metric may fail to capture interesting regularity between data.

Alternatively, we can use a latent variable model to construct a representation of data. The latent variable then becomes a low-dimensional sufficient statistic for the raw data when quantifying similarity. The simplest latent variable model we will consider has the following generative process: $z \sim p_{\mathbf{dm}}(z)$; $x \sim p_{\mathbf{dm}}(x; f(z;\theta))$ where $p_{\mathbf{dm}}(z)$ is a simple distribution such as $\mathcal{N}(0, I)$. The use of MLPs in the conditional distributions allow the model to fit highly complex data despite the use of a simple prior. When f is parameterized by a Multi-Layer Perceptron (MLP), the resulting model is a deep generative model. We will refer to this model (Kingma & Welling, 2014; Rezende *et al.* , 2014) as the **Data Model** (with probabilities denoted with subscript **dm**).

The generative process assumes datapoints are drawn independently. Using variational inference with an inference network (Hinton *et al.* , 1995) to approximate the posterior distribution, $p_{\mathbf{rm}}(z|x)$, the model can be learned by maximizing a lower bound on the log-likelihood of the data.

$$\begin{aligned}\log p_{\mathbf{dm}}(x;\theta) \geq & \mathop{\mathbb{E}}_{q_{\mathbf{dm}}(z|x;\phi)} \left[\log p_{\mathbf{dm}}(x|z;\theta))\right] \\ & - \mathrm{KL}(\, q_{\mathbf{dm}}(z|x;\phi) || p_{\mathbf{dm}}(z)\,) = \mathcal{L}(x;\theta,\phi),\end{aligned} \tag{5.1}$$

With a Gaussian distribution as the variational approximation: $q_{\mathbf{dm}}(z|x;\phi) \sim \mathcal{N}\left(\mu_\phi(x), \Sigma_\phi(x)\right)$ where $\mu_\phi(x), \Sigma_\phi(x)$ are (differentiable, parametric, with parameters ϕ) functions of the observation x. As before, Eq. 5.1 is differentiable in θ, ϕ (Kingma & Welling, 2014; Rezende *et al.* , 2014) and the model parameters (θ, ϕ) can be learned via gradient ascent on $\mathcal{L}(x;\theta,\phi)$.

With the variational approximation, $q_{\mathbf{dm}}(z|x;\phi)$, to map from data to latent space, would computing overlap in the posterior distributions of points in $\mathcal{Q}$ and x_t suffice to identify similar points? The answer is *sometimes*. While unsupervised learning will tend to put *similar* points together, the notion of similarity encoded in the latent space need not correspond to the notion of similarity required for a task at test time. We require a way to guide the structure of the latent space to be better suited for a task.

5.2.2 Reasoning model

Introducing hierarchy into the generative process is one way to guide the structure of latent variables. In Figure 5-2 (b) is a simple hierarchical model that makes explicit the insight that similar datapoints should have similar latent spaces. It defines the following generative process for a set of similar objects $\mathcal{Q}$: $p_{\mathbf{rm}}(\mathcal{Q}) = \int_w \int_z p_{\mathbf{rm}}(w) \prod_{q=1}^{Q} p_{\mathbf{rm}}(z_q|w) p_{\mathbf{rm}}(x_q|z_q)$. The random variable w defines the context of $\mathcal{Q}$. It may denote the label or class identity of points in $\mathcal{Q}$ but more broadly is a representation of the properties that points in $\mathcal{Q}$ satisfy. For notational convenience and because we can express reasoning about similarity as a probabilistic query in this model, we refer to it as the **Reasoning Model**.

The Neural Statistician (Edwards & Storkey, 2016) uses $\text{KL}(p(w|x_t)||p(w|\mathcal{Q}))$ to quantify the similarity between x_t and $\mathcal{Q}$ in a model similar to the one in Figure 5-2 (b). In this work, we pose the estimation of similarity between objects as hypothesis testing in a hierarchical deep generative model. The conditional independences in Figure 5-2 (b) enforce that x_t is independent of w given z_t, i.e. the per-data-point latent variables serve as a sufficient statistic to quantify comparisons between multiple datapoints. The conditional density $p(x_t|z_t)$ is a map from the representation space to the data while $p(z_t|w)$ dictates how the latent space of a datapoint behaves as a function of property encoded in w.

5.2.3 Bayes factor

To score the similarity between two objects (in this case x_t and set $\mathcal{Q}$) under the Reasoning Model, we turn to the likelihood ratio between the joint distribution of x_t and $\mathcal{Q}$ and the product of their marginals. If x_t and $\mathcal{Q}$ are drawn from the same joint distribution, then there exists a random variable w that governs the distribution of the *latent* spaces $z_t, z_1, \ldots, z_Q$. With slight abuse of notation[1], Figure 5-2 (a) depicts this scenario when $\mathcal{Q} = \{x_1, x_2\}$. If x_t and $\mathcal{Q}$ are not similar, then their latent spaces will have different distributions, and they are children of different realizations of w (see Figure 5-2 (b)). With that in mind, the score function we use to measure similarity is given by (**Bayes Factor**):

$$\frac{p(x_t, \mathcal{Q})}{p(x_t)p(\mathcal{Q})} = \frac{p(x_t|\mathcal{Q})}{p(x_t)} = \text{score}(x_t, \mathcal{Q}) \tag{5.2}$$

[1]We re-use Figure 5-2 to denote both the instantiation of a hypothesis and the generative process

The log-score is the pointwise mutual information (Fano, 1949), a measure of association that is frequently used in applications such as natural language processing (Church & Hanks, 1990). The Bayes Factor normalizes the posterior predictive density of the target point conditioned on the query by the target's marginal likelihood under the model. It also has an information theoretic interpretation. Letting $h(x) = -\log p(x)$ denote the self-information (or surprisal), then $\log \text{score}(x_t, \mathcal{Q}) = h(x_t) - h(x_t|\mathcal{Q})$ intuitively denotes the surprise (quantified in nats or bits) from observing x_t when having already observed $\mathcal{Q}$.

Similarity in Latent Space: Equation 5.2 captures an intuitive notion of similarity but evaluating $p(x_t)$, the marginal density of the target, is typically intractable (except in hierarchical models that lie in the exponential family (Ghahramani & Heller, 2005)). Furthermore, an importance sampling based Monte-Carlo estimator for $p(x_t)$ will involve a high-dimensional integral in the data x_t. We therefore propose the following decomposition of the score function that evaluates the Bayes Factor in the target datapoint's (lower dimensional) latent space:

$$\begin{aligned}
\frac{p_{\mathbf{rm}}(x_t|\mathcal{Q})}{p_{\mathbf{rm}}(x_t)} &= \frac{1}{p_{\mathbf{rm}}(x_t)} \int_{z_t} p_{\mathbf{rm}}(x_t, z_t|\mathcal{Q}) \qquad (5.3)\\
&= \frac{1}{p_{\mathbf{rm}}(x_t)} \int_{z_t} p_{\mathbf{rm}}(x_t|z_t) p_{\mathbf{rm}}(z_t|\mathcal{Q})\\
&= \frac{1}{p_{\mathbf{rm}}(x_t)} \int_{z_t} \frac{p_{\mathbf{rm}}(z_t|x_t) p_{\mathbf{rm}}(x_t)}{p_{\mathbf{rm}}(z_t)} p_{\mathbf{rm}}(z_t|\mathcal{Q})\\
&= \int_{z_t} \underbrace{\frac{p_{\mathbf{rm}}(z_t|x_t)}{p_{\mathbf{rm}}(z_t)}}_{\textbf{Relative Posterior Likelihood}} \underbrace{p_{\mathbf{rm}}(z_t|\mathcal{Q})}_{\textbf{Latent Reasoning Network}}.
\end{aligned}$$

The estimator above formalizes the intuition for comparing points laid out in Section 5.1. The query-conditional posterior-predictive density over the latent space of the target datapoint, $p_{\mathbf{rm}}(z_t|\mathcal{Q})$, reasons about points in the query and represents them as a density in latent space, The **Relative Posterior Likelihood**, $\frac{p_{\mathbf{rm}}(z_t|x_t)}{p_{\mathbf{rm}}(z_t)}$ scores how likely the target point is to have come from the relevant part of latent space.

5.3 Hierarchical models with compound priors

To compute the ratio $\frac{p_{\mathbf{rm}}(z_t|x_t)}{p_{\mathbf{rm}}(z_t)}$, we need to marginalize w_t. However, under certain assumptions about the conditional distributions in the **Reasoning Model**, we will see that approximating this ratio becomes simpler.

Assumption 1. Priors with Compound Distributions

$$\int_w p_{\mathbf{rm}}(w)p_{\mathbf{rm}}(z|w)dw = p_{\mathbf{dm}}(z)$$

Assumption 2. Matching conditional likelihoods

$$p_{\mathbf{rm}}(x|z) = p_{\mathbf{dm}}(x|z)$$

Lemma 5.3.1. *Matching posterior marginals*

$$p_{\boldsymbol{dm}}(z|x) = p_{\boldsymbol{rm}}(z|x)$$

Proof. Follows from Bayes rule and Assumption 1, 2. □

Lemma 5.3.2. *Matching marginal likelihoods*

Under Assumption 1 and 2:

$$p_{\boldsymbol{dm}}(x) = p_{\boldsymbol{rm}}(x)$$

Proof.

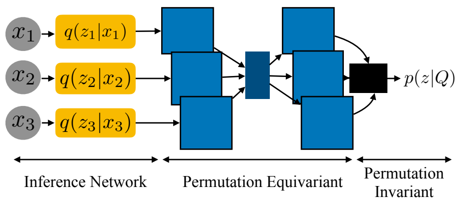
$$\begin{aligned} p_{\mathbf{rm}}(x) &= \int_w \int_z p_{\mathbf{rm}}(w)p_{\mathbf{rm}}(z|w)p_{\mathbf{rm}}(x|z)]dzdw \\ &= \int_z p_{\mathbf{dm}}(z)p_{\mathbf{dm}}(x|z)dz = p_{\mathbf{dm}}(x) \end{aligned}$$

□

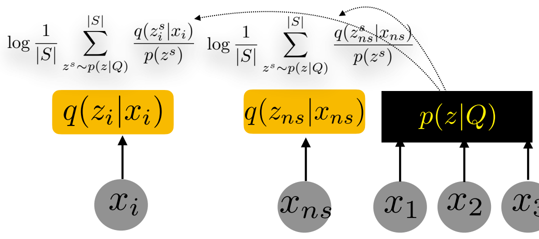
The conditions above state when we can take an instance of the **Data Model** discussed in Section 5.2.1 and transform it into an instance of the **Reasoning Model** in Section 5.2.2 while preserving the marginal likelihood of the data.

This transformation has a few implications. The first is when evaluating the Bayes Factor; if we work in a class of **Reasoning Models** that satisfy Assumption 1, then we can evaluate the **Relative Posterior Likelihood** using the prior and posterior distribution of the associated **Data Model**. With Lemma 5.3.1 and Assumption 1:

$$\frac{p_{\mathbf{rm}}(x_t|\mathcal{Q})}{p_{\mathbf{rm}}(x_t)} = \int_{z_t} \underbrace{\frac{p_{\mathbf{dm}}(z_t|x_t)}{p_{\mathbf{dm}}(z_t)}}_{\textbf{Relative Posterior Likelihood}} \underbrace{p_{\mathbf{rm}}(z_t|\mathcal{Q})}_{\textbf{Latent Reasoning Network}}$$



(a) **Latent Reasoning Network**


$\log \frac{1}{|S|} \sum_{z^s \sim p(z|Q)}^{|S|} \frac{q(z_i^s|x_i)}{p(z^s)}$
$\log \frac{1}{|S|} \sum_{z^s \sim p(z|Q)}^{|S|} \frac{q(z_{ns}^s|x_{ns})}{p(z^s)}$
$q(z_i|x_i)$
$q(z_{ns}|x_{ns})$
$p(z|Q)$
x_i
x_{ns}
x_1
x_2
x_3

(b) **Loss function**

Figure 5-3: **Latent Reasoning Networks (LRN) and loss function:** On the left is a diagrammatic representation of $p_{\mathbf{rm}}(z_t|\mathcal{Q})$. On the right is a depiction of Monte-Carlo sampling (with samples from the LRN) to evaluate Bayes factor. x_i is a point similar to those in the query $\mathcal{Q} = \{x_1, x_2, x_3\}$, while x_{ns} is not. We suppress subscripts in the figure.

where $p_{\mathbf{dm}}(z_t)$ is typically fixed ahead of time (e.g. $\mathcal{N}(0;\mathbb{I})$) and we can do inference for $p_{\mathbf{dm}}(z_t|x_t)$ (or approximate it using the inference network $q_{\mathbf{dm}}(z|x;\phi)$).

The second implication is that part of the **Reasoning Model**, $p_{\mathbf{rm}}(x|z)$, can be learned ahead of time. This gives us the flexibility to *warm-start* the **Reasoning Model** using a *pre-trained* **Data Model** whose $p_{\mathbf{dm}}(z)$ can be expressed according to Assumption 1. In this way, even if we do not know which property will be used to organize datapoints into sets at test time, we can still learn a generic low-dimensional representation of the dataset. We will make use of this when we discuss the learning framework in Section 5.5. For now, what remains is how we can specify $p_{\mathbf{rm}}(w), p_{\mathbf{rm}}(z|w)$ in order to evaluate $p_{\mathbf{rm}}(z_t|\mathcal{Q})$.

5.4 Latent Reasoning Networks

Although $p_{\mathbf{rm}}(z_t|\mathcal{Q}) = \int_w p_{\mathbf{rm}}(z_t|w)p_{\mathbf{rm}}(w|\mathcal{Q})dw$, finding both $p_{\mathbf{rm}}(w)$ and $p_{\mathbf{rm}}(z|w)$ that satisfy Assumption 1 may prove challenging and so we will make use of another computational trick. To evaluate the Bayes Factor we only need a way to sample from $p_{\mathbf{rm}}(z_t|\mathcal{Q})$ i.e. the posterior predictive distribution given the query, of the target's latent representation. Our strategy therefore, will instead be to parameterize and learn $p_{\mathbf{rm}}(z_t|\mathcal{Q})$ directly from data.

Without $p_{\mathbf{rm}}(w)$ and $p_{\mathbf{rm}}(z|w)$, we lose the ability to sample from the Reasoning Model but by amortizing $p_{\mathbf{rm}}(z_t|\mathcal{Q})$ we obtain a fast way to evaluate the Bayes Factor at test time. $p_{\mathbf{rm}}(z_t|\mathcal{Q})$ must *reason* about how the latent spaces of points in $\mathcal{Q}$ are related and parameterize a distribution over the latent space of the target datapoint x_t;

this distribution must characterize the property represented by points in $\mathcal{Q}$. Therefore, we refer to this amortizated, parameteric posterior-predictive distribution as a *Latent Reasoning Network*. Since we do not know the functional form of this distribution we will parameterize it as a non-linear function of the query $\mathcal{Q}$.

To construct the LRN, we require neural architectures capable of operating over sets. We make use of two primitives for such neural architectures proposed by Zaheer *et al.* (2017). These functions operate over sets of vectors $\mathcal{Q} = \{x_1, \dots, x_Q\}, x_q \in \mathbb{R}^n$. We will use the notation $\mathbb{R}^{n \times |\mathcal{Q}|}$ to denote a set of size $|\mathcal{Q}|$ where each element is an n-dimensional vector. We design the LRN, with the following three properties:

A] Parameter Sharing: We share parameters between the inference network of the Data Model and the LRN. A direct consequence of this choice is that the LRN now has the ability to change the way inference is done in the Data model. The first stage of the LRN uses the inference network of the Data Model to map from the set $\mathcal{Q}$ to a set of each point's variational parameters

B] Exchangeability: The output of the LRN must not depend on the order of elements in $\mathcal{Q}$. We achieve this by using the functions proposed by (Zaheer *et al.* , 2017): $g : \mathbb{R}^{n \times |\mathcal{Q}|} \to \mathbb{R}^{m \times |\mathcal{Q}|}$ is a permutation equivariant function that maps from sets of n dimensional vectors to sets of m dimensional vectors while ensuring that if the input elements were permuted, then the output elements would also be permuted identically. The form of g is given by $g(\mathcal{Q}) = \left[\rho\left(W_1^{\text{eq}} x_q + W_2^{\text{eq}}(\sum_{q'} x_{q'})\right)\right]_{q=1}^{|\mathcal{Q}|}$ where $W_1^{\text{eq}} \in \mathbb{R}^{m \times n}$, $W_2^{\text{eq}} \in \mathbb{R}^{m \times n}$ and ρ is an elementwise nonlinearity. We use compositions of the function g in the second stage of the LRN to learn about how the variational parameters between points in $\mathcal{Q}$ relate to one-another and map to a set of intermediate representations.

C] Distributions in latent space: The network must parameterize a valid density in latent space; this is satisfied by construction. To go from the set of intermediate representations to the parameters of $p(z_t|\mathcal{Q})$, we leverage the following permutation invariant function: $f(\mathcal{Q}) = \rho\left(\sum_q (W^{\text{inv}} x_q + b)\right)$, $f : \mathbb{R}^{n \times |\mathcal{Q}|} \to \mathbb{R}^m$ where $W^{\text{inv}} \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$ are linear operators and ρ is an elementwise non-linearity.

With $\mu(\mathcal{Q};\gamma,\phi), \Sigma(\mathcal{Q};\gamma,\phi)$ as parameteric functions of set $\mathcal{Q}$, we can express the probability distribution $p_{\mathbf{rm}}(z_t|\mathcal{Q};\gamma,\phi) = \mathcal{N}(\mu(\mathcal{Q};\gamma,\phi), \Sigma(\mathcal{Q};\gamma,\phi))$. γ denotes the parameters of the permutation equivariant and invariant layers while ϕ represent the parameters shared with $q_{\mathbf{dm}}(z|x;\phi)$. We visualize the LRN in Figure 5-3a.

5.5 Learning

The learning procedure we use is based on a combination of doing unsupervised learning to learn a good representation alongside a supervised max-margin loss to ground the representation for a specific task. We discuss each separately and then highlight how they are combined.

Unsupervised Learning: Since we use **Reasoning Models** that satisfy Assumption 1, 2, we make use of the transformation between the **Data Model** and **Reasoning Model** in Section 5.3. We maximize the likelihood of a given dataset using the lower-bound in Equation 5.1. A consequence of doing variational learning of the Data Model is that we can use $q_{\mathbf{dm}}(z|x;\phi)$ to approximate the Bayes Factor.

Max-Margin Learning: We expect that the Bayes Factor in Equation 5.3 takes a high value when the target point x_t is *similar* to $\mathcal{Q}$ and a low value when x_t is dissimilar to $\mathcal{Q}$. But how do we know what points form $\mathcal{Q}$? This will depend on the test-time task. We assume we are given labels that define the property encompassed in sets of datapoints.

Assumption 3. For L datapoints in $\mathcal{D}$, we have $\mathcal{Y} = \{y_{x_1}, \ldots, y_{x_L}\},\ y_l \in \{1, \ldots, K\}$ where y_{x_i} is the label for x_i that takes one of K unique labels. We define $\mathbb{N}^{\mathcal{Q}}_{x_i} = \{x_k \ s.t.\ y_{x_k} \in \mathcal{Y}\ \&\ y_{x_k} = y_{x_i}\}$, $\mathbb{N}^{\not\mathcal{Q}}_{x_i} = \{x_k \ s.t.\ y_{x_k} \in \mathcal{Y}\ \&\ y_{x_k} \neq y_{x_i}\}$ to be sets of datapoints that have the same label as x_i and those that do not.

We will assume that a point can only have a single label. Here, the labels characterize the property we want to base our similarity judgements on. Therefore, learn the parameters of $p(z_t|\mathcal{Q};\gamma,\phi)$ using the following (supervised) loss function:

$$\mathcal{L}_{\mathbf{mm}}(x;\gamma,\phi) = \mathbb{E}_{\mathcal{Q}\sim\mathbb{N}^{\mathcal{Q}}_{x}}\mathbb{E}_{\mathcal{Q}_{ns}\sim\mathbb{N}^{\not\mathcal{Q}}_{x}}\frac{1}{|\mathcal{Q}_{ns}|}\sum_{x_{ns}\in\mathcal{Q}_{ns}}\max(\log\operatorname{score}(x_{ns},\mathcal{Q}) - \log\operatorname{score}(x,\mathcal{Q}) + \Delta, 0). \tag{5.4}$$

The loss function maximizes the difference between the log-Bayes Factor for points that lie within the set $\mathcal{Q}$ and those that do not (they lie in $\mathcal{Q}_{ns}$). The $\log\operatorname{score}(x,\mathcal{Q})$, in Equation 5.3, is evaluated via Monte-Carlo sampling and the log-sum-exp trick. The expectation is differentiable with respect to γ,ϕ via the reparameterization trick (Kingma & Welling, 2014; Rezende *et al.* , 2014). For the margin Δ we use the mean-squared-error between the the posterior means of x, x_{ns}. We provide a visual depiction of how the loss is evaluated using the LRN in Figure 5-3.

Combined Loss: With the unsupervised learning objective for the **Data Model** and the supervised max-margin loss function (Equation 5.4) for the LRN, we obtain the following loss to jointly learn θ, ϕ, γ:

$$\begin{aligned}\min_{\theta.\phi.\gamma} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{C+1} \left[-\mathcal{L}(x_i; \theta, \phi)\right] + \\ \frac{C}{C+1} \mathbb{I}[x_i \in \mathcal{Y}] \mathcal{L}_{\mathbf{mm}}(x_i; \gamma, \phi)\end{aligned} \tag{5.5}$$

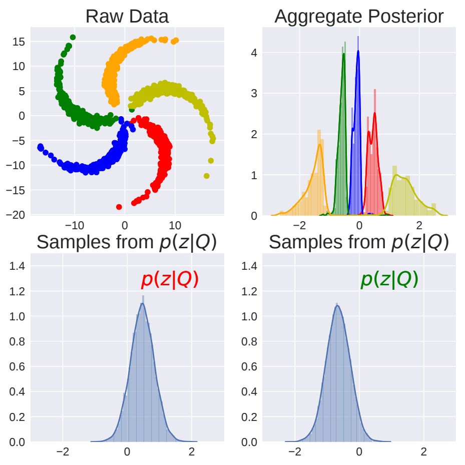
where C is a regularization constant that trades off between the supervised and the unsupervised loss. The unsupervised loss learns a representation space constrained to lie close to the prior while explaining the data under the generative model. The max-margin loss modifies this representation space so that dissimilar points are kept apart. Note that Equation 5.5 is no longer a valid bound on the marginal likelihood of the training set (for $C > 0$).

5.6 Evaluation

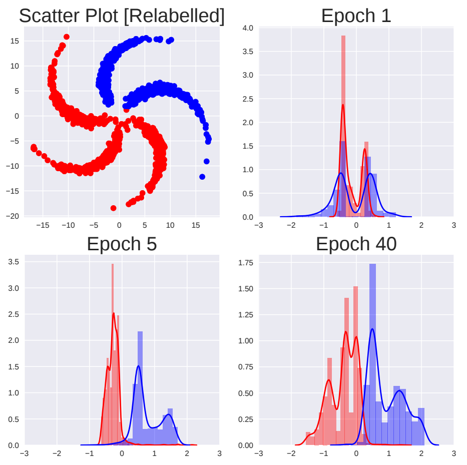
The goal of this section is threefold: (1) to study whether $p_{\mathbf{rm}}(z|\mathcal{Q})$is learnable from data using the max-margin learning objective – we expect this to be challenging since we learn the parameters of a model that is itself used to evaluate the the score function in the loss; (2) studying the role of parameter sharing between the inference network and the LRN – i.e. whether the latter can change the former in adversarial scenarios; and (3) studying the utility of the framework for few-shot learning.

We will release code in Keras (Chollet *et al.* , 2015). Appendix A contains detailed information on the neural architectures of the deep generative models used in the evaluation. We learn parameters with a learning rate of 0.00005 and adaptive momentum updates given by ADAM (Kingma & Ba, 2014). We set the value C separately for each experiment. When there is a task to be solved, C can be set using the validation data. When using a pre-trained Data Model, we found it useful to anneal C from a higher to a lower value so that the task-specific supervised term can overcome (potentially) suboptimal latent spaces learned from unsupervised data. We use the following datasets for our study:

Synthetic Pinwheel: A synthetic dataset of two-dimensional points arranged on a



(a) **Data and aggregate posterior:** (Top Left) Pinwheel data; (Top right) Aggregate posterior density of a learned (unconditional) deep generative model coloured by class membership. (Bottom row) Sampling from $p_{\textbf{rm}}(z|\mathcal{Q})$ where the colour denotes the class membership of points in $\mathcal{Q}$.



(b) **Learning dynamics:** (Top left) Visualization of *adversarially labelled data* (relative to the learned aggregate posterior in Figure 5-4a (top right)). The remaining plots are class coloured visualizations of the aggregate posterior (during training) while allowing the LRN to fine-tune the latent space of the DGM.

Figure 5-4: **Qualitative evaluation on pinwheel data:** Studying how the latent space of the data changes over the course of fine-tuning on the synthetic, pinwheel dataset.

pinwheel taken from the work of Johnson *et al.* (2016). We depict the raw data in Figure 5-4a. The dataset is created with five labels.

MNIST digits: 50000 black and white images of handwritten digits (LeCun, 1998).

MiniImagenet: A subsampled set of images taken from the Imagenet repository setup for the task of k-shot learning by Vinyals *et al.* (2016). We use the train-validate-test split kindly provided by Ravi & Larochelle (2016).

5.6.1 Learning $p(z|\mathcal{Q})$

As a sanity check, we begin by first training a deep generative model (without labels and using a one-dimensional latent space) on the Pinwheel dataset. We visualize the raw-data and learned aggregate posterior $\sum_x q_{\textbf{dm}}(z|x;\phi)$ in Figure 5-4a (top row). We see that the unsupervised learning alone induces class separation in the aggregate posterior distribution. Using the learned model, we hold fixed parameters: θ, ϕ and learn the parameters γ of the LRN using the loss function in 5.4 with $C = 2000$. We form a kernel density estimate of samples from $p_{\mathbf{rm}}(z|\mathcal{Q})$ using randomly constructed sets of points derived from the red and green clusters. In Figure 5-4a (bottom row), we see that samples from the LRN correspond to regions of the latent space associated with $\mathcal{Q}$. On synthetic examples, the LRN finds regions of latent space corresponding to points from a query $\mathcal{Q}$.

5.6.2 Changing inductive biases at test-time

Previously, we worked with a model where the structure of the latent space (as seen in the aggregate posterior distribution) formed during unsupervised learning coincided with how points were grouped into sets. Here, we study what happens where the notion of which points are similar changes at test time. We relabel the pinwheel dataset so that the yellow and orange points form one class while the green, red and blue form the other (see Figure 5-4b, top left). This corresponds to an *adversarial* labelling of the data since we use a deep generative model in which points in the same class are far apart in the learned latent space. If we keep θ, ϕ fixed then $p_{\mathbf{rm}}(z|\mathcal{Q})$ (whose output is parameterized as a unimodal Gaussian distribution) cannot capture the relevant subspace.

We have two choices here; we can either consider richer parameterizations for $p_{\mathbf{rm}}(z|\mathcal{Q})$ that are capable of capturing multi-modal structure in the latent space using techniques proposed by Rezende & Mohamed (2015), or we can instead allow the $p_{\mathbf{rm}}(z|\mathcal{Q})$ to *change* the underlying latent space of the generative model by back-propagating through the parameters of the inference network. Here, we opt for the latter, though the former is an avenue for future work.

We minimize Equation 5.5 while annealing the constant C from $1000 \rightarrow 1$ linearly through the course of training. To gain insight into the learning dynamics of the LRN during training, we visualize the aggregate posterior of the generative model (via the fine-tuned inference network) in Figure 5-4b through the course of training. The role of this adversarial scenario is to highlight two important points (1) unsupervised learning is typically unidentifiable and may not learn a representation appropriate to all tasks and (2) learning with the latent reasoning network can overcome a suboptimal (relative to the task at hand) representation and transform it to a more suitable one.

5.6.3 Modeling high-dimensional data

Inducing diversity in latent space: Moving beyond low-dimensional data, we study learning LRNs on MNIST digits. We use a Data Model with a two-dimensional latent space for this experiment. We begin by training the model in a fully unsupervised manner and visualize the learned latent space in the form of the aggregate posterior (Figure 5-5a [left]). Although there is some class separability, we find that the unsupervised learning algorithm concentrates much of the probability mass together.

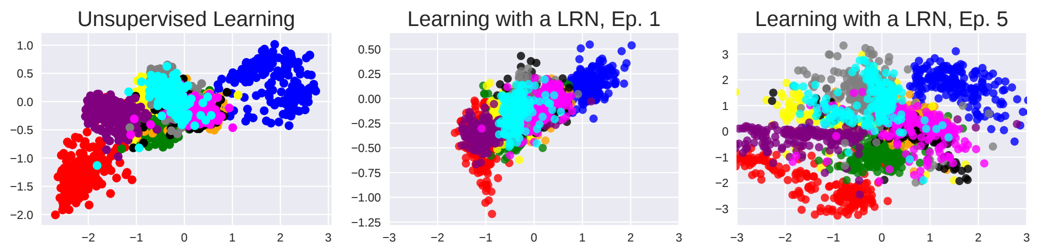
We re-learn the same model with the loss in Equation 5.5 where C is set to 3000 (and annealed to 1). We again visualize the new aggregate posterior distribution of the Data Model in Figure 5-5a (middle and right). When learning with Equation 5.5, the inference network uses more of the latent space in the model because the max-margin loss pushes points in different classes further apart.

Qualitative Analysis of MNIST digits: To validate our method, we provide visualizations on the MNIST dataset. We select a handful of labelled examples $\mathcal{Q}$ (Figure 5-5b, left) and visualize both their posterior means and samples from $p(z|\mathcal{Q})$ (Figure 5-5b, middle). Then, for each sample from $p_{\mathbf{rm}}(z|\mathcal{Q})$, we evaluate the fine-tuned $p_{\mathbf{dm}}(x|z)$ and visualize the images in Figure 5-5b (right). We see that the generative model fine-tuned with the learning algorithm retains its ability to generate meaningful samples.

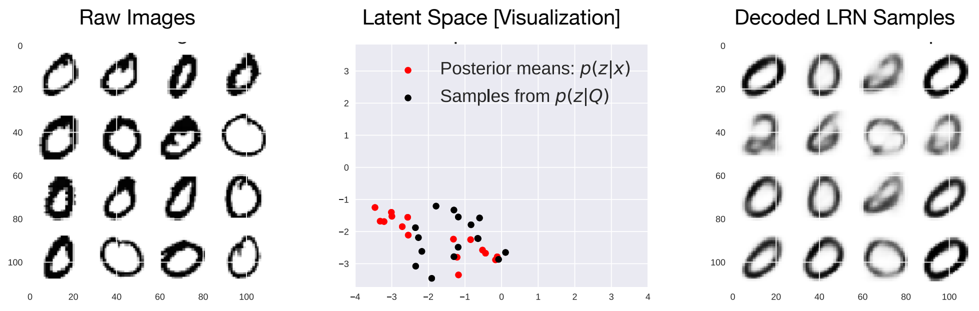
5.6.4 Few-shot learning with the Bayes factor

The task of k-shot learning is to identify the class an object came from given a single example from 5 other classes (1-shot, 5-way). In the 5-shot, 5-way task. there are 5 examples provided from each of the 5 potential classes. We use an LRN with a deep-discriminative model to obtain near state of the art performance in few-shot learning on the MiniImagenet dataset.

Following (Bauer *et al.* , 2017), who show that discriminative models alone form powerful baselines for this task on this dataset, we pretrain an 18 layer Resnet (He *et al.* , 2016) convolutional neural network to predict class labels at training time. We use early stopping on a validation set based on the nearest neighbor performance of the learned embeddings (obtained from the final layer of the ResNet) to identify the best model. Building a good generative model of the images in MiniImagenet is difficult and so instead, we use the *fixed* embeddings as a 256 dimensional proxy



(a) **Training dynamics for MNIST:** Aggregate (two-dimensional) posterior of deep generative model of MNIST (coloured by label). The left corresponds to a model trained with unsupervised data only; the middle & right show the aggregate posteriors for a model fine-tuned using Equation 5.5.



(b) **Test-time evaluation of LRN on MNIST:** On the left are a set of query points $\mathcal{Q}$ drawn from the same class, in the middle, we visualize samples from $q_{\mathbf{dm}}(z|x;\phi)$ for each of the points and $p_{\mathbf{rm}}(z|\mathcal{Q})$. On the right is the output of the fine-tuned conditional density $p_{\mathbf{dm}}(x|z)$ for samples drawn from $p_{\mathbf{rm}}(z|\mathcal{Q})$.

Figure 5-5: **Qualitative evaluation on MNIST:** Studying the effect of fine-tuning the latent space of the data model on MNIST.

for each image. We initialize $q_{\mathbf{dm}}(z|x;\phi)$ with the pretrained Resnet and set up a deep generative model to maximize the likelihood of the fixed embeddings (after discriminative pre-training).

For this task, when comparing to the many different approaches proposed, it is challenging to control for both the depth of the encoder that parameterizes the representation and the various algorithmic approach used to tackle the problem using the representation. Therefore, our two take-aways from Table 5.1 are: (1) on the 1 shot and 5 shot task, we outperform a strong nearest neighbors baseline created using fixed (but learned) embeddings suggesting that our algorithmic approach bears promise for this task and (2) the method is competitive with other state of the art approaches.

Table 5.1: **5-way MiniImagenet task:** Accuracies for few-shot learning on the MiniImagenet task. The first row contains our method where higher is better.

Model	1-shot	5-shot
Nearest Neighbor	51.4 ± 0.08	67.5 ± 0.08
Ours [Resnet18 encoder]	53.5 ± 0.08	68.8 ± 0.08
Matching Networks (Vinyals *et al.* , 2016)	46.6	60.0
MAML (Finn *et al.* , 2017)	48.7	63.1
Prototypical Nets (Snell *et al.* , 2017)	49.4	68.2
MetaNets (Munkhdalai & Yu, 2017)	49.2	*
TCML (Mishra *et al.* , 2018)	56.7	68.9

5.7 Related work

Max Margin Learning: Max margin parameter estimation has been widely used in machine learning (e.g. in structural SVMs (Yu & Joachims, 2009) and in discriminative Markov networks (Zhu & Xing, 2009)). (Li *et al.* , 2015a) give a doubly stochastic subgradient algorithm for regularized maximum likelihood estimation when dealing with max-margin posterior constraints.

(Zaheer *et al.* , 2017) experiment with max-margin learning using a variant of the DeepSets model to predict a scalar score conditioned on a set. While (Zaheer *et al.* , 2017) cite the estimator in (Ghahramani & Heller, 2005) as motivation for their model, they do not explicitly use, parameterize, or differentiate through the Bayes Factor in a *generative* model of data.

Inductive Transfer and Metric Learning: Lake *et al.* (2013) use probabilistic inference in a hierarchical model to classify unseen examples by their probability of being in a new class. Instead of the Bayes Factor, they use the posterior predictive obtained via the use of a MCMC algorithm to score target points relative to a query. (Ghahramani & Heller, 2005) evaluate the Bayes factor analytically in exponential family distributions. What we gain in for sacrificing tractability is the ability to work within a richer class of models. Though not motivated within the context of a hierarchical model, (Engel *et al.* , 2018) use an adversarial loss to recognize regions of latent space that correspond to points with a specified class.

Vinyals *et al.* (2016) learn a parametric K-nearest neighbor classifiers to predict whether a target item is within the same class as k-others. (Snell *et al.* , 2017) associate a point with a prototype within a set and use it to answer whether an object is in the same class as others. (Bauer *et al.* , 2017) show that the features from a ResNet (He *et al.* , 2016) model already provide a powerful feature representation in which a k-nearest neighbor classifier performs remarkably well. The Neural Statistician (Edwards & Storkey, 2016) learns a model similar [2] to the **Reasoning Model** in Figure 5-2 (b) by maximizing the likelihood of sets $\mathcal{Q}$. Their method does not use the Bayes Factor to score items; it also does not permit easy initialization with pre-trained Data Models since the full model is trained with queries.

We tune the latent space of a deep generative model to enhance class separability for test time tasks. By contrast, meta learning algorithms learn to tune the parameters of an algorithm or a model. (Finn *et al.* , 2017) prime the parameters of a neural network to have high accuracy at test time using second order gradient information.

Our work has close parallels with metric-learning; here the metric learned lies in the latent space of a deep generative model. (Bar-Hillel *et al.* , 2005) proposed Relevant Component Analysis, an optimization problem that jointly performs (linear) dimensionality reduction and learns a Mahalanobis metric using queries.

[2] Their model does not enforce the conditional independence statement $x_t \perp\!\!\!\perp \mathcal{Q}|z_t$

5.8 Discussion

We seek good, task-specific inductive biases to quantify how similar a point is to a set. We give new theoretical and practical constructs towards this goal. We break up the problem into two parts: learn a good representation and tune the learned representation for a specific notion of similarity. Using the latent space in a deep generative model as our representation, we use the Bayes Factor to quantify similarity.

We derive conditions under which there exists an equivalence between a generative model where data are generated independently and a hierarchical model that jointly generates sets of (similar) points. Using this insight, we derive a differentiable estimator for the Bayes Factor; the estimator poses the comparison between a point and a set as overlap in latent space. With the Bayes Factor as a differentiable scoring mechanism, we give a max-margin learning algorithm capable of changing the inductive bias of a (potentially pre-trained) deep generative model. To evaluate the Bayes Factor, we propose a neural architecture for a *latent reasoning network*: a set conditional density that amortizes the posterior predictive distribution of a hierarchical model.

Our approach has limitations. By directly parameterizing the posterior predictive density, and not the prior $p_{\mathbf{rm}}(w)$ and conditional $p_{\mathbf{rm}}(z|w)$, we lose the ability to sample points from the hierarchical generative model. Working with a set of models in which Assumption 1 holds may implicitly only find posterior predictive densities under relatively simple model families of $p_{\mathbf{rm}}(w)$ and $p_{\mathbf{rm}}(z|w)$. Finally, enforcing that property identity in w is conditionally independent of the data x, given the representation z, may make for a challenging learning problem – z has to represent both the property and variability in the property conditional distribution of the data.

An avenue of future work is leveraging vast amounts of unlabeled data for representation learning informed by a small amount of supervision to guide either during learning, or after learning, the structured of the learned space. Yet another interesting direction would be to learn LRNs that parameterize distributions over hierarchies of latent variables.

Chapter 6

Deep Markov Models

In the previous three chapters, we described algorithms for learning and prediction with deep generative models of static, high-dimensional data. However, one of the key challenges we outline in Chapter 1 is the temporal nature in which observational clinical data manifests. In this chapter we introduce Deep Markov Models (DMMs), a deep generative model of sequential data. DMMs are a Gaussian state space model wherein the conditional probabilities are parameterized by deep neural networks. We derive an efficient learning algorithm for this model and showcase its flexibility in unsupervised learning on a wide variety of datasets including a cohort of diabetic patients.

Gaussian state space models have been used for decades as generative models of sequential data. They admit an intuitive probabilistic interpretation, have a simple functional form, and enjoy widespread adoption. We introduce a unified algorithm to efficiently learn a broad class of linear and non-linear state space models, including variants where the emission and transition distributions are modeled by deep neural networks. Our learning algorithm simultaneously learns a compiled inference network and the generative model, leveraging a structured variational approximation parameterized by recurrent neural networks to mimic the posterior distribution. We apply the learning algorithm to both synthetic and real-world datasets, demonstrating its scalability and versatility. We find that using the structured approximation to the posterior results in models with significantly higher held-out likelihood.

6.1 Introduction

Models of sequence data such as hidden Markov models (HMMs) and recurrent neural networks (RNNs) are widely used in machine translation, speech recognition, and computational biology. Linear and non-linear Gaussian state space models (GSSMs, Fig. 6-1) are used in applications including robotic planning and missile tracking. However, despite huge progress over the last decade, efficient learning of non-linear models from complex high dimensional time-series remains a major challenge. Our paper proposes a unified learning algorithm for a broad class of GSSMs, and we introduce an inference procedure that scales easily to high dimensional data, compiling approximate (and where feasible, exact) inference into the parameters of a neural network.

In engineering and control, the parametric form of the GSSM model is often known, with typically a few specific parameters that need to be fit to data. The most commonly used approaches for these types of learning and inference problems are often computationally demanding, e.g. dual extended Kalman filter (Wan & Nelson, 1997), expectation maximization (Briegel & Tresp, 1999; Ghahramani & Roweis, 1999) or particle filters (Schön *et al.* , 2011). Our compiled inference algorithm can easily deal with high-dimensions both in the observed and the latent spaces, without compromising the quality of inference and learning.

When the parametric form of the model is unknown, we propose learning *Deep Markov Models* (DMM), a class of generative models where linear emission and transition distributions are replaced with complex multi-layer perceptrons (MLPs). These are GSSMs that retain the Markovian structure of HMMs, but leverage the representational power of deep neural networks to model complex high dimensional data. If one augments a DMM model such as the one presented in Fig. 6-1 with edges from the observations x_t to the latent states of the following time step z_{t+1}, then the DMM can be seen to be similar to, though more restrictive than, stochastic RNNs (Bayer & Osendorfer, 2014) and variational RNNs (Chung *et al.* , 2015).

Our learning algorithm performs stochastic gradient ascent on a variational lower bound of the likelihood. Instead of introducing variational parameters for each data point, we *compile* the inference procedure at the same time as learning the generative model. This idea was originally used in the wake-sleep algorithm for unsupervised learning (Hinton *et al.* , 1995), and has since led to state-of-the-art results for unsupervised learning of deep generative models (Kingma & Welling, 2014; Mnih & Gregor, 2014;

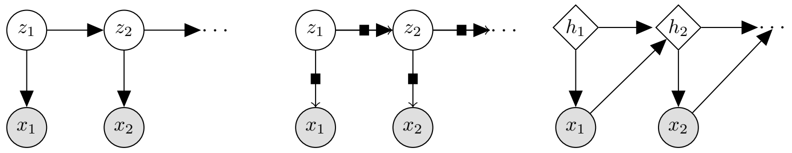


Figure 6-1: **Generative Models of Sequential Data:** (**Top Left**) Hidden Markov Model (HMM), (**Top Right**) Deep Markov Model (DMM) ■ denotes the neural networks used in DMMs for the emission and transition functions. (**Bottom**) Recurrent Neural Network (RNN), ◊ denotes a deterministic intermediate representation. Code for learning DMMs and reproducing our results may be found at: `github.com/clinicalml/structuredinference`

Rezende *et al.* , 2014).

Specifically, we introduce a new family of *structured inference networks*, parameterized by recurrent neural networks, and evaluate their effectiveness in three scenarios: (1) when the generative model is known and fixed, (2) in parameter estimation when the functional form of the model is known and (3) for learning deep Markov models. By looking at the structure of the true posterior, we show both theoretically and empirically that inference for a latent state should be performed using information *from its future*, as opposed to recent work which performed inference using only information from the past Chung *et al.* (2015); Gan *et al.* (2015); Gregor *et al.* (2015), and that a structured variational approximation outperforms mean-field based approximations. Our approach may easily be adapted to learning more general generative models, for example models with edges from observations to latent states.

Finally, we learn a Deep Markov Model on a polyphonic music dataset and on a dataset of electronic health records (a complex high dimensional setting with missing data). We use the model learned on health records to ask queries such as "what would have happened to patients had they not received treatment", and show that our model correctly identifies the way certain medications affect a patient's health.

Related Work: Learning GSSMs with MLPs for the transition distribution was considered by Raiko & Tornio (2009). They approximate the posterior with non-linear dynamic factor analysis Valpola & Karhunen (2002), which scales quadratically with the observed dimension and is impractical for large-scale learning.

Recent work has considered variational learning of time-series data using structured inference or recognition networks. Archer *et al.* propose using a Gaussian approximation to the posterior distribution with a block-tridiagonal inverse covariance. Johnson *et al.* use a conditional random field as the inference network for time-series models. Concurrent to our own work, Fraccaro *et al.* also learn sequential generative models using structured inference networks parameterized by recurrent neural networks.

Bayer & Osendorfer and Fabius & van Amersfoort create a stochastic variant of RNNs by making the hidden state of the RNN at every time step be a function of independently sampled latent variables. Chung *et al.* apply a similar model to speech data, sharing parameters between the RNNs for the generative model and the inference network. Gan *et al.* learn a model with discrete random variables, using a structured inference network that only considers information from the past, similar to Chung *et al.* and Gregor *et al.* 's models. In contrast to these works, we use information from the future within a structured inference network, which we show to be preferable both theoretically and practically. Additionally, we systematically evaluate the impact of the different variational approximations on learning.

Watter *et al.* construct a first-order Markov model using inference networks. However, their learning algorithm is based on data tuples over consecutive time steps. This makes the strong assumption that the posterior distribution can be recovered based on observations at the current and next time-step. As we show, for generative models like the one in Fig. 6-1, the posterior distribution at any time step is a function of *all* future (and past) observations.

6.2 Setup

Gaussian State Space Models: We consider both inference and learning in a class of latent variable models given by: We denote by z_t a vector valued latent variable and by x_t a vector valued observation. A sequence of such latent variables and observations is denoted $\vec{z}, \vec{x}$ respectively.

$$z_t \sim \mathcal{N}(G_\alpha(z_{t-1}, \Delta_t), S_\beta(z_{t-1}, \Delta_t)) \qquad \textit{(Transition)} \qquad (6.1)$$

$$x_t \sim \Pi(F_\kappa(z_t)) \qquad \textit{(Emission)} \qquad (6.2)$$

We assume that the distribution of the latent states is a multivariate Gaussian with a mean and covariance which are differentiable functions of the previous latent state and Δ_t (the time elapsed of time between $t-1$ and t). The multivariate observations x_t are distributed according to a distribution Π (e.g., independent Bernoullis if the data is binary) whose parameters are a function of the corresponding latent state z_t. Collectively, we denote by $\theta = \{\alpha, \beta, \kappa\}$ the parameters of the generative model.

Eq. 6.1 subsumes a large family of linear and non-linear Gaussian state space models. For example, by setting $G_\alpha(z_{t-1}) = G_t z_{t-1}, S_\beta = \Sigma_t, F_\kappa = F_t z_t$, where G_t, Σ_t and F_t are matrices, we obtain linear state space models. The functional forms and initial parameters for $G_\alpha, S_\beta, F_\kappa$ may be pre-specified.

Variational Learning: Using recent advances in variational inference we optimize a variational lower bound on the data log-likelihood. We will make use of an *inference network* or *recognition network* Hinton *et al.* (1995); Kingma & Welling (2014); Mnih & Gregor (2014); Rezende *et al.* (2014), a neural network which approximates the intractable posterior. This is a parametric conditional distribution that is optimized to perform inference. Throughout this paper we will use θ to denote the parameters of the generative model, and ϕ to denote the parameters of the inference network.

For the remainder of this section, we consider learning in a Bayesian network whose joint distribution factorizes as: $p(x, z) = p_\theta(z)p_\theta(x|z)$. The posterior distribution $p_\theta(z|x)$ is typically intractable. Using the well-known variational principle, we posit an approximate posterior distribution $q_\phi(z|x)$ to obtain the following lower bound on the marginal likelihood:

$$\log p_\theta(x) \geq \mathop{\mathbb{E}}_{q_\phi(z|x)} [\log p_\theta(x|z)] - \mathrm{KL}(\, q_\phi(z|x)||p_\theta(z)\,), \tag{6.3}$$

where the inequality is by Jensen's inequality. Kingma & Welling; Rezende *et al.* use a neural net (with parameters ϕ) to parameterize q_ϕ. The challenge in the resulting optimization problem is that the lower bound in Eq. 6.3 includes an expectation w.r.t. q_ϕ, which implicitly depends on the network parameters ϕ. When using a Gaussian variational approximation $q_\phi(z|x) \sim \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$, where $\mu_\phi(x), \Sigma_\phi(x)$ are parametric functions of the observation x, this difficulty is overcome by using *stochastic backpropagation*: a simple transformation allows one to obtain unbiased Monte Carlo estimates of the gradients of $\mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)]$ with respect to ϕ. The KL term in Eq. 6.3 can be estimated similarly since it is also an expectation. When the prior $p_\theta(z)$ is normally distributed, the KL and its gradients may be obtained analytically.

6.3 A factorized variational lower bound

We leverage stochastic backpropagation to learn generative models given by Eq. 6.1, corresponding to the graphical model in Fig. 6-1. Our insight is that for the purpose of inference, we can use the Markov properties of the generative model to guide us in deriving a structured approximation to the posterior. Specifically, the posterior factorizes as:

$$p(\vec{z}|\vec{x}) = p(z_1|\vec{x}) \prod_{t=2}^{T} p(z_t|z_{t-1}, x_t, \ldots, x_T). \tag{6.4}$$

To see this, use the independence statements implied by the graphical model in Fig. 6-1 to note that $p(\vec{z}|\vec{x})$, the true posterior, factorizes as:

$$p(\vec{z}|\vec{x}) = p(z_1|\vec{x}) \prod_{t=2}^{T} p(z_t|z_{t-1}, \vec{x})$$

Now, we notice that $z_t \perp\!\!\!\perp x_1, \ldots, x_{t-1}|z_{t-1}$, yielding the desired result. The significance of Eq. 6.4 is that it yields insight into the structure of the exact posterior for the class of models laid out in Fig. 6-1.

We directly mimic the structure of the posterior with the following factorization of the variational approximation:

$$\begin{aligned} q_\phi(\vec{z}|\vec{x}) &= q_\phi(z_1|x_1, \ldots, x_T) \prod_{t=2}^{T} q_\phi(z_t|z_{t-1}, x_t, \ldots, x_T) \\ \text{s.t.} \quad & q_\phi(z_t|z_{t-1}, x_t, \ldots, x_T) \sim \\ & \mathcal{N}\left(\mu_\phi(z_{t-1}, x_t, \ldots, x_T), \Sigma_\phi(z_{t-1}, x_t, \ldots, x_T)\right) \end{aligned} \tag{6.5}$$

where μ_ϕ and Σ_ϕ are functions parameterized by neural nets. Although q_ϕ has the option to condition on all information across time, Eq. 6.4 suggests that in fact it suffices to condition on information from the future and the previous latent state. The previous latent state serves as a summary statistic for information from the past.

Exact Inference: We can match the factorization of the true posterior using the inference network but using a Gaussian variational approximation for the approximate posterior over each latent variable (as we do) limits the expressivity of the inferential model, except for the case of linear dynamical systems where the posterior distribution is Normally distributed. However, one could augment our proposed inference network with recent innovations that improve the variational approximation to allow for multi-modality Rezende & Mohamed (2015); Tran *et al.* (2016). Such modifications could yield black-box methods for exact inference in time-series models, which we leave for future work.

Deriving a variational lower bound: For a generative model (with parameters

θ) and an inference network (with parameters ϕ), we are interested in $\max_\theta \log p_\theta(\vec{x})$. For ease of exposition, we instantiate the derivation of the variational bound for a single data point $\vec{x}$ though we learn θ, ϕ from a corpus.

The lower bound in Eq. 6.3 has an analytic form of the KL term only for the simplest of transition models G_α, S_β between z_{t-1} and z_t (Eq. 6.1). One could estimate the gradient of the KL term by sampling from the variational model, but that results in high variance estimates and gradients. We use a different factorization of the KL term (obtained by using the prior distribution over latent variables), leading to the variational lower bound we use as our objective function:

$$\mathcal{L}(\vec{x};(\theta,\phi)) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{q_\phi(z_t|\vec{x})} [\log p_\theta(x_t|z_t)] \\ - \mathrm{KL}(q_\phi(z_1|\vec{x})||p_\theta(z_1)) - \sum_{t=2}^{T} \mathop{\mathbb{E}}_{q_\phi(z_{t-1}|\vec{x})} [\mathrm{KL}(q_\phi(z_t|z_{t-1},\vec{x})||p_\theta(z_t|z_{t-1}))]. \tag{6.6}$$

The key point is the resulting objective function has more stable analytic gradients. Without the factorization of the KL divergence in Eq. 6.6, we would have to estimate $\mathrm{KL}(q(\vec{z}|\vec{x})||p(\vec{z}))$ via Monte-Carlo sampling, since it has no analytic form. In contrast, in Eq. 6.6 the individual KL terms *do* have analytic forms. Section 6.3.1 simplifies the lower bound we use during learning while Section 6.3.2 derives the analytic forms for the KL divergence term in the simplification.

6.3.1 Simplifying the lower bounds

We can derive the bound on the likelihood $\mathcal{L}(\vec{x};(\theta,\phi))$ as follows:

$$\log p_\theta(\vec{x}) \geq \int_{\vec{z}} q_\phi(\vec{z}|\vec{x}) \log \frac{p_\theta(\vec{z}) p_\theta(\vec{x}|\vec{z})}{q_\phi(\vec{z}|\vec{x})} d\vec{z} = \mathop{\mathbb{E}}_{q_\phi(\vec{z}|\vec{x})} [\log p_\theta(\vec{x}|\vec{z})] - \mathrm{KL}(q_\phi(\vec{z}|\vec{x})||p_\theta(\vec{z}))$$

$$(\textit{ Using } x_t \perp\!\!\!\perp x_{\neg t}|z_t \text{ })$$

$$= \sum_{t=1}^{T} \mathop{\mathbb{E}}_{q_\phi(z_t|\vec{x})} [\log p_\theta(x_t|z_t)] - \mathrm{KL}(q_\phi(\vec{z}|\vec{x})||p_\theta(\vec{z})) = \mathcal{L}(\vec{x};(\theta,\phi)) \tag{6.7}$$

In the following we omit the dependence of q on $\vec{x}$, and omit the subscript ϕ. We can show that the KL divergence between the approximation to the posterior and the prior simplifies as:

$$\mathrm{KL}(q(z_1,\dots,z_T)||p(z_1,\dots,z_T)) = \int_{z_1}\dots\int_{z_T} q(z_1)\dots q(z_T|z_{T-1})\log\frac{p(z_1,\dots,z_T)}{q(z_1)..q(z_T|z_{T-1})}$$

(Factorization of the variational distribution)

$$= \int_{z_1}\dots\int_{z_T} q(z_1)\dots q(z_T|z_{T-1})\log\frac{p(z_1)p(z_2|z_1)\dots p(z_T|z_{T-1})}{q(z_1)\dots q(z_T|z_{T-1})}$$

(Factorization of the prior)

$$= \int_{z_1}\dots\int_{z_T} q(z_1)\dots q(z_T|z_{T-1})\log\frac{p(z_1)}{q(z_1)}+$$

$$\sum_{t=2}^{T}\int_{z_1}\dots\int_{z_T} q(z_1)\dots q(z_T|z_{T-1})\log\frac{p(z_t|z_{t-1})}{q(z_t|z_{t-1})}$$

$$= \int_{z_1} q(z_1)\log\frac{p(z_1)}{q(z_1)} + \sum_{t=2}^{T}\int_{z_{t-1}}\int_{z_t} q(z_t)\log\frac{p(z_t|z_{t-1})}{q(z_t|z_{t-1})}$$

(Each expectation over z_t is constant for $t \notin \{t, t-1\}$)

$$= \mathrm{KL}(q(z_1)||p(z_1)) + \sum_{t=2}^{T}\mathop{\mathbb{E}}_{q(z_{t-1})}[\mathrm{KL}(q(z_t|z_{t-1})||p(z_t|z_{t-1}))]$$

For evaluating the marginal likelihood on the test set, we can use the following Monte-Carlo estimate:

$$p(\vec{x}) \approxeq \frac{1}{S}\sum_{s=1}^{S}\frac{p(\vec{x}|\vec{z}^{(s)})p(\vec{z}^{(s)})}{q(\vec{z}^{(s)}|\vec{x})} \quad \vec{z}^{(s)} \sim q(\vec{z}|\vec{x}) \tag{6.8}$$

This may be derived in a manner akin to the one depicted in Appendix E in Rezende *et al.* (2014) or Appendix D in Kingma & Welling (2014).

The log likelihood on the test set is computed using:

$$\log p(\vec{x}) \approxeq \log\frac{1}{S}\sum_{s=1}^{S}\exp\log\left[\frac{p(\vec{x}|\vec{z}^{(s)})p(\vec{z}^{(s)})}{q(\vec{z}^{(s)}|\vec{x})}\right] \tag{6.9}$$

Eq. 6.9 may be computed in a numerically stable manner using the log-sum-exp trick.

6.3.2 Analytic forms of the KL divergence

Maximum likelihood learning requires us to compute:

$$\begin{aligned}&\mathrm{KL}(q(z_1,\dots,z_T)||p(z_1,\dots,z_T))\\&=\mathrm{KL}(q(z_1)||p(z_1))+\sum_{t=2}^{T-1}\mathop{\mathbb{E}}_{q(z_{t-1})}[\mathrm{KL}(q(z_t|q_{t-1})||p(z_t|z_{t-1}))]\end{aligned}\tag{6.10}$$

The KL divergence between two multivariate Gaussians q, p with respective means and covariances $\mu_q, \Sigma_q, \mu_p, \Sigma_p$ can be written as:

$$\mathrm{KL}(q||p)=\frac{1}{2}(\underbrace{\log\frac{|\Sigma_p|}{|\Sigma_q|}}_{(a)}-D+\underbrace{\mathrm{Tr}(\Sigma_p^{-1}\Sigma_q)}_{(b)}+\underbrace{(\mu_p-\mu_q)^T\Sigma_p^{-1}(\mu_p-\mu_q)}_{(c)})\tag{6.11}$$

The choice of q and p is suggestive. using Eq. 6.10 & 6.11, we can derive a closed form for the KL divergence between $q(z_1\dots z_T)$ and $p(z_1\dots z_T)$. μ_q, Σ_q are the outputs of the variational model. Our functional form for μ_p, Σ_p is based on our generative and can be summarized as:

$$\mu_{p1}=0\qquad\Sigma_{p1}=\mathbb{1}\qquad\qquad\mu_{pt}=G(z_{t-1},u_{t-1})=G_{t-1}\qquad\Sigma_{pt}=\Delta\vec{\sigma}$$

Here, Σ_{pt} is assumed to be a learned diagonal matrix and Δ a scalar parameter.

Term (a) For $t=1$, we have:

$$\log\frac{|\Sigma_{p1}|}{|\Sigma_{q1}|}=\log|\Sigma_{p1}|-\log|\Sigma_{q1}|=-\log|\Sigma_{q1}|\tag{6.12}$$

For $t>1$, we have:

$$\log\frac{|\Sigma_{pt}|}{|\Sigma_{qt}|}=\log|\Sigma_{pt}|-\log|\Sigma_{qt}|=D\log(\Delta)+\log|\vec{\sigma}|-\log|\Sigma_{qt}|\tag{6.13}$$

Term (b) For $t=1$, we have:

$$\mathrm{Tr}(\Sigma_{p1}^{-1}\Sigma_{q1})=\mathrm{Tr}(\Sigma_{q1})\tag{6.14}$$

For $t > 1$, we have:

$$\mathrm{Tr}(\Sigma_{pt}^{-1}\Sigma_{qt}) = \frac{1}{\Delta}\mathrm{Tr}(\mathrm{diag}(\vec{\sigma})^{-1}\Sigma_{qt}) \tag{6.15}$$

Term (c) For $t = 1$, we have:

$$(\mu_{p1} - \mu_{q1})^T\Sigma_{p1}^{-1}(\mu_{p1} - \mu_{q1}) = ||\mu_{q1}||^2 \tag{6.16}$$

For $t > 1$, we have:

$$(\mu_{pt} - \mu_{qt})^T\Sigma_{pt}^{-1}(\mu_{pt} - \mu_{qt}) = \Delta(G_{t-1} - \mu_{qt})^T \mathrm{diag}(\vec{\sigma})^{-1}(G_{t-1} - \mu_{qt}) \tag{6.17}$$

Rewriting Eq. 6.10 using Eqns. 6.12, 6.13, 6.14, 6.15, 6.16, 6.17, we get:

$$\mathrm{KL}(q(z_1,\ldots,z_T)||p(z_1,\ldots,z_T)) = \frac{1}{2}((T-1)D\log(\Delta)\log|\vec{\sigma}| - \sum_{t=1}^{T}\log|\Sigma_{qt}|$$
$$+ \mathrm{Tr}(\Sigma_{q1}) + \frac{1}{\Delta}\sum_{t=2}^{T}\mathrm{Tr}(\mathrm{diag}(\vec{\sigma})^{-1}\Sigma_{qt}) + ||\mu_{q1}||^2$$
$$+ \Delta\sum_{t=2}^{T}\mathop{\mathbb{E}}_{z_{t-1}}\left[(G_{t-1} - \mu_{qt})^T \mathrm{diag}(\vec{\sigma})^{-1}(G_{t-1} - \mu_{qt})\right])$$

6.3.3 Learning with gradient ascent

The objective in Eq. 6.6 is differentiable in the parameters of the model (θ, ϕ). If the generative model θ is fixed, we perform gradient ascent of Eq. 6.6 in ϕ. Otherwise, we perform gradient ascent in both ϕ and θ. We use stochastic backpropagation Kingma & Welling (2014); Rezende *et al.* (2014) for estimating the gradient w.r.t. ϕ. Note that the expectations are only taken with respect to the variables z_{t-1}, z_t, which are the sufficient statistics of the Markov model. For the KL terms in Eq. 6.6, we use the fact that the prior $p_\theta(z_t|z_{t-1})$ and the variational approximation to the posterior $q_\phi(z_t|z_{t-1}, \vec{x})$ are both Normally distributed, and hence their KL divergence may be estimated analytically.

Algorithm 4 depicts an overview of the learning algorithm. We outline the algorithm for a mini-batch of size one, but in practice gradients are averaged across stochastically

Algorithm 4 Learning a DMM with stochastic gradient descent: We use a single sample from the recognition network during learning to evaluate expectations in the bound. We aggregate gradients across mini-batches.

```
Inputs: Dataset $\mathcal{D}$
        Inference Model: $q_\phi(\vec{z}|\vec{x})$
        Generative Model: $p_\theta(\vec{x}|\vec{z}), p_\theta(\vec{z})$
while notConverged() do
    1. Sample datapoint: $\vec{x} \sim \mathcal{D}$
    2. Estimate posterior parameters (Evaluate $\mu_\phi, \Sigma_\phi$)
    3. Sample $\hat{\vec{z}} \sim q_\phi(\vec{z}|\vec{x})$
    4. Estimate conditional likelihood: $p_\theta(\vec{x}|\hat{\vec{z}})$ & KL
    5. Evaluate $\mathcal{L}(\vec{x};(\theta,\phi))$
    6. Estimate MC approx. to $\nabla_\theta \mathcal{L}$
    7. Estimate MC approx. to $\nabla_\phi \mathcal{L}$
    (Use stochastic backpropagation to move gradients with respect to $q_\phi$ inside
    expectation)
    8. Update $\theta, \phi$ using ADAM (Kingma & Ba, 2014)
end while
```

sampled mini-batches of the training set. We take a gradient step in θ and ϕ, typically with an adaptive learning rate such as Kingma & Ba (2014).

6.4 Structured Inference Networks

We now detail how we construct the variational approximation q_ϕ, and specifically how we model the mean and diagonal covariance functions μ and Σ using recurrent neural networks (RNNs).

Since our implementation only models the diagonal of the covariance matrix (the vector valued variances), we denote this as σ^2 rather than Σ. This parameterization cannot in general be expected to be equal to $p_\theta(\vec{z}|\vec{x})$, but in many cases is often a reasonable approximation. We use RNNs due to their ability to scale well to large datasets.

Table 6.1 details the different choices for inference networks that we evaluate. The Deep Kalman Smoother **DKS** corresponds exactly to the functional form suggested by Eq. 6.4, and is our proposed variational approximation. The **DKS** smoothes information from the past (z_t) and future ($x_t, \ldots x_T$) to form the approximate posterior distribution.

Table 6.1: **Inference networks:** BRNN refers to a Bidirectional RNN and comb.fxn is shorthand for combiner function.

Inference network	Variational approximation for z_t	Implemented with
MF-LR	$q(z_t \vert x_1, \dots x_T)$	BRNN
MF-L	$q(z_t \vert x_1, \dots x_t)$	RNN
ST-L	$q(z_t \vert z_{t-1}, x_1, \dots x_t)$	RNN & comb.fxn
DKS	$q(z_t \vert z_{t-1}, x_t, \dots x_T)$	RNN & comb.fxn
ST-LR	$q(z_t \vert z_{t-1}, x_1, \dots x_T)$	BRNN & comb.fxn

We also evaluate other possibilities for the variational models (inference networks) q_ϕ: two are mean-field models (denoted **MF**) and two are structured models (denoted **ST**). They are distinguished by whether they use information from the past (denoted **L**, for left), the future (denoted **R**, for right), or both (denoted **LR**). See Fig. 6-2 for an illustration of two of these methods. Each one is conditional on a different subset of the observations to summarize information in the input sequence $\vec{x}$. **DKS** corresponds to **ST-R**.

The hidden states of the RNN parameterize the variational distribution, which go through what we call the "combiner function". We obtain the mean μ_t and diagonal covariance σ_t^2 for the approximate posterior at each time-step in a manner akin to Gaussian belief propagation. Specifically, we interpret the hidden states of the forward and backward RNNs as parameterizing the mean and variance of two Gaussian-distributed "messages" summarizing the observations from the past and the future, respectively. We then multiply these two Gaussians, performing a variance-weighted average of the means. All operations should be understood to be performed element-wise on the corresponding vectors. $h_t^{\text{left}}, h_t^{\text{right}}$ are the hidden states of the RNNs that run from the past and the future respectively (see Fig. 6-2).

Combiner function for mean field approximations: For the **MF-LR** inference network, the mean μ_t and diagonal variances σ_t^2 of the variational distribution $q_\phi(z_t|\vec{x})$ are predicted using the output of the RNN (not conditioned on z_{t-1}) as follows, where $\text{softplus}(x) = \log(1 + \exp(x))$:

$$\mu_{\text{r}} = W_{\mu_{\text{r}}}^{\text{right}} h_t^{\text{right}} + b_{\mu_{\text{r}}}^{\text{right}}, \qquad \sigma_{\text{r}}^2 = \text{softplus}(W_{\sigma_{\text{r}}^2}^{\text{right}} h_t^{\text{right}} + b_{\sigma_{\text{r}}^2}^{\text{right}})$$

$$\mu_{\text{l}} = W_{\mu_{\text{l}}}^{\text{left}} h_t^{\text{left}} + b_{\mu_{\text{l}}}^{\text{left}}, \qquad \sigma_{\text{l}}^2 = \text{softplus}(W_{\sigma_{\text{l}}^2}^{\text{left}} h_t^{\text{left}} + b_{\sigma_{\text{l}}^2}^{\text{left}})$$

$$\mu_t = \frac{\mu_{\text{r}}\sigma_{\text{l}}^2 + \mu_{\text{l}}\sigma_{\text{r}}^2}{\sigma_{\text{r}}^2 + \sigma_{\text{l}}^2}; \;\; \sigma_t^2 = \frac{\sigma_{\text{r}}^2\sigma_{\text{l}}^2}{\sigma_{\text{r}}^2 + \sigma_{\text{l}}^2}$$

Combiner function for structured approximations: The combiner functions for the structured approximations are implemented as:

$$\begin{aligned}
&\textit{(For } \textbf{ST-LR}\textit{)}\\
&h_{\text{combined}} = \frac{1}{3}(\tanh(Wz_{t-1}+b) + h_t^{\text{left}} + h_t^{\text{right}}),\\
&\textit{(For } \textbf{DKS}\textit{)}\\
&h_{\text{combined}} = \frac{1}{2}(\tanh(Wz_{t-1}+b) + h_t^{\text{right}}),\\
&\textit{(Posterior Means and Covariances)}\\
&\mu_t = W_\mu h_{\text{combined}} + b_\mu, \qquad \sigma_t^2 = \text{softplus}(W_{\sigma^2} h_{\text{combined}} + b_{\sigma^2})
\end{aligned}$$

The combiner function uses the tanh non-linearity from z_{t-1} to approximate the transition function (alternatively, one could share parameters with the generative model), and here we use a simple weighting between the components.

Related work: Archer *et al.* ; Gao *et al.* use $q(\vec{z}|\vec{x}) = \prod_t q(z_t|z_{t-1},\vec{x})$ where $q(z_t|z_{t-1},\vec{x}) = \mathcal{N}(\mu(x_t), \Sigma(z_{t-1}, x_t, x_{t-1}))$. The key difference from our approach is that this parameterization (in particular, conditioning the posterior means only on x_t) does not account for the information from the future relevant to the approximate posterior distribution for z_t.

Johnson *et al.* interleave predicting the local variational parameters of the graphical model (using an inference network) with steps of message passing inference. A key difference between our approach and theirs is that we rely on the structured inference network to predict the optimal local variational parameters directly. In contrast, in Johnson *et al.* , any suboptimalities in the initial local variational parameters may be overcome by the subsequent steps of optimization at additional computational cost.

Chung *et al.* propose the Variational RNN (VRNN) in which Gaussian noise is introduced at each time-step of a RNN. Chung *et al.* use an inference network that shares parameters with the generative model and only uses information from the past. If one views the noise variables and the hidden state of the RNN at time-step t together as z_t, then a factorization similar to Eq. 6.6 can be shown to hold, although the KL term would no longer have an analytic form since $p_\theta(z_t|z_{t-1}, x_{t-1})$ would not be Normally distributed. Nonetheless, our same structured inference networks (i.e. using an RNN to summarize observations from the future) could be used to improve the tightness of the variational lower bound, and our empirical results suggest that it would result in better learned models.

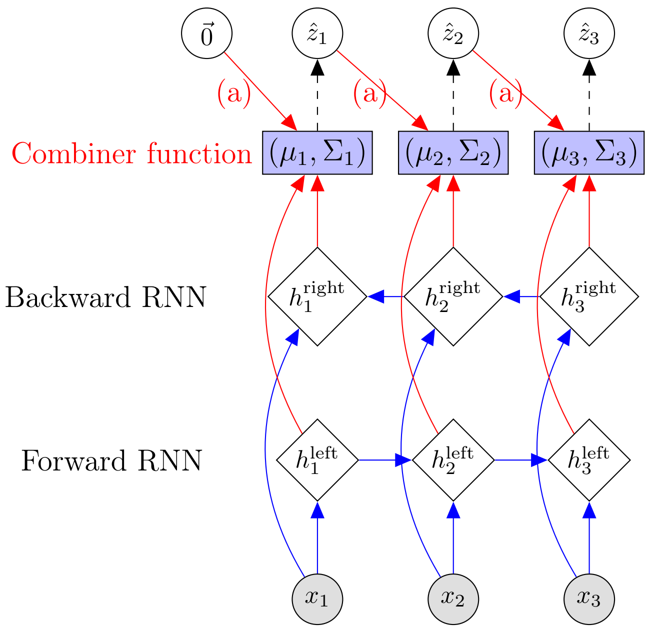


Figure 6-2: **Structured Inference Networks: MF-LR** and **ST-LR** variational approximations for a sequence of length 3, using a bi-directional recurrent neural net (BRNN). The BRNN takes as input the sequence $(x_1, \dots x_3)$, and through a series of non-linearities denoted by the blue arrows it forms a sequence of hidden states summarizing information from the left and right (h_t^{left} and h_t^{right}) respectively. Then through a further sequence of non-linearities which we call the "combiner function" (marked (a) above), and denoted by the red arrows, it outputs two vectors μ and Σ, parameterizing the mean and diagonal covariance of $q_\phi(z_t|z_{t-1}, \vec{x})$ of Eq. 6.5. Samples $\hat{z}_t$ are drawn from $q_\phi(z_t|z_{t-1}, \vec{x})$, as indicated by the black dashed arrows. For the structured variational models **ST-LR**, the samples $\hat{z}_t$ are fed into the computation of μ_{t+1} and Σ_{t+1}, as indicated by the red arrows with the label (a). The mean-field model does *not* have these arrows, and therefore computes $q_\phi(z_t|\vec{x})$. We use $\hat{z}_0 = \vec{0}$. The inference network for **DKS** (ST-R) is structured like that of ST-LR except without the RNN from the past.

6.5 Deep Markov Models

Following Raiko *et al.* (2006), we apply the ideas of deep learning to non-linear continuous state space models. When the transition and emission function have an unknown functional form, we parameterize $G_\alpha, S_\beta, F_\kappa$ from Eq. 6.1 with deep neural networks. See Fig. 6-1 (right) for an illustration of the graphical model.

Emission function: We parameterize the emission function F_κ using a two-layer MLP (multi-layer perceptron), $\text{MLP}(x, \text{NL}_1, \text{NL}_2) = \text{NL}_2(W_2\text{NL}_1(W_1x + b_1) + b_2))$, where NL denotes non-linearities such as ReLU, sigmoid, or tanh units applied element-wise to the input vector. For modeling binary data,

$$F_\kappa(z_t) = \text{sigmoid}(W_{\text{emission}}\text{MLP}(z_t, \text{ReLU}, \text{ReLU}) + b_{\text{emission}})$$

parameterizes the mean probabilities of independent Bernoullis.

Gated transition function: We parameterize the transition function from z_t to z_{t+1} using a gated transition function inspired by Gated Recurrent Units (Chung *et al.* , 2014), instead of an MLP. Gated recurrent units (GRUs) are a neural architecture that parameterizes the recurrence equation in the RNN with gating units to control the flow of information from one hidden state to the next, conditioned on the observation. Unlike GRUs, in the DMM, the transition function is not conditional on any of the observations. All the information must be encoded in the completely stochastic latent state. To achieve this goal, we create a Gated Transition Function. We would like the model to have the flexibility to choose a linear transition for some dimensions while having a non-linear transitions for the others. We adopt the following parameterization, where $\mathbb{I}$ denotes the identity function and $\odot$ denotes element-wise multiplication:

$$\begin{aligned}
&g_t = \text{MLP}(z_{t-1}, \text{ReLU}, \text{sigmoid}) \;\; \textit{(Gating Unit)}\\
&h_t = \text{MLP}(z_{t-1}, \text{ReLU}, \mathbb{I}) \;\; \textit{(Proposed mean)}\\
&\textit{(Transition Mean } G_\alpha \textit{ and } S_\beta\textit{)}\\
&\mu_t(z_{t-1}) = (1 - g_t) \odot (W_{\mu_p} z_{t-1} + b_{\mu_p}) + g_t \odot h_t\\
&\sigma_t^2(z_{t-1}) = \text{softplus}(W_{\sigma_p^2}\text{ReLU}(h_t) + b_{\sigma_p^2})
\end{aligned}$$

Note that the mean and covariance functions both share the use of h_t. In our experiments, we initialize W_{μ_p} to be the identity function and b_{μ_p} to 0. The parameters of the emission and transition function form the set θ.

6.6 Evaluation

We use Adam Kingma & Ba (2014) with a learning rate of 0.0008 to train the DMM. In the models we trained, the hidden dimension was set to be 100 for the emission distribution and 200 in the transition function. We typically used RNN sizes from one of $\{400, 600\}$ and a latent dimension of size 100. We study the inference algorithm and the model on three datasets.

6.6.1 Synthetic data

Dataset: We consider simple linear and non-linear GSSMs. To train the inference networks we use $N = 5000$ datapoints of length $T = 25$. We consider both one and two dimensional systems for inference and parameter estimation. We compare our results using the training value of the variational bound $\mathcal{L}(\vec{x}; (\theta, \phi))$ (Eq. 6.6) and the RMSE $= \sqrt{\frac{1}{N}\frac{1}{T}\sum_{i=1}^{N}\sum_{t=1}^{T}[\mu_\phi(x_{i,t}) - z^*_{i,t}]^2}$, where z^* correspond to the true underlying z's that generated the data.

Compiling exact inference: We seek to understand whether inference networks can accurately compile exact posterior inference into the network parameters ϕ for linear GSSMs when exact inference is feasible. For this experiment we optimize Eq. 6.6 over ϕ, while θ is fixed to a synthetic distribution given by a one-dimensional GSSM. We compare results obtained by the various approximations we propose to those obtained by an implementation of Kalman smoothing (Duckworth, 2016) which performs *exact inference*. Fig. 6-3 (top and middle) depicts our results. The proposed **DKS** (i.e., **ST-R**) and **ST-LR** outperform the mean-field based variational method **MF-L** that only looks at information from the past. **MF-LR**, however, is often able to catch up when it comes to RMSE, highlighting the role that information from the future plays when performing posterior inference, as is evident in the posterior factorization in Eq. 6.4. Both **DKS** and **ST-LR** converge to the RMSE of the exact Smoothed KF, and moreover their lower bound on the likelihood becomes tight.

Approximate inference and parameter estimation: Here, we experiment with applying the inference networks to synthetic non-linear generative models as well as using **DKS** for learning a subset of parameters within a fixed generative model. On synthetic non-linear datasets (see supplemental material) we find, similarly, that the structured variational approximations are capable of matching the performance of inference using a smoothed Unscented Kalman Filter Wan & Van Der Merwe (2000) on held-out data. Finally, Fig. 6-4 illustrates a toy instance where we successfully perform parameter estimation in a synthetic, two-dimensional, non-linear GSSM.

Experimental setup: We used an RNN size of 40 in the inference networks used for the synthetic experiments.

Linear SSMs: Fig. 6-5 (N=500, T=25) depicts the performance of inference networks, only now using held out data to evaluate the RMSE and the upper bound. We find that the results echo those in the training set, and that on unseen data points, the inference networks, particularly the structured ones, are capable of generalizing

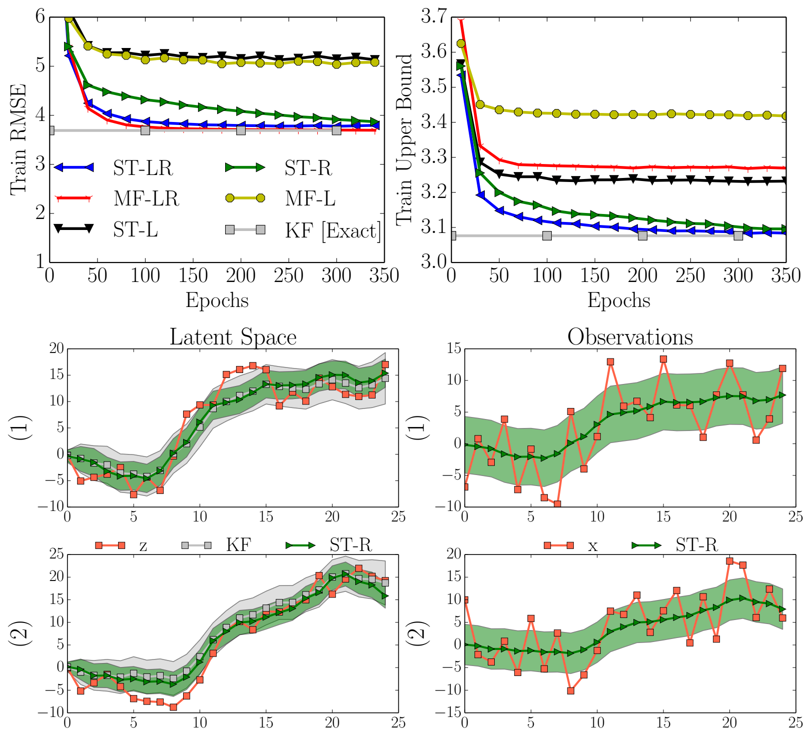


Figure 6-3: **Synthetic evaluation:** (**Top & Bottom**) Compiled inference for a *fixed* linear GSSM: $z_t \sim \mathcal{N}(z_{t-1} + 0.05, 10)$, $x_t \sim \mathcal{N}(0.5z_t, 20)$. The training set comprised $N = 5000$ one-dimensional observations of sequence length $T = 25$. **(Top left)** RMSE with respect to true z^* that generated the data. **(Top right)** Variational bound during training. The results on held-out data are very similar (see supplementary material). **(Bottom four plots)** Visualizing inference in two sequences (denoted (1) and (2)); Left panels show the Latent Space of variables z, right panels show the Observations x. Observations are generated by the application of the emission function to the posterior shown in Latent Space. Shading denotes standard deviations.

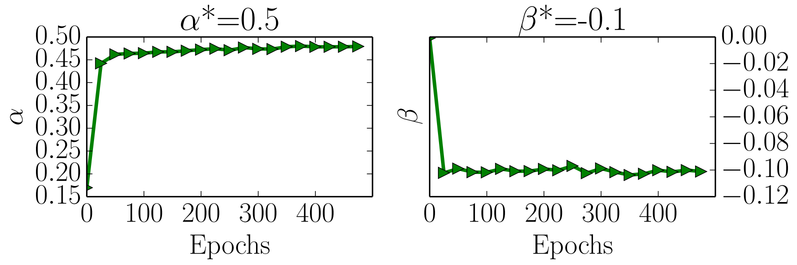


Figure 6-4: **Parameter estimation:** Learning parameters α, β in a two-dimensional non-linear GSSM. $N = 5000, T = 25$ $\vec{z}_t \sim \mathcal{N}([0.2z^0_{t-1} + \tanh(\alpha z^1_{t-1}); 0.2z^1_{t-1} + \sin(\beta z^0_{t-1})], 1.0)$ $\vec{x}_t \sim \mathcal{N}(0.5\vec{z}_t, 0.1)$ where $\vec{z}$ denotes a vector, [] denotes concatenation and superscript denotes indexing.

compiled inference.

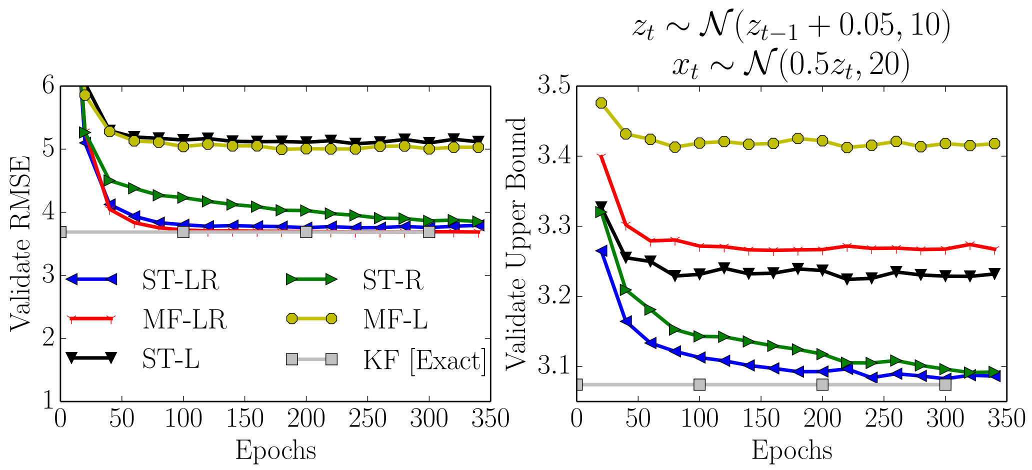


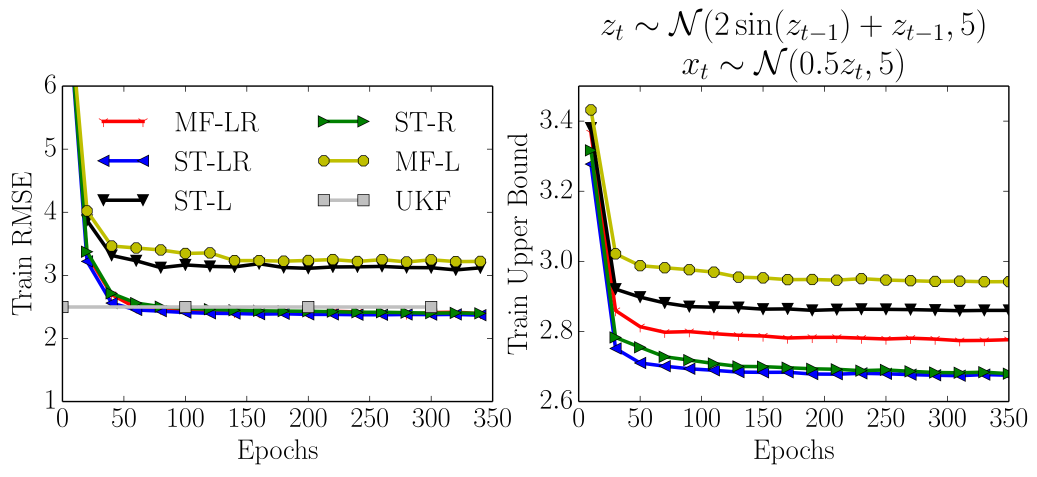
Figure 6-5: **Inference in a linear SSM on held-out data:** Performance of inference networks on held-out data using a generative model with Linear Emission and Linear Transition

Non-linear SSMs: Fig. 6-6 considers learning inference networks on a synthetic non-linear dynamical system ($N = 5000, T = 25$). We find once again that inference networks that match the posterior realize faster convergence and better training (and validation) accuracy.

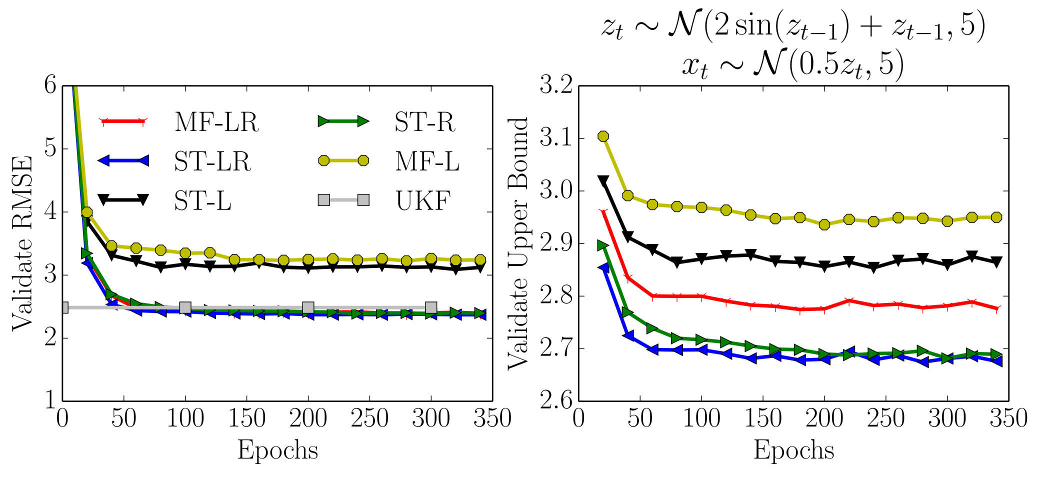
Visualizing posterior estimations: In Fig. 6-7 we visualize the posterior estimates obtained by the inference network. We run posterior inference on the training set 10 times and take the empirical expectation of the posterior means and covariances of each method. We compare posterior estimates with those obtained by a smoothed Unscented Kalman Filter (UKF) Wan & Van Der Merwe (2000).

6.6.2 Polyphonic music

Dataset: We train DMMs on polyphonic music data Boulanger-Lewandowski *et al.* (2012). An instance in the sequence comprises an 88-dimensional binary vector corresponding to the notes of a piano. We learn for 2000 epochs and report results based on early stopping using the validation set. We report held-out negative log-likelihood (NLL) in the format "a (b) {c}". a is an importance sampling based estimate of the NLL (details in supplementary material); $b = \frac{1}{\sum_{i=1}^{N} T_i} \sum_{i=1}^{N} -\mathcal{L}(\vec{x}; \theta, \phi)$ where T_i is the length of sequence i. This is an upper bound on the NLL, which facilitates comparison to RNNs; TSBN Gan *et al.* (2015) (in their code) report $c = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{T_i} \mathcal{L}(\vec{x}; \theta, \phi)$. We compute this to facilitate comparison with their work.



(a) Performance on training data



(b) Performance on held-out data

Figure 6-6: **Inference in a don-linear SSM:** Performance of inference networks trained with data from a Linear Emission and Non-linear Transition SSM

As in (Sønderby *et al.* , 2016a), we found annealing the KL divergence in the variational bound ($\mathcal{L}(\vec{x};(\theta,\phi))$) from 0 to 1 over 5000 parameter updates got better results.

Mean-Field vs Structured Inference Networks: Table 6.2 shows the results of learning a DMM on the polyphonic music dataset using **MF-LR**, **ST-L**, **DKS** and **ST-LR**. **ST-L** is a structured variational approximation that only considers information from the past and, up to implementation details, is comparable to the one used in Gregor *et al.* (2015). Comparing the negative log-likelihoods of the learned models, we see that the looseness in the variational bound (which we first observed in the synthetic setting in Fig. 6-3 top right) significantly affects the ability to learn. **ST-LR** and **DKS** substantially outperform **MF-LR** and **ST-L**. This adds credence

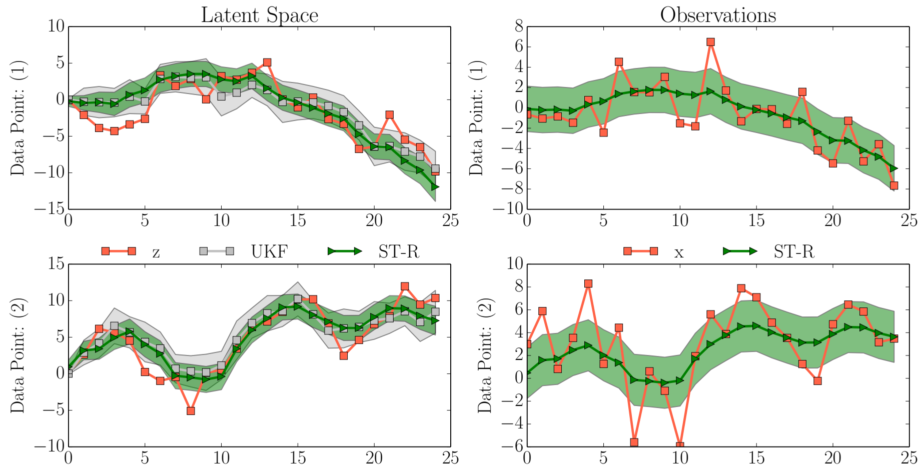


Figure 6-7: **Inference on non-linear synthetic data:** Visualizing inference on training data. Generative Models: (a) Linear Emission and Non-linear Transition z^* denotes the latent variable that generated the observation. x denotes the true data. We compare against the results obtained by a smoothed Unscented Kalman Filter (UKF) (Wan & Van Der Merwe, 2000). The column denoted "Observations" denotes the result of applying the emission function of the respective generative model on the posterior estimates shown in the column "Latent Space". The shaded areas surrounding each curve μ denotes $\mu \pm \sigma$ for each plot.

to the idea that by taking into consideration the factorization of the posterior, one can perform better inference and, consequently, learning, in real-world, high dimensional settings. Note that the **DKS** network has half the parameters of the **ST-LR** and **MF-LR** networks.

A Generalization of the DMM: To display the efficacy of our inference algorithm to model variants beyond first-order Markov Models, we further augment the DMM with edges from x_{t-1} to z_t and from x_{t-1} to x_t. We refer to the resulting generative model as DMM-Augmented (Aug.). Augmenting the DMM with additional edges realizes a richer class of generative models.

We show that **DKS** can be used *as is* for inference on a more complex generative model than DMM, while making gains in held-out likelihood. All following experiments use **DKS** for posterior inference.

The baselines we compare to in Table 6.3 have more complex generative models than the DMM. STORN has edges from x_{t-1} to z_t given by the recurrence update and TSBN has edges from x_{t-1} to z_t as well as from x_{t-1} to x_t. HMSBN shares the same structural properties as the DMM, but is learned using a simpler inference network.

Table 6.2: **Comparing inference networks:** Test negative log-likelihood on polyphonic music of different inference networks trained on a DMM with a fixed structure (lower is better). The numbers inside parentheses are the variational bound.

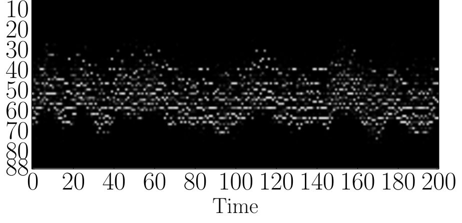
Inference Network	JSB	Nottingham	Piano	Musedata
DKS (i.e., **ST-R**)	6.605 (7.033)	3.136 (3.327)	8.471 (8.584)	7.280 (7.136)
ST-L	7.020 (7.519)	3.446 (3.657)	9.375 (9.498)	8.301 (8.495)
ST-LR	6.632 (7.078)	3.251 (3.449)	8.406 (8.529)	7.127 (7.268)
MF-LR	6.701 (7.101)	3.273 (3.441)	9.188 (9.297)	8.760 (8.877)

In Table 6.3, as we increase the complexity of the generative model, we obtain better results across all datasets.

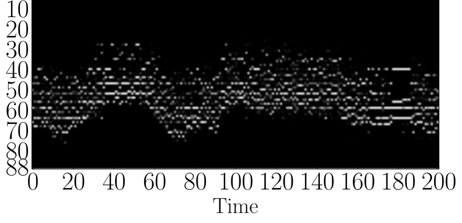
The DMM outperforms both RNNs and HMSBN everywhere, outperforms STORN on JSB, Nottingham and outperform TSBN on all datasets except Piano. Compared to LV-RNN (that optimizes the inclusive KL-divergence), DMM-Aug obtains better results on all datasets except JSB. This showcases our flexible, structured inference network's ability to learn powerful generative models that compare favourably to other state of the art models.

Samples: Fig. 6-8 depicts mean probabilities of samples from the DMM trained on JSB Chorales (Boulanger-Lewandowski *et al.* , 2012). MP3 songs corresponding to two different samples from the best DMM model learned on each of the four polyphonic data sets may be found in the code repository.

Experiments with NADE: We also experimented with Neural Autoregressive Density Estimators (NADE) (Larochelle & Murray, 2011) in the emission distribution for DMM-Aug and denote it DMM-Aug-NADE. In Table 6.4, we see that DMM-Aug-NADE performs comparably to the state of the art RNN-NADE on JSB, Nottingham and Piano.



(a) Sample 1



(b) Sample 2

Figure 6-8: Two samples from the DMM trained on JSB Chorales

Table 6.3: **Evaluation against baselines:** Test negative log-likelihood (lower is better) on Polyphonic Music Generation dataset. **Table Legend**: RNN Boulanger-Lewandowski *et al.* (2012), LV-RNN Gu *et al.* (2015), STORN Bayer & Osendorfer (2014), TSBN, HMSBN Gan *et al.* (2015).

Methods	JSB	Nottingham	Piano	Musedata
DMM	6.388 (6.926) {6.856}	2.770 (2.964) {2.954}	7.835 (7.980) {8.246}	6.831 (6.989) {6.203}
DMM-Aug.	6.288 (6.773) {6.692}	2.679 (2.856) {2.872}	7.591 (7.721) {8.025}	6.356 (6.476) {5.766}
HMSBN	(8.0473) {7.9970}	(5.2354) {5.1231}	(9.563) {9.786}	(9.741) {8.9012}
STORN	6.91	2.85	7.13	6.16
RNN	8.71	4.46	8.37	8.13
TSBN	{7.48}	{3.67}	{7.98}	{6.81}
LV-RNN	3.99	2.72	7.61	6.89

6.6.3 EHR Patient Data

Learning models from large observational health datasets is a promising approach to advancing precision medicine and could be used, for example, to understand which medications work best, for whom.

However, working with EHR data poses some technical challenges: EHR data are noisy, high dimensional and difficult to characterize easily. Patient data is rarely contiguous over large parts of the dataset and is often missing (not at random). We learn a DMM on the data showing how to handle the aforementioned technical challenges.

Dataset: The dataset we use comprises 5000 diabetic patients using data from a major health insurance provider. The observations of interest are: A1c level (hemoglobin A1c, a protein for which a high level indicates that the patient is diabetic) and glucose (blood sugar). We bin glucose into quantiles and A1c into clinically meaningful bins. The observations also include age, gender and ICD-9 diagnosis codes for co-morbidities of diabetes such as congestive heart failure, chronic kidney disease and obesity. There are 48 binary observations for a patient at every time-step. We group each patient's data (over 4 years) into three month intervals, yielding a sequence

Table 6.4: **Experiments with NADE Emission:** Test negative log-likelihood (lower is better) on Polyphonic Music Generation dataset. **Table Legend**: RNN-NADE (Boulanger-Lewandowski *et al.* , 2012)

Methods	JSB	Nottingham	Piano	Musedata
DMM-Aug.-NADE	5.118 (5.335) {5.264}	2.305 (2.347) {2.364}	7.048 (7.099) {7.361}	6.049 (6.115) {5.247}
RNN-NADE	5.19	2.31	7.05	5.60

of length 18.

Graphical Model: Fig. 6-9 represents the generative model we use when $T = 4$. The model captures the idea of an underlying time-evolving latent state for a patient (z_t) that is solely responsible for the diagnosis codes and lab values (x_t) we observe. In addition, the patient state is modulated by drugs (u_t) prescribed by the doctor. We may assume that the drugs prescribed at any point in time depend on the patient's entire medical history though in practice, the dotted edges in the Bayesian network never need to be modeled since x_t and u_t are always assumed to be observed. A natural line of follow up work would be to consider learning when u_t is missing or latent.

We make use of time-varying (binary) drug prescription u_t for each patient by augmenting the DMM with an additional edge every time step. Specifically, the DMM's transition function is now $z_t \sim \mathcal{N}(G_\alpha(z_{t-1}, u_{t-1}), S_\beta(z_{t-1}, u_{t-1}))$ (cf. Eq. 6.1). In our data, each u_t is an indicator vector of eight anti-diabetic drugs including Metformin and Insulin, where Metformin is the most commonly prescribed first-line anti-diabetic drug.

Emission & transition function: The choice of emission and transition function to use for such data is not well understood. In Fig. 6-10 (right), we experiment with variants of DMMs and find that using MLPs (rather than linear functions) in the emission and transition function yield the best generative models in terms of held-out likelihood. In the Chapter 7, we will improve upon these choices and show how to leverage insights from pharmacology to design better transition functions. In these experiments, the hidden dimension was set as 200 for the emission and transition functions. We used an RNN size of 400 and a latent dimension of size 50. We use the **DKS** as our inference network for learning.

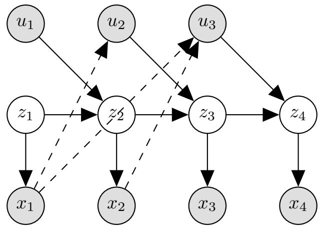


Figure 6-9: **DMM for medical data:** The DMM (from Fig. 6-1) is augmented with external actions u_t representing medications presented to the patient. z_t is the latent state of the patient. x_t are the observations that we model. Since both u_t and x_t are always assumed observed, the conditional distribution $p(u_t|x_1,\ldots,x_{t-1})$ may be ignored during learning.

Learning with missing data: In the EHR dataset, a subset of the observations (e.g. A1C and Glucose values used to assess blood-sugar levels for diabetics) are frequently missing in the data. We marginalize them out during learning, which is straightforward within the probabilistic semantics of our Bayesian network. The sub-network of the original graph we are concerned with is the emission function since missingness affects our ability to evaluate $\log p(x_t|z_t)$ (the first term in Eq. 6.6). The missing random variables are leaves in the Bayesian sub-network (comprised of the emission function). Consider a simple example of two modeling two observations at time t, namely m_t, o_t. The log-likelihood of the data (m_t, o_t) conditioned on the latent variable z_t decomposes as $\log p(m_t, o_t|z_t) = \log p(m_t|z_t) + \log p(o_t|z_t)$ since the random variables are conditionally independent given their parent. If m is missing and marginalized out while o_t is observed, then our log-likelihood is: $\log \int_m p(m_t, o_t|z_t) = \log(\int_m p(m_t|z_t)p(o_t|z_t)) = \log p(o_t|z_t)$ (since $\int_m p(m_t|z_t) = 1$) i.e we effectively ignore the missing observations when estimating the log-likelihood of the data. In practice, we track indicators denoting whether A1C values and Glucose values were observed in the data. These are used as markers of missingness. During batch learning, at every time-step t, we obtain a matrix $B = \log p(x_t|z_t)$ of size batch-size $\times$ 48, where 48 is the dimensionality of the observations, comprising the log-likelihoods of every dimension for patients in the batch. We multiply this with a matrix of M. M has the same dimensions as B and has a 1 if the patient's A1C value was observed and a 0 otherwise. For dimensions that are never missing, M is always 1.

The effect of anti-diabetic medications: As an illustrative example of how DMMs could be used in precision medicine in the future, we ask a counterfactual question using the DMM: what *would have happened* to a patient had anti-diabetic drugs not been prescribed? This is causal query that in general, is impossible to answer without typically untestable (Pearl, 2009) assumptions. We will require that the causal effect under the model in Figure 6-9 be identifiable, no unobserved confounding over

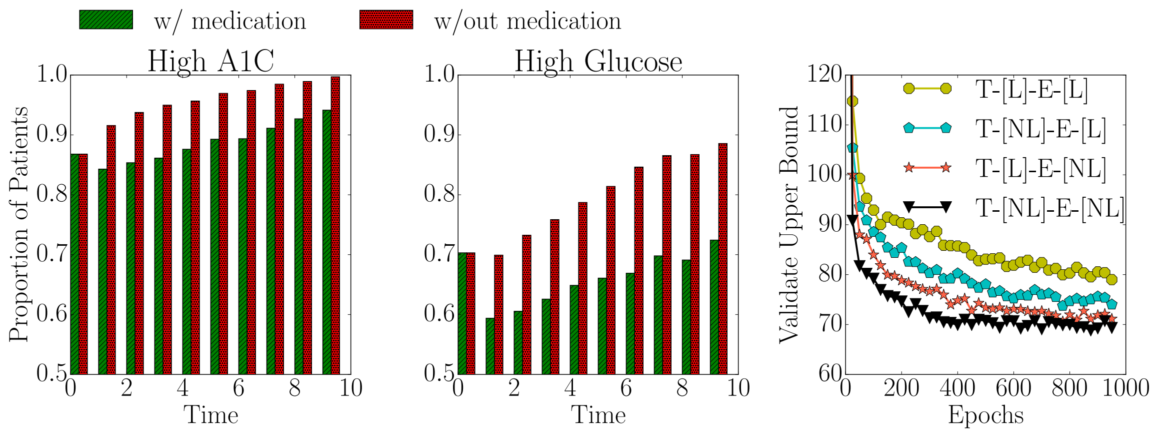


Figure 6-10: **Left two plots; Estimating counterfactuals with DMM:** The x-axis denotes the number of 3-month intervals after prescription of Metformin. The y-axis denotes the proportion of patients (out of a test set size of 800) who, after their first prescription of Metformin, experienced a high level of A1C. In each tuple of bar plots at every time step, the left aligned bar plots (green) represent the population that received diabetes medication while the right aligned bar plots (red) represent the population that did not receive diabetes medication. **(Rightmost plot)** Upper bound on negative-log likelihood for different DMMs trained on the medical data. (T) denotes "transition", (E) denotes "emission", (L) denotes "linear" and (NL) denotes "non-linear".

time, and the assumption of positive support. We refer the reader to (Chakraborty, 2013; Hernán & Robins, 2020) for a thorough discussion on the assumptions necessary for causal inference to be feasible from sequential, observational data.

The experiment we will conduct asks what happens to a patient under a treatment plan that is never observed (namely that of not prescribing medication). This is by no means a clinically meaningful experiment; rather, it serves to illustrate how deep generative models can find use a as nonlinear structural equation model Pearl (2012).

We are interested in the patient's blood-sugar level as measured by the widely-used A1C blood-test. We perform inference using held-out patient data leading up to the time k of first prescription of Metformin and let T denote the maximum length of the patient's clinical data. From the posterior mean, we perform ancestral sampling tracking two latent trajectories: (1) the factual: where we sample new latent states conditioned on the medication the patient actually received and (2) the counterfactual: where we sample conditioned on not receiving any drugs for all remaining timesteps (i.e u_k set to the zero-vector). We reconstruct the patient observations $x_k, \ldots, x_T$, threshold the predicted values of A1C levels into high and low and visualize the average number of high A1C levels we observe among the synthetic patients in both scenarios. This is an example of performing do-calculus Pearl (2009) in order to estimate model-based counterfactual effect. More formally, we can pose our experiment as one of comparing $p(x_{k+1:T}|x_{1:k}, u_{1:T})$ with $p(x_{k+1:T}|x_{1:k}, u_{1:t}, \text{do}(u_{k+1:T}) = 0)$.

The results are shown in Fig. 6-10. On average, the model has learned that patients who were prescribed anti-diabetic medication had more controlled levels of A1C than patients who did not receive any medication. Despite being an aggregate effect, this is interesting because it is a phenomenon that coincides with our intuition but was confirmed by the model in an entirely unsupervised manner. Note that in our dataset, most diabetic patients are indeed prescribed anti-diabetic medications, making the counterfactual prediction harder.

Sampling a patient: We visualize samples from the DMM trained on medical data in Fig. 6-11 The model captures correlations within timesteps as well as variations in A1C level and Glucose level across timesteps. It also captures rare occurrences of comorbidities found amongst diabetic patients.

6.7 Discussion

This chapter introduces Deep Markov Models alongside a black-box variational learning algorithm. The underlying methodological principle we propose is to build the inference network in a manner that mimics the factorization structure in the true posterior distribution (under the generative model). In the context of learning algorithms hierarchical deep generative models of static data, (Sønderby *et al.* , 2016b) were among the first to make use of this principle. Concurrent to our own work, (Fraccaro *et al.* , 2016) also make use of this principle in building learning algorithms for sequential models of time-series data. (Webb *et al.* , 2018) provide an algorithm for faithfully inverting the dependency structure in any generative model, empirically demonstrating that adherence to this principle yields gains in generalization across a variety of deep generative models. By making use of an inference network, the space complexity of our learning algorithm depends neither on the sequence length T nor on the training set size N, offering massive savings compared to classical variational inference methods.

Our work has spurred further research into inference networks as well as applications and extensions of sequential deep generative models. Toyer *et al.* (n.d.) study the application of DMMs towards human pose forecasting. Che *et al.* (2018a) develop hierarchical DMMs where hierarchies of latent variables capture patterns in multi-rate, high-dimensional time-series data. Finally, Zhi-Xuan *et al.* (2020) extend DMMs to learn unified, time-varying representations of multi-modal data. An open source implementation of DMMs is also available in the probabilistic programming package Pyro (Bingham *et al.* , 2019).

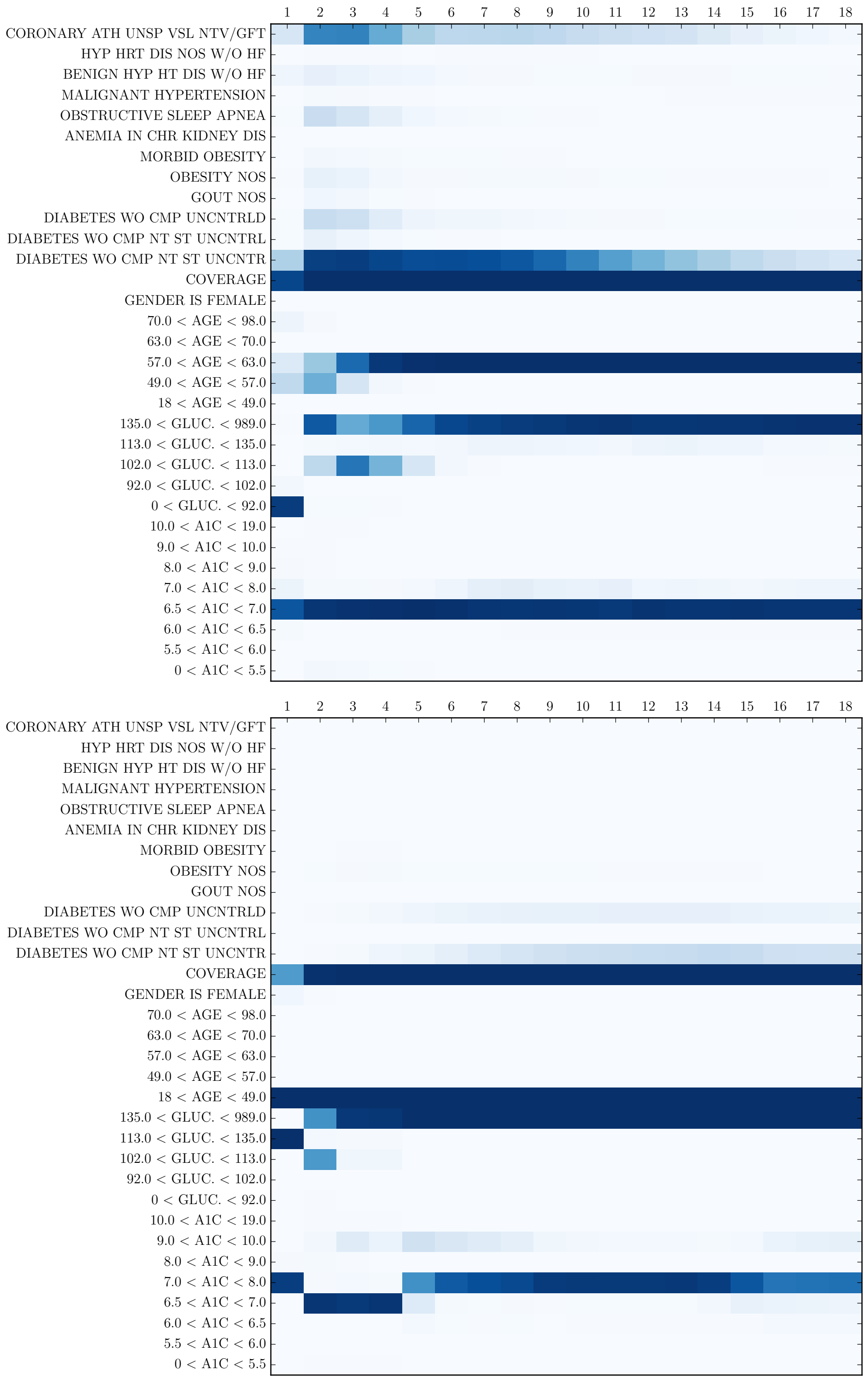


Figure 6-11: **Patient data generated by a DMM** Samples of a patient generated by the model. The x-axis denotes time and the y-axis denotes the observations. The intensity of the color denotes its value between zero and one

Chapter 7

Inductive biases for clinical data

In Chapter 6 we showed how to make use of structure in the graphical model to derive learning algorithms for nonlinear state space models of clinical data. To make predictions from longitudinal data or deconstruct salient structure within, we need good sequential models. However, modeling longitudinal observations in the presence of time-varying interventions is challenging. In this chapter, we study whether we can improve upon the use of multi-layer perceptrons in the conditional probability distributions of DMMs.

Models parameterize *intervention effect functions* (IEFs), which determine how the model responds to an intervention, in different ways. A common choice, that we made in the previous chapter, for high-dimensional data is to use neural networks. However, datasets in healthcare can be small, leaving such approaches prone to overfitting. We show how to make deep learning practical in the low-data regime by building new neural architectures inspired by ideas from pharmacology. In doing so, we show how practitioners can use domain knowledge and patterns in time-varying interventions to construct IEFs. In various non-linear, sequential models of disease progression, across both synthetic and real-world data, our proposed IEF yields dramatic improvements in generalization where other representation learning approaches overfit.

7.1 Introduction

Deep generative models capture changes in high-dimensional, longitudinal observations using time-varying hidden representations, such as Recurrent Neural Networks (RNNs)

(Chung *et al.* , 2014) or nonlinear state space models (Krishnan *et al.* , 2017; Fraccaro *et al.* , 2016). When control signals, or interventions, drive variation in observations, models condition their representations on the intervention to capture this variation. Functions used to capture the effect of an intervention have various names. For example, in model-based reinforcement learning (RL), *dynamics functions* or *action-dependent state transition* functions (Chiappa *et al.* , 2017; Oh *et al.* , 2015) simulate observations in response to control signals. In causal inference, *dose response functions* (Silva, 2016; Schwab *et al.* , 2019) capture variation in a biomarker as a result of a drug dosage. We call such functions *intervention effect functions* (IEFs): $\mathbb{IEF}(S_t, U_t, B)$. S_t denotes a representation in a model that undergoes change due to a (possibly high-dimensional) intervention U_t and static covariates B.

In healthcare, IEFs can be used to build decision support tools by enabling practitioners to ask and answer counterfactuals using models learned from observational data (Rubin, 1974; Pearl *et al.* , 2009). Schulam & Saria (2017); Silva (2016) use observational data to learn Gaussian processes (GPs) that, under strong assumptions on the data, characterize counterfactuals over how a single intervention affects a single biomarker over time. Soleimani *et al.* (2017) propose multi-output GPs to model variation in multiple biomarkers. Biases in data can hinder learning IEFs in time-varying settings; consequently, Lim (2018) use propensity weighting to adjust for time-dependent confounders. We seek to extend these successes to representation learning based models for two reasons. Firstly, disease progression is increasingly being tracked not just through a patient's time-varying clinical biomarkers but also through their genetics; the integration of such high-dimensional, multi-modal information is therefore vital, and an area where representation learning shows enormous promise (Wu & Goodman, 2018). Secondly, representation learning gives us myriad ways to transfer and combine domain knowledge; one example of this is Sachan *et al.* (2017), who show that embeddings pre-trained on unlabeled medical text data yield better predictive performance on biomedical named entity recognition compared to general purpose embeddings. Ultimately, for representation learning models to be useful in clinical decision support, we need good counterfactual models. A good counterfactual model must answer factuals well. Thus, we focus on building unsupervised models of observational clinical data conditioned on time-varying interventions.

In reality, clinical datasets may be small due to rarity of chronic diseases, or due to costs incurred in the collection and curation of rich, multi-modal patient datasets. We need models for unsupervised learning that are practical even with a few hundred samples. Using neural networks in IEFs of unsupervised models (Krishnan *et al.* , 2017; Lipton *et al.* , 2015) for such data risks overfitting. One may overcome the limited-data problem by using domain expertise; for example, in model-based RL, Du & Narasimhan (2019); Scholz *et al.* (2014) use the physics of how objects interact to design the dynamics function. This approach can reduce sample complexity, but relies on domain knowledge. We seek to understand what the *correct* domain knowledge is in settings where it is not easily obtainable and difficult to formalize, and how one should leverage it when building IEFs for unsupervised models of disease progression.

This chapter makes several contributions towards both machine learning and its applications to healthcare. First, we propose a novel neural architecture for an IEF, $\mathbb{PK}$-$\mathbb{PD}_{\mathbb{IEF}}$, that blends mathematical models from pharmacology with deep learning. The IEF is flexible and leverages unique structure in the treatments prescribed to chronically ill patients. Second, we show that the incorporation of domain knowledge in unsupervised models of high-dimensional clinical data aids generalization in the low data regime. We study the use of $\mathbb{PK}$-$\mathbb{PD}_{\mathbb{IEF}}$ in three unsupervised models (on both synthetic and real-world patient data) and find strong differential improvements in generalization conferred from the use of $\mathbb{PK}$-$\mathbb{PD}_{\mathbb{IEF}}$ when data is scarce. Qualitatively, the neural architecture is interpretable, and captures known clinical knowledge regarding the treatment effect. Third, we release code for the $\mathbb{PK}$-$\mathbb{PD}_{\mathbb{IEF}}$ and the ML-MMRF dataset, which is a curated, pre-processed subset of the CoMMpass study (Multiple Myeloma Research Foundation & others, 2011) set up for the machine learning and healthcare communities to study these and other questions.

7.2 Setup

To ground our discussion, we focus on IEFs tailored for clinical data of chronically ill patients. Chronic diseases (e.g. cancer, heart disease) are those which require long-term medical attention and result in one or more organ systems being compromised. The progression of chronic disease is tracked via clinical biomarkers whose evolution is influenced by static factors like age, genetics and medical history. Patient data for those suffering from such diseases may be very limited, making these disease cohorts ideal for studying questions about the generalization of unsupervised models in the low-data regime. Patients suffering from such diseases tend to have pre-determined schedules where their progress is measured and treatments are prescribed. We therefore turn to discrete-time models as a reasonable approximation of the underlying data generating process. We review three models of sequential data conditioned on interventions and highlight the IEF within each one.

Notation: Let $B \in \mathbb{R}^J$ denote baseline data that are static, i.e. individual-specific covariates. Let $\mathbf{U} = \{U_0, \ldots, U_{T-1}\}$; $U_t \in \mathbb{R}^L$ be a sequence of L dimensional interventions for an individual. An element of U_t may be binary, to denote prescription of a drug, or real-valued, to denote the dosage. Let $\mathbf{X} = \{X_1, \ldots, X_T\}$; $X_t \in \mathbb{R}^M$ denote the sequence of real-valued, M dimensional clinical biomarkers. An element of X_t may denote a serum lab value or blood count, which is used by clinicians to measure organ function as a proxy for disease severity. We assume access to a dataset $\mathcal{D} = \{(\mathbf{X}^1, \mathbf{U}^1, B^1), \ldots, (\mathbf{X}^N, \mathbf{U}^N, B^N)\}$. Unless required, we ignore the superscript denoting the index of the datapoint and denote concatenation with []. The goal of our work is to build models of $\mathbf{X}$ conditioned on $\mathbf{U}, B$. We denote the parameters of a model by θ, which may comprise weight matrices or the parameters of functions that index θ. Each model will have an $\mathbb{IEF}(S_{t-1}, U_{t-1}, B)$.

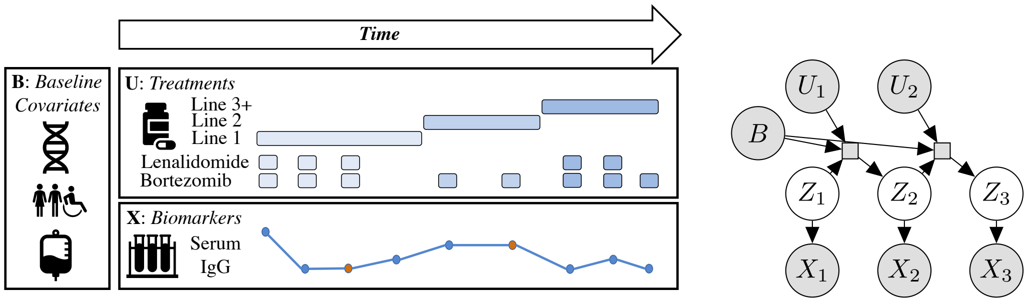


Figure 7-1: **Patient Data (Left):** Illustration of data from a multiple myeloma patient. Baseline (static) data typically consists of genomics, demographics, and initial labs. Longitudinal data typically includes laboratory values (e.g. serum IgG) and treatments. Baseline data is usually complete, but longitudinal measurements are frequently missing at various time points. The data tells a rich story of a patient's disease trajectory and the resulting treatment decisions. For example, a deviation of a lab value from a healthy range (e.g. spike in serum IgG) might prompt a move to the next line of therapy. Missing data (e.g. points in red) in this case are forward filled. **Unsupervised Models of Sequential Data (Right):** We show a State Space Model (SSM) of $\mathbf{X}$ (the longitudinal biomarkers) conditioned on B (genetics, denographics) and $\mathbf{U}$ (binary indicators of treatment and line of therapy). The rectangle depicts the $\mathbb{IEF}(S_{t-1}, U_{t-1}, B)$, where $S_{t-1} = Z_{t-1}$.

7.2.1 First Order Markov Models (FOMMs)

FOMMs assume observations are conditionally independent of the past given the previous observation, intervention and baseline covariates:

$p(\mathbf{X}|\mathbf{U}, B) = \prod_{t=1}^{T} p(X_t|X_{t-1}, U_{t-1}, B); X_t \sim \mathcal{N}(\mu_\theta(X_{t-1}, U_{t-1}, B), \Sigma_\theta(X_{t-1}, U_{t-1}, B)),$

Σ_θ is a linear function of the input composed with the softplus function to ensure positivity. The IEF is $\mu_\theta(S_{t-1}, U_{t-1}, B)$, where $S_{t-1} = X_{t-1}$, captures variation in X_t. If μ_θ is a linear function of the concatenation of its inputs, we refer to the model as $\mathbf{FOMM}_{\text{Linear}}$. $\mathbf{FOMM}_{\text{NL}}$ refers to the model where μ_θ is a two-layer neural network.

Maximum Likelihood Estimation of θ: We learn the model by maximizing $\max_\theta \log p(\mathbf{X}|\mathbf{U}, B)$. Using the factorization structure in the joint distribution of the generative model, we obtain: $\log p(\mathbf{X}|\mathbf{U}, B) = \sum_{t=1}^{T} \log p(X_t|X_{t-1}, U_{t-1}, B)$. Each $\log p(X_t|X_{t-1}, U_{t-1}, B)$ is estimable as the log-likelihood of the observed multi-variate X_t under a Gaussian distribution whose (diagonal) variance is a function $\Sigma_\theta(X_{t-1}, U_{t-1}, B)$ and whose mean is given by the IEF, $\mu_\theta(X_{t-1}, U_{t-1}, B)$. Since each $\log p(X_t|X_{t-1}, U_{t-1}, B)$ is a differentiable function of θ, its sum is differentiable as well, and we may use automatic differentiation to derive gradients of the log-likelihood with respect to θ in order to perform gradient ascent. When any dimension of X_t is missing, that dimension's log-likelihood is ignored (corresponding to marginalization over that random variable) during learning.

7.2.2 Gated Recurrent Neural Network (GRUs)

GRUs (Chung *et al.* , 2014) are auto-regressive models of sequential observations i.e. $p(\mathbf{X}|\mathbf{U}, B) = \prod_{t=1}^{T} p(X_t|X_{<t}, U_{<t}, B))$. GRUs use an intermediate hidden state $h_t \in \mathbb{R}^H$ at each time-step as a proxy for what the model has inferred about the sequence of data until t. The GRU dynamics govern how h_t evolves via an update gate F_t, and a reset gate R_t:

$$F_t = \sigma(W_z \cdot [X_t, U_t, B] + V_z h_{t-1} + b_z), \quad R_t = \sigma(W_r \cdot [X_t, U_t, B] + V_r h_{t-1} + b_r)$$
$$h_t = F_t \odot h_{t-1} + (1 - F_t) \odot \tanh(W_h \cdot [X_t, U_t, B] + V_h(R_t \odot h_{t-1}) + b_h)$$

$\theta = \{\ W_z, W_r, W_h \in \mathbb{R}^{H\times(M+L+J)}; V_z, V_r, V_h \in \mathbb{R}^{H\times H}; b_z, b_r, b_h \in \mathbb{R}^H\}$ are learned parameters and σ is the sigmoid function. The effect of interventions may be felt in any of the above time-varying representations and so the IEF in the GRU is distributed across the computation of the forget gate, reset gate and the hidden state, i.e. $S_t = [F_t, R_t, h_t]$. We refer to this model as **GRU**.

Maximum Likelihood Estimation: We learn the model by maximizing $\max_\theta \log p(\mathbf{X}|\mathbf{U}, B)$. Using the factorization structure in the joint distribution of the generative model, we obtain: $\log p(\mathbf{X}|\mathbf{U}, B) = \sum_{t=1}^{T} \log p(X_t|X_{<t}, U_{<t}, B)$. At each point in time the hidden state of the GRU, h_t, summarizes $X_{<t}, U_{<t}, B$. Thus, the model assumes $X_t \sim \mathcal{N}(\mu_\theta(h_t), \Sigma_\theta(h_t))$.

At each point in time, $\log p(X_t|X_{<t}, U_{<t}, B)$ is the log-likelihood of a multi-variate Gaussian distribution which depends on θ. As before, we may use automatic differentiation to derive gradients of the log-likelihood with respect to θ in order to perform gradient ascent. When any dimension of X_t is missing, that dimension's log-likelihood is ignored (corresponding to marginalization over that random variable) during learning.

7.2.3 State Space Models (SSMs)

SSMs capture longer-term dependencies in sequential data via a time-varying latent state, as in Figure 7-1 (right). Z_t is a low-dimensional representation of the high-dimensional X_t. We experiment with Deep Markov Models (Krishnan *et al.* , 2017):

$$p(\mathbf{X}|\mathbf{U}, B) = \int_Z \prod_{t=1}^{T} p(Z_t|Z_{t-1}, U_{t-1}, B; \theta) p(X_t|Z_t; \theta) dZ$$
$$Z_t \sim \mathcal{N}(\mu_\theta(Z_{t-1}, U_{t-1}, B), \Sigma_\theta^t(Z_{t-1}, U_{t-1}, B)), \quad X_t \sim \mathcal{N}(\kappa_\theta(Z_t), \Sigma_\theta^e(Z_t))$$

$\Sigma_\theta^t, \Sigma_\theta^e, \kappa_\theta(Z_t)$ are linear functions of a concatenation of their inputs composed with the softplus function to ensure positivity. The IEF is $\mu_\theta(S_{t-1}, U_{t-1}, B)$, where $S_{t-1} = Z_{t-1}$. **SSM**$_{\text{Linear}}$ and **SSM**$_{\text{NL}}$ refer to models where μ_θ is linear and non-linear (two-layer neural network), respectively.

Maximum Likelihood Estimation: We learn the model by maximizing $\max_\theta \log p(\mathbf{X}|\mathbf{U}, B)$. Using the factorization structure in the joint distribution of the generative model, we obtain: $\log p(\mathbf{X}|\mathbf{U}, B) = \sum_{t=1}^{T} \log p(X_t|X_{t-1}, U_{t-1}, B)$. Each $\log p(X_t|X_{t-1}, U_{t-1}, B)$ is estimable as the log-likelihood of the observed multi-variate X_t under a Gaussian distribution whose (diagonal) variance is a function $\Sigma_\theta(X_{t-1}, U_{t-1}, B)$ and whose mean is given by the IEF, $\mu_\theta(X_{t-1}, U_{t-1}, B)$. Since each $\log p(X_t|X_{t-1}, U_{t-1}, B)$ is a differentiable function of θ, its sum is differentiable as well, and we may use automatic differentiation to derive gradients of the log-likelihood with respect to θ in order to perform gradient ascent. When any dimension of X_t is missing, that dimension's log-likelihood is ignored (corresponding to marginalization over that random variable) during learning.

7.2.4 Missing data

Clinical biomarkers may not always be observed. When a variable in the conditioning set is missing, e.g. X_{t-1} when evaluating the FOMM's IEF μ_θ, we use a proxy for X_{t-1} obtained via forward-fill imputation. In Figure 7-1 (left), the dots in red for serum IgG are forward filled from their previous values. When evaluating the likelihood, if X_t is missing, we marginalize it out, i.e. it does not contribute towards the likelihood of the data. We assume that $\mathbf{U}, B$ are always observed.

7.2.5 Pharmacokinetic-Pharmacodynamic (PK-PD) models

To gain improvements in sample complexity when doing unsupervised learning in low-data regimes, we need domain expertise to parameterize $\mathbb{IEF}(S_{t-1}, U_{t-1}, B)$. We turn to the pharmacokinetics (PK) & pharmacodynamics (PD) literature. Pharmacokinetics is concerned with how drugs move in the body, and pharmacodynamics studies the body's response to drugs. We review three PK-PD models in this section.

PK-PD models typically comprise two components: the first is a proxy for disease burden, and the second is the treatment effect, or rather the effect that treatment has on disease burden. Disease burden, denoted $S(t)$, is quantified in different ways depending on the disease. Models of chronic disease progression might track a single clinical biomarker as a proxy for disease burden. We will denote the effect of treatment by $E(t)$. Unless otherwise specified, the quantities we describe in this section are real-valued scalars, which may be constrained to be positive.

Linear A linear model is one of the simplest disease progression models that is used for tracking the dynamics of tumor volume $S(t)$ (Klein, 2009):

$$S(t) = S(0) + (\alpha + E(t)) \cdot t,$$

Here $E(t)$ is the scalar, real-valued treatment dose. Linear models have also been been used successfully to describe progression of biomarkers in neurological disorders such as Alzheimer's disease (Doyle *et al.* , 2014), and Huntington's disease (Warner & Sampaio, 2016).

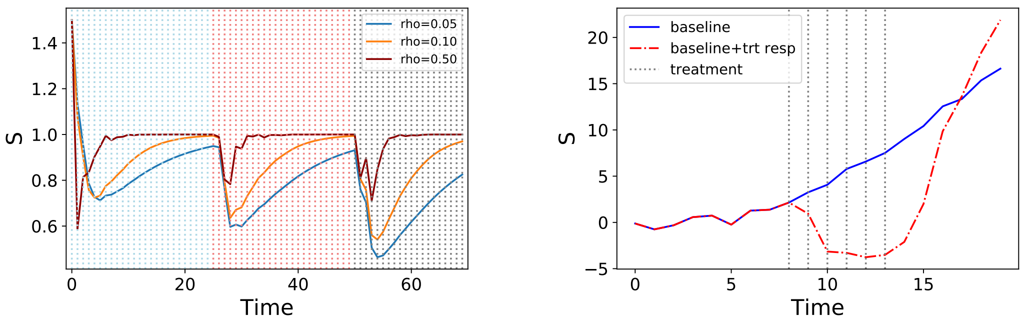


Figure 7-2: **Pharmacodynamic-Pharmacokinetic Treatment Effect Functions:** Visualizing PK/PD treatment response models. Curves denote the scalar biomarker being modeled and vertical lines denote treatment. *Left:* Log Cell Kill. The various curves (green, yellow, red) represent different parameterizations of the function. Here, (for visualization purposes) a single treatment is always present throughout time, but may be under a different line of therapy based on the shaded region. For each line, a sharp decline is followed by a rise in tumor volume, prompting a change in therapy line. Each curve corresponds to distinct rates of biomarker growth, parametrized by ρ. *Right:* Biomarker value under the Treatment Exponential model. After maintaining the response with treatments, a regression towards baseline (in blue; depicting what would have happened had no treatment been prescribed) occurs when treatment is stopped.

Log-Cell Kill The log-cell kill hypothesis (Norton, 2014) states that a given dose of chemotherapy results in killing a constant fraction of tumor cells rather than a constant number of cells. The Log Cell Kill model, a popular choice for modeling the tumor size in solid cell tumors(Lim, 2018; West & Newton, 2017), can be described by the following ordinary differential equation (ODE),

$$\frac{dS(t)}{dt} = -\beta_c C(t) S(t),$$

where $C(t)$ is the concentration of a chemotherapeutic drug over time. $C(t)$ is specified as follows: $C(t) = C_{max} e^{\frac{-\log(2)}{\text{half-life}} t}$, where C_{max} is the maximum concentration of the drug (i.e. the dose at which the drug was given), half-life is the half-life of the drug, and β_c is a parameter that represents the drug effect on tumor size .

Variants of the model also incorporate the kinetics of tumor growth (Evain & Benzekry, 2016; Lim, 2018; Grassberger & Paganetti, 2016), where the evolution of tumor volume, S, is described via an ODE:

$$\frac{dS(t)}{dt} = \rho \log\left(\frac{K}{S(t)}\right).$$

ρ, the growth rate, and K, the tumor carrying capacity, determine the growth curve of the tumor. An analytic expression for the tumor dynamics of the log cell kill model that incorporates tumor growth is:

$$S(t) = S(t-1) \cdot (1 + \rho \log(K/S(t-1)) - \beta_c C(t)), \tag{7.1}$$

In Figure 7-2 (left), we show an example of the dynamics of the log-cell kill model combined with this form of Gompertzian growth.

Treatment Exponential The third treatment effect model is inspired by disease progression models for chronic diseases. This model was used by Xu *et al.* (2016) to estimate individualized treatment-effect curves in patients with Chronic Kidney Disease (CKD) (Xu *et al.* , 2016). Given a treatment, $\mathbf{a}_\tau$, let $E(t-\tau)$ be the response curve for $t \geq \tau$ of administering this treatment regimen at time τ. $E(t)$ is parametrized as

$$E(t) = \begin{cases} b_0 + \alpha_1/[1 + \exp(-\alpha_2(t - \frac{\gamma_l}{2}))], \text{if } 0 \leq t < \gamma_l \\ b_l + \alpha_0/[1 + \exp(\alpha_3(t - \frac{3\gamma_l}{2}))], \text{if } t \geq \gamma_l \end{cases} \tag{7.2}$$

with six free parameters: $\{\alpha_1, \alpha_2, \alpha_3, \gamma_l, b_0, b_l\}$. $\alpha_1 \in \mathbb{R}$ represents the maximum value and its sign determines whether there is an increase or decrease of lab markers in response to treatment. $\alpha_2 \in (0,1)$ and $\alpha_3 \in (0,1)$ model the steepness of the curves. Finally, $\gamma_l \in \mathcal{R}$ denotes the switching point. The motivation behind using this functional form of $g(t)$ is that it admits a flexible "U"-shaped curve, as shown in Figure 7-2, by concatenating two sigmoid curves. Allowing the parameters of the function to vary alters the switching point between the two sigmoid curves as well as the slope of ascent or descent. Thus, this function can capture whether a treatment causes a patient's lab value to increase or decrease over time as well as the rate at which it does so before converging to a stable value. We visualize the ability of this model to capture "U"-shaped intervention effects in Figure 7-2 (right).

7.3 Intervention Effect Functions for clinical data

We are now ready to describe the way in which we construct $\mathbb{IEF}(S_{t-1}, U_{t-1}, B)$. This task is difficult since unlike other domains, we lack good mechanistic models for how combinations of drugs affect multiple biomarkers in the short and long-term. However, there is structure in clinical data that can aid us. Our exposition, moving forward, will focus on chronic diseases. First, we recognize that treatments for chronic diseases are not given in isolation but are often prescribed as parts of contiguous plans of treatment known as *lines of therapy*. Second, chronic diseases, despite differences in how they manifest, may share similarities in mechanisms behind how drugs affect their progression. To that end, we posit that functions known to capture the mechanistic effect of drugs contain knowledge we can transfer to design IEFs for chronic diseases where we lack good mechanistic knowledge. We will refer to IEFs for diseases with known mechanisms as *domain expert modules* and propose a neural architecture that trades off between them. Our architecture does so by using data to guide how important a domain expert is in deciding how a representation varies over time. To our knowledge, both of the above have not been studied in the context of unsupervised learning of high-dimensional clinical data. We will use Figure 7-1 (left), which depicts data from a patient suffering from a chronic disease, as a guide in our discussion.

7.3.1 Capturing lines of therapy with local and global clocks

Many representation learning based approaches (Choi *et al.* , 2016b; Krishnan *et al.* , 2017; Choi *et al.* , 2016a; Lipton *et al.* , 2015) use binary or continuous indicators to designate the prescription or dosage of drugs in U_t. However, chronic diseases are treated with more than one drug (combination therapy) following clinically accepted guidelines known as lines of therapy. For example, first line therapies often represent combinations prioritized due to their efficacy in clinical trials; subsequent lines may be decided by clinician experience. Lines of therapy index treatment plans that span multiple time-steps and are often laid out by clinicians at first diagnosis. We incorporate line of therapy as one-hot vectors in $U_t[: K]\ \forall t$ where K is the maximal line of therapy. In doing so, we implicitly capture the clinician's *intention* when prescribing drug combinations.

Lines of therapy typically change when drug combinations fail, or due to adverse side effects. In Figure 7-1 (left), the doctor may change the line of therapy once the combinations of drugs cease to be effective in modulating the behavior of serum IgG. By using line of therapy, an $\mathbb{IEF}(S_{t-1}, U_{t-1}, B)$ can infer vital information such as how long a patient has been on a line in the representation S_t. We conjecture, however, that using line of therapy is not enough in a low-data setting. Neural Turing Machines Graves *et al.* (2014) can learn to count occurrences of observations in their history, but may fail when data is at a premium. (Che *et al.* , 2018b) use time since the last observation to help RNNs learn well when data is missing. (Koutnik *et al.* , 2014) partition the hidden states in an RNNs so they are updated at different time-scales. To explicitly enforce our IEFs can capture time since change in line of therapy, we use *clocks* to track the time elapsed since an event.

We augment our interventional vector, U_t, with two more dimensions. A global clock, gc, captures time elapsed since $T = 0$, i.e. $U_t[K] = \mathrm{gc}_t = t$. A local clock, lc, captures time elapsed since a line of therapy began; i.e. $U_t[K+1] = \mathrm{lc}_t = t - p_t$ where p_t denotes the index of time when the line last changed. $\mathbb{IEF}(S_{t-1}, U_{t-1}, B)$ can, by using the local clock, modulate S_t to capture patterns such as: the longer a line of therapy is deployed, the less or (more) effective it may be. For the patient in Figure 7-1 (left), we can see that the first dimension of $\mathbf{U}$ denoting line of therapy would be $[0,0,0,0,1,1,2,2,2]$. Line 0 was used four times, line 1 used twice, line 2 used thrice. Then $p = [0,0,0,0,4,4,6,6,6,6]$, $gc = [0,1,2,3,4,5,6,7,8,9]$ and $lc = [0,1,2,3,0,1,0,1,2,3]$. Next, we highlight how these clocks are put to use.

7.3.2 Domain expert IEF modules for clinical data

There are a few challenges, though, to the use of PK-PD models in parameterizing $\mathbb{IEF}(S_{t-1}, U_{t-1}, B)$. First, PK-PD models are often constructed to quantify dose response for physiological markers at the micro-scale, such as the evolution of the volume of a non-small cell lung tumor (Geng *et al.* , 2017). However, such markers may not be among the multivariate biomarkers used to characterize disease progression in clinical practice. We hypothesize that given suitably diverse observations in X_t, one or more of the dimensions of the representation S_t can implicitly capture the value of the unobserved physiological marker. Second, PK-PD models are often univariate, modeling how a single marker changes in response to variation in the drug. Our work designs functions to generalize PK-PD models to work with multivariate representations, S_{t-1}. The final challenge is knowing *which* PK-PD models to make use of. We describe three new IEFs, each using PK-PD dynamics of a different disease, designed to capture properties we may expect in representations of chronic disease data.

Saturated Linear: (Klein, 2009) study the use of linear functions in characterizing dose-responses in solid cancerous tumors. To allow for the representations we use to increase or decrease as a linear function of the treatments and the line of therapy, we propose the use of a (bounded) linear effect.

$$g_1(S_{t-1}, U_{t-1}, B) = \text{LIN}(S_{t-1}, U_{t-1}, B) = S_{t-1} \odot \tanh(b_{\text{lin}} + W_{\text{lin}}[U_{t-1}, B]) \tag{7.3}$$

where $b_{\text{lin}} \in \mathbb{R}^Q, W_{\text{lin}} \in \mathbb{R}^{Q\times(L+J)}$.

Log-Cell Kill: The log-cell kill model is a classical model of tumor volume in solid cell tumors (Lim, 2018; West & Newton, 2017). It is derived from the log-cell kill hypothesis, which states that administering a dose of chemotherapy kills a constant fraction of tumor cells regardless of the size of the tumor. While chronic diseases may not have a single observation that characterizes the organ system (akin to tumor volume), we hypothesize that representations, S_t, of the observed clinical biomarkers may benefit from mimicking the dynamics exhibited by tumor volume when exposed to chemotherapeutic agents. Therefore, we design the following IEF:

$$g_2(S_{t-1}, U_{t-1}, B) = \text{LC}(S_{t-1}, U_{t-1}, B) = S_{t-1} \odot (1 - \rho \log(S_{t-1}^2) - \beta \exp(-\delta \cdot \text{lc}_{t-1})), \tag{7.4}$$

where $\beta = \tanh(W_{lc}U_{t-1} + b_{lc})$. $W_{lc} \in \mathbb{R}^{Q\times L}, b_{lc} \in \mathbb{R}^Q, \delta \in \mathbb{R}^Q$ and $\rho \in \mathbb{R}^Q$ are learned.

Treatment Exponential: (Xu *et al.* , 2016) propose a Bayesian nonparameteric model of creatinine, a marker of kidney function, in patients suffering from Chronic Kidney Disease. The model can track the dynamics of creatinine due to treatment, but is limited to operating on a single biomarker. We extend their IEF to modeling high dimensional representations, S_t, making use of information in the lines of therapy via the clocks (Section 7.3.1). We refer to this as the Treatment Exponential IEF.

$$g_3(S_{t-1}, U_{t-1}, B) = \text{TE}(\cdot) = \begin{cases} b_0 + \alpha_{1,t-1}/[1 + \exp(-\alpha_{2,t-1}(\text{lc}_{t-1} - \frac{\gamma_l}{2}))], \text{if } 0 \leq \text{lc}_{t-1} < \gamma_l \\ b_l + \alpha_{0,t-1}/[1 + \exp(\alpha_{3,t-1}(\text{lc}_{t-1} - \frac{3\gamma_l}{2}))], \text{if } \text{lc}_{t-1} \geq \gamma_l \end{cases} \tag{7.5}$$

The parameters of this model have meaning. $\alpha_{1,t-1} = W_d[S_{t-1}, U_{t-1}, B] + b_d$, where $W_d \in \mathbb{R}^{Q\times(Q+L+J)}, b_d \in \mathbb{R}^Q$ is used to control whether TE is positive or negative. $\alpha_{2,t-1}, \alpha_{3,t-1}$, and γ_l control the steepness and duration of the effect. We restrict these characteristics to be similar for drugs administered under the same strategy (or line of therapy). Thus, we parameterize: $[\alpha_2, \alpha_3, \gamma]_{t-1} = \sigma(W_e \cdot U_{t-1}[0] + b_e)$. If there are three lines of therapy, $W_e \in \mathbb{R}^{3\times3}, b_e \in \mathbb{R}^3$ and the biases, $b_0 \in \mathbb{R}^Q$ and $b_l \in \mathbb{R}^Q$, are learned. Finally, $\alpha_{0,t-1} = (\alpha_{1,t-1} + 2b_0 - b_l)/(1 + \exp(-\alpha_{3,t-1}\gamma/2))$ will ensure that the effect peaks at $t = \text{lc}_t + \gamma$.

7.3.3 PK-PD Intervention Effect Function

Having proposed three functions (g_1, g_2, g_3) from diverse domains, we discuss the construction of $\mathbb{IEF}(S_{t-1}, U_{t-1}, B)$.

Grounded Mixture of Domain Experts (GroMoDE): In general, if the user specifies D domain expert modules, $g_d : \mathbb{R}^{Q+L+J} \mapsto \mathbb{R}^Q, d \in [1, D]$, then we'd like each domain expert module to capture a *different* way in which the representation, S_{t-1}, responds to interventions U_{t-1}. Given inputs S_{t-1}, U_{t-1}, B, and a gating function $\boldsymbol{\delta}(S_{t-1}, U_{t-1}, B) = [\delta_1, \delta_2, \dots, \delta_K]$ that (optionally) may be a function of the inputs, we propose the following IEF:

$$\mathbb{IEF}(S_{t-1}, U_{t-1}, B) = \sum_{d=1}^{D} \sigma(\boldsymbol{\delta})_d \odot g_d(S_{t-1}, U_{t-1}, B) \tag{7.6}$$

$\sigma(\boldsymbol{\delta})_i$ in Equation 7.6 refers to taking the softmax of $\boldsymbol{\delta}$ and then selecting the dth element of the resulting vector. The intervention effect term $\mathbb{IEF}(S_{t-1}, U_{t-1}, B)$ is a soft-mixture of domain expert modules weighted by δ_i. We refer to this architecture as the Grounded Mixture of Domain Experts (GroMoDE). Unlike the popular Mixture-of-Experts (MoE) architecture (Jacobs *et al.* , 1991; Jordan & Jacobs, 1994), each g_d we use does *not* come from the same hypothesis class, but rather has a functional form *grounded* in the hypothesis class represented by a domain expert module. The architecture multiplexes between various domain expert modules to determine intervention effects, allowing the data to guide which domain expert is appropriate for each dimension of S_t. The gating can be adapted based not only on S_{t-1}, but also on the line of therapy at that time and the time-elapsed from the beginning of the line.

$\mathbb{PK}$-$\mathbb{PD}_{\mathbb{IEF}}$: We use the GroMoDe to parameterize the effect of an intervention. We assume the effect is additive in representation space and that the representation S_t will be one wherein the assumption of additivity holds. Using g_1, g_2, g_3 from Section 7.3.2:

$$\begin{aligned}&\mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}(S_{t-1}, U_{t-1}, B) = S_{t-1}+ \\ &[\sigma(\boldsymbol{\delta})_1 \odot \mathrm{LIN}(S_{t-1}, U_{t-1}) + \sigma(\boldsymbol{\delta})_2 \odot \mathrm{LC}(S_{t-1}, U_{t-1}) + \sigma(\boldsymbol{\delta})_3 \odot \mathrm{TE}(S_{t-1}, U_{t-1})]\end{aligned} \tag{7.7}$$

Unsupervised Models of Clinical Data: The $\mathbb{PK}$-$\mathbb{PD}_{\mathbb{IEF}}$ can be instantiated in each of the three sequential models we highlighted in Section 7.2 as follows:

1. **FOMM**$_{\text{PK-PD}}$ sets $\mu_\theta(X_{t-1}, U_{t-1}, B) = \mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}(X_{t-1}, U_{t-1}, B)$

2. **SSM**$_{\text{PK-PD}}$ sets $\mu_\theta(Z_{t-1}, U_{t-1}, B) = \mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}(Z_{t-1}, U_{t-1}, B)$

3. **GRU**$_{\text{PK-PD}}$ is obtained by modifying the dynamics of the GRU. We chunk the output of the IEF, $o_t = \mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}(X_{t-1}, U_{t-1}, B)$, into three equally sized vectors: o_t^f, o_t^r, o_t^h. Then,

$$F_t = \sigma(o_t^f + V_z h_{t-1} + b_z), \quad R_t = \sigma(o_t^r + V_r h_{t-1} + b_r)$$
$$h_t = F_t \odot h_{t-1} + (1 - F_t) \odot \tanh(o_t^h + V_h(R_t \odot h_{t-1}) + b_h)$$

7.4 Datasets

We describe the construction and curation of a synthetic dataset to mimic the progression of disease in patients and a real-world dataset comprising patients undergoing treatment for cancer. We will evaluate our methods on these data.

7.4.1 Synthetic data

Each synthetic patient is assigned $B \in \mathbb{R}^6$. B determines how biomarkers, $X_t \in \mathbb{R}^2$, behave in the absence of treatment. $U_t \in \mathbb{R}^4$, comprises the line of therapy ($K = 2$), the local clock, and a single binary variable indicating when treatment is prescribed. We train on 50/1000 samples and evaluate on five held-out sets of size 50000.

Below, we outline the general principles that the synthetic data we design is based on:

- We sample six random baseline values from a standard normal distribution.
- Two of the six baseline values determine the natural (untreated) progression of the two-dimensional longitudinal trajectories. They do so as follows: depending on which quadrant the baseline data lie in, we assume that the patient has one of four subtypes.
- Each of the four subtypes typifies different patterns by which the biomarkers behave such as whether they both go up, both go down, one goes up, one goes down etc. To see a visual example of this, we refer the reader to Figure 7-3 (left).

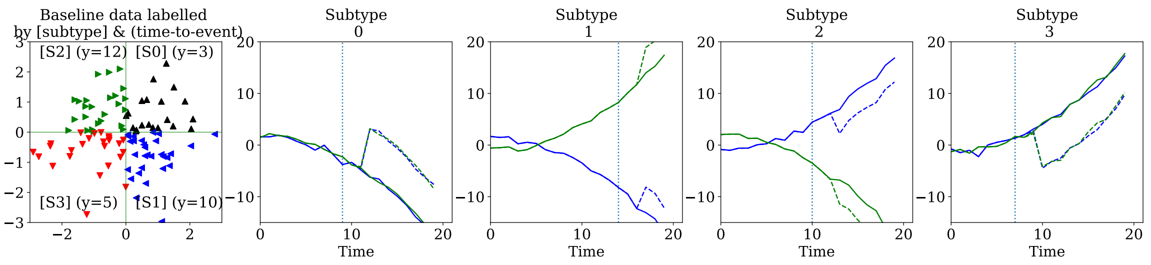


Figure 7-3: **Visualization of synthetic data:** Left: A visualization of "patient"'s baseline data (colored and marked by patient subtype). Right four plots: Examples of patient's longitudinal trajectories along with treatment response. The blue and green longitudinal data denote two diffrent patient biomarkers. Gray-dotted line represents intervention. The subtypes may, optionally, be correlated with patient outcomes as highlighted using the values of y. We do not use the outcomes in this chapter, but do so later in the thesis.

Baseline The generative process for the baseline covariates is $B \sim \mathcal{N}(0; \mathbb{I}); B \in \mathbb{R}^6$.

Treatments (Interventions): There is a single drug (denoted by a binary random variable) that may be withheld (in the first line of therapy) or prescribed in the second line of therapy. $d_i \sim$ Unif.$[0, 18]$ denotes when the single drug is administered (and the second line of treatment begins). d_i is the point at which the local clock resets. We can summarize the generative process for the treatments as follows:

$$\begin{aligned}
& d \sim \text{Unif.}[0, 18] \\
& \text{line}_t = 0 \text{ if } t < d \text{ and } 1 \text{ otherwise} \\
& lc_t[0] = 1 \text{ if } t < d \text{ and } 0 \text{ otherwise} \\
& lc_t[1] = 0 \text{ if } t < d \text{ and } 1 \text{ otherwise}
\end{aligned} \tag{7.8}$$

where $lc_t[0], lc_t[1]$ denote the one-hot encoding for line of therapy. Next we describe the intervention effect function that we use. The functional form of TE$(\cdot)$ is re-stated below for convenience,

$$\text{TE}(\text{lc}_t) = \begin{cases} b_0 + \alpha_1/[1 + \exp(-\alpha_2(\text{lc}_t - \frac{\gamma_l}{2}))], \text{if } 0 \leq \text{lc}_t < \gamma_l \\ b_l + \alpha_0/[1 + \exp(\alpha_3(\text{lc}_t - \frac{3\gamma_l}{2}))], \text{if } \text{lc}_t \geq \gamma_l \end{cases} \tag{7.9}$$

The parameters that we use to generate the data are: $\alpha_2 = 0.6, \alpha_3 = 0.6, \gamma_l = 2, b_l = 3$, and $\alpha_1 = [10, 5, -5, -10]$, which we vary based on patient subtype. We set $\alpha_0 = (\alpha_1 + 2b_0 - b_l)/(1 + \exp(-\alpha_3\gamma_l)/2)$ to ensure that the treatment effect peaks at $t = \text{lc}_t + \gamma_l$ and $b_0 = -\alpha_1/(1 + \exp(\alpha_2 \cdot \gamma_l/2))$ for attaining TE$(0) = 0$.

Biomarkers: We are now ready to describe the full generative process of the longitudinal biomarkers.

$$
\begin{aligned}
&\text{Recall: } B_{1\ldots,6} \sim \mathcal{N}(0; I), \\
&f_d(t) = 2 - 0.05t - 0.005t^2, \quad f_u(t) = -1 + 0.0001t + 0.005t^2, \\
&X_1(t); X_2(t) = \\
&\begin{cases}
f_d(t) + \mathrm{TE}(\mathrm{lc}_t) + \mathcal{N}(0, 0.25);\ f_d(t) + \mathrm{TE}(\mathrm{lc}_t) \\
\qquad + \mathcal{N}(0, 0.25), B_1 \geq 0, B_2 \geq 0 \text{ if subtype 1} \\
f_d(t) + \mathrm{TE}(\mathrm{lc}_t) + \mathcal{N}(0, 0.25);\ f_u(t) + \mathrm{TE}(\mathrm{lc}_t) \\
\qquad + \mathcal{N}(0, 0.25), B_1 \geq 0, B_2 < 0 \text{ if subtype 2} \\
f_u(t) + \mathrm{TE}(\mathrm{lc}_t) + \mathcal{N}(0, 0.25);\ f_d(t) + \mathrm{TE}(\mathrm{lc}_t) \\
\qquad + \mathcal{N}(0, 0.25), B_1 < 0, B_2 \geq 0 \text{ if subtype 3} \\
f_u(t) + \mathrm{TE}(\mathrm{lc}_t) + \mathcal{N}(0, 0.25);\ f_u(t) + \mathrm{TE}(\mathrm{lc}_t) \\
\qquad + \mathcal{N}(0, 0.25), B_1 < 0, B_2 < 0 \text{ if subtype 4,}
\end{cases}
\end{aligned}
\tag{7.10}
$$

Intuitively, the above generative process captures the idea that without any effect of treatment, the biomarkers follow the patterns implied by the subtype (encoded in the first two dimensions of the baseline data). However the effect of interventions is felt more prominently after the d, the random variable denoting time at which treatment was prescribed.

7.4.2 Multiple Myleoma - ML-MMRF

Multiple myeloma is a rare and incurable plasma cell cancer with nearly $30,000$ new cases every year in the United States. The Multiple Myeloma Research Foundation (MMRF) CoMMpass study releases de-identified clinical data for 1143 patients suffering from multiple myeloma, an incurable plasma cell cancer. We will release code that pre-processes features from the CoMMpass study files to construct ML-MMRF, a publicly available dataset with clinical and interventional data alongside rich genetic profiles of patients.

Inclusion Criteria: All patients are aligned to the start of treatment, which is made according to current standard of care (not random assignment). To enroll in the CoMMpass study, patients must be newly diagnosed with symptomatic multiple myeloma, which coincides with the start of treatment. Patients must be eligible for treatment with an immunomodulator or a proteasome inhibitor, two of the most common first line drugs, and they must begin treatment within 30 days of the baseline bone marrow evaluation (Multiple Myeloma Research Foundation & others, 2011).

Features

Genomic Data: RNA-sequencing of CD38+ bone marrow cells was available for 769 patients. Samples were collected at initiation into the study, pre-treatment. For these patients, we used the Seurat package version 2.3.4 Butler *et al.* (2018) in R to identify variable genes, and we then limit downstream analyses to these genes. We use principal component analysis (PCA) to further reduce the dimensionality of the data, and the projection of each patient's gene expression on to the first 40 principal components serves as the genetic features used in our model.

Baseline Data B: Baseline data includes genetic PCA scores, lab values at the patient's first visit, gender, age, and the revised ISS stage. The baseline data also includes binary variables detailing the patient's myeloma subtype, including whether or not they have heavy chain myeloma, are IgG type, IgA type, IgM type, kappa type, or lambda type. Additionally, the following labs are measured at baseline, as well as longitudinally at subsequent visits: absolute neutrophil count (x10^9/l), albumin (g/l), blood urea nitrogen (mmol/l), calcium (mmol/l), serum creatinine (umol/l), glucose (mmol/l), hemoglobin (mmol/l), serum kappa (mg/dl), serum m protein (g/dl), platelet count x10^9/l, total protein (g/dl), white blood count x10^9/l, serum iga (g/l), serum igg (g/l), serum igm (g/l), serum lambda (mg/dl).

Longitudinal Data X: Longitudinal data is measured approximately every 2 months and includes lab values and treatment information. The biomarkers are real-valued numbers whose values evolve over time. They include: absolute neutrophil count (x10^9/l), albumin (g/l), blood urea nitrogen (mmol/l), calcium (mmol/l), serum creatinine (umol/l), glucose (mmol/l), hemoglobin (mmol/l), serum kappa (mg/dl), serum m protein (g/dl), platelet count x10^9/l, total protein (g/dl), white blood count x10^9/l, serum iga (g/l), serum igg (g/l), serum igm (g/l), serum lambda (mg/dl).

Treatment information U: This includes the line of therapy (we group all lines beyond line 3 as line 3+) the patient is on at a given point in time, and the local clock denoting the time elapsed since the last line of therapy. We also include the following treatments as (binary, indicating prescription) features in our model: lenalidomide, dexamethasone, cyclophosphamide, carfilzomib, bortezomib. The aforementioned are the top five drugs by frequency in the MMRF dataset.

Data processing

Longitudinal biomarkers $\mathbf{X}$: Labs are first clipped to five times the median value to correct for outliers or data errors in the registry. They are then normalized to their healthy ranges (obtained via a literature search) as (unnormalized labs - healthy minimum value) / (healthy maximum value - healthy minimum value), and then multiplied by a scaling factor of lab-dependent scaling factor to ensure that most values lie within the range $[-8, 8]$. This dataset has significant missingness, with $\sim 66\%$ of the longitudinal markers missing. In addition, there is right censorship in the dataset, with around 25% of patients getting censored over time. Missing values are represented as zeros but a separate mask tensor where 1 denotes observed and 0 denotes missing is used to marginalize out missing variables during learning.

Baseline B: The biomarkers in the baseline are clipped to five times their median values. Patients without gene expression data (in the PCA features) are assigned the average normalized PCA score of their 5 nearest neighbors, using the Minkowski distance metric calculated on FISH features, ISS stage, and age.

Our results are obtained using 5-fold cross evaluation (60/20/20 split) with cohorts balanced on age and overall survival time. As a representative example, one fold has 439 train, 211 validation, and 301 test examples. The median length in each of these sets is 11-12 time steps. There is missingness in the biomarkers, with 66% of the observations missing.

7.5 Evaluation

We answer the following questions: What benefits does $\mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}$ confer upon unsupervised models of clinical data? Which model families benefit the most? How does the GroMoDE architecture aid with introspection into the model's functionality? We study these questions on two datasets *in the low-data regime* ($\sim$ 100-1000 samples), where little prior work exists. The data faithfully represent the multi-modal nature by which chronic disease progression is tracked in the clinic.

Experimental setup: All models are trained to solve $\arg\min_\theta -\log p(\mathbf{X}|\mathbf{U}, B; \theta)$ via stochastic gradient descent using ADAM (Kingma & Ba, 2014) with a learning rate of 0.001 for 5000 epochs. Latent variable models minimize a bound on this quantity. L1 and L2 regularization is applied in one of two ways: either we regularize all model parameters, or we regularize all weight matrices except those associated with $\boldsymbol{\delta}$ in Eq. 7.7. We search over regularization strengths of $0.01, 0.1, 1, 10$. For the RNN and the state space models, we vary the hidden dimensions to be between $100, 250$, and 500. We use five-fold cross validation (with early stopping) for selecting the best hyper-parameters.

Table 7.1: **Synthetic data:** Lower is better. We report held-out negative log likelihood (or a bound on it for SSM models) with std. dev. on several model families to study generalization in the synthetic setting.

Training Set Size	FOMM Linear	FOMM NL	FOMM PK-PD	GRU	GRU PK-PD	SSM Linear	SSM NL	SSM PK-PD
50	71.06 +/- .03	58.80 +/- .03	**56.81 +/- .04**	56.65 +/- .11	**53.49 +/- .04**	64.12 +/- .06	80.82 +/- .09	**63.72 +/- .03**
1000	62.93 +/- .03	**57.16 +/- .03**	57.81 +/- .02	31.09 +/- .02	**29.27 +/- .01**	53.84 +/- .02	44.75 +/- .02	**44.57 +/- .03**

Table 7.2: **ML-MMRF:** Higher is better. Each number is the fraction (with std. dev.) of held-out patients for which the model that uses $\mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}$ has a lower negative log-likelihood (or bound on it) than a model in the same family that uses a different IEF.

FOMM PK-PD vs. FOMM Linear	FOMM PK-PD vs. FOMM NL	GRU PK-PD vs. GRU	SSM PK-PD vs. SSM Linear	SSM PK-PD vs. SSM NL	SSM PK-PD vs. SSM MOE
0.792 (0.405)	0.668 (0.457)	0.420 (0.489)	0.776 (0.414)	0.750 (0.431)	0.706 (0.454)

Baselines: $\mathbf{FOMM}_{\text{NL}}, \mathbf{SSM}_{\text{NL}}, \mathbf{GRU}$ are represent representation learning models with typical choices for parameterizing the IEF. $\mathbf{FOMM}_{\text{Linear}}$, $\mathbf{SSM}_{\text{Linear}}$, which use linear functions for the IEF, are popular choices when learning models in the low-data regime. $\mathbf{SSM}_{\text{MOE}}$ refers to models whose IEFs (μ_θ (in Section 7.2)) are a mixture of three, 2-layer neural networks, rather than $\mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}$. This control quantifies the effect of grounding each expert using domain knowledge.

7.5.1 Quantitative analysis

Table 7.1 depicts negative log-likelihoods on held-out synthetic data across different models, where a lower number implies better generalization. The non-linearity of the synthetic data makes unsupervised learning a challenge for $\mathbf{FOMM}_{\text{Linear}}$ at 50 samples, allowing $\mathbf{FOMM}_{\text{PK-PD}}$ to easily outperform it. In contrast, $\mathbf{FOMM}_{\text{NL}}$ can capture non-linearities in the data, making it a strong baseline even at 50 samples. Yet, $\mathbf{FOMM}_{\text{PK-PD}}$ outperforms it. At 1000 samples, $\mathbf{FOMM}_{\text{NL}}$ is able to learn

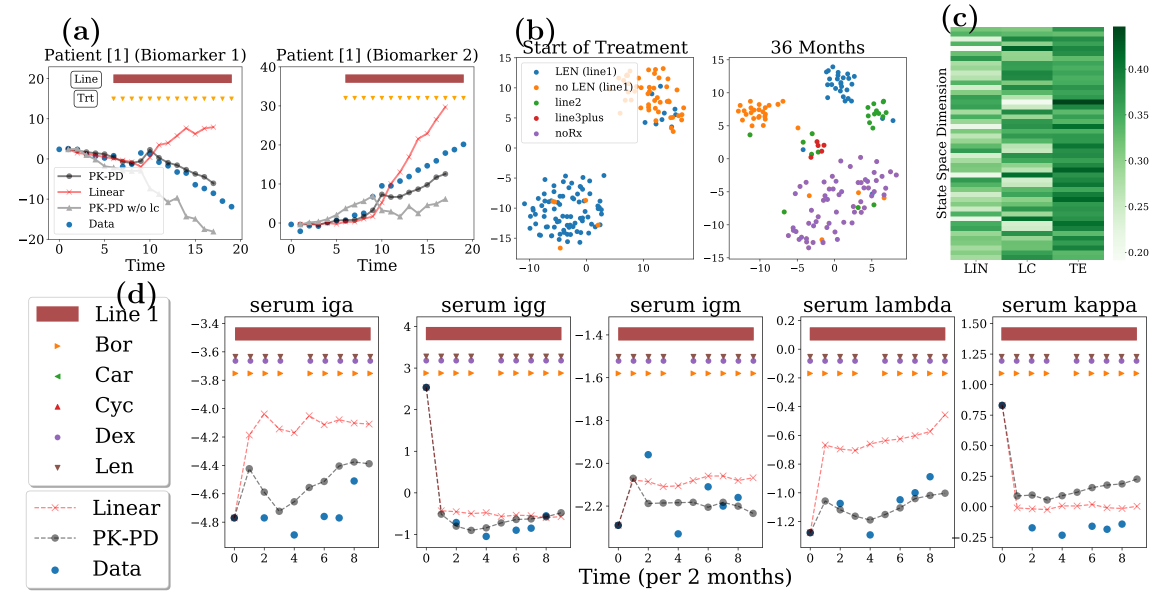


Figure 7-4: **Visualizations of learned SSM models**: (a) *Synthetic*: Forward samples (conditioned only on B) from $\mathbf{SSM}_{\text{PK-PD}}$ (o), $\mathbf{SSM}_{\text{Linear}}$ (x), $\mathbf{SSM}_{\text{PK-PD}}$ without local clocks ($\triangle$), for a single patient. Blue circles (o) denote ground truth. The markers above the trajectories represent treatments prescribed across time. (b) *ML-MMRF*: We visualize the TSNE representations of each held-out patient's α_1 parameter (in the TE module) at the start of treatment and three years in. (c) *ML-MMRF*: For $\mathbf{SSM}_{\text{PK-PD}}$, we visualize weights, $\boldsymbol{\delta}$, on each domain expert module (LIN, LC, TE) across state space dimensions. (d) *ML-MMRF*: Each column is a different biomarker containing forward samples (conditioned only on B) from $\mathbf{SSM}_{\text{PK-PD}}$ (o) and $\mathbf{SSM}_{\text{linear}}$ (x) of a single patient. As in the synthetic samples, blue circles denote ground truth, and the markers above the trajectories represent treatments prescribed across time. y-axis shows biomarker levels (normalized to be between -8 and 8).

enough about the dynamics to improve its performance relative to $\mathbf{FOMM}_{\text{PK-PD}}$. **GRU** is a strong model on this dataset, but in both data regimes, the $\mathbf{GRU}_{\text{PK-PD}}$ improves generalization. Finally, $\mathbf{SSM}_{\text{PK-PD}}$ outperforms $\mathbf{SSM}_{\text{Linear}}, \mathbf{SSM}_{\text{NL}}$ across the board. $\mathbf{SSM}_{\text{NL}}$ overfits quickly on 50 samples but recovers most of its performance when learning with 1000 samples.

Unsupervised learning of the ML-MMRF data is challenging due to high-dimensionality of the (often missing) biomarkers, which vary due to combinations of drugs prescribed over time. For each held-out point, $\Delta_i = 1$ when the negative log-likelihood of that datapoint is lower under a model that uses $\mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}$ and $\Delta_i = 0$ when it is not. In Table 7.2, we report $\frac{1}{N}\sum_{i=1}^{N}\Delta_i$, the proportion of data for which the $\mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}$ model yields better results.

We observe improvements in generalization across both FOMMs and SSMs with the use of $\mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}$. We do not see discernible gains from $\textbf{GRU}_{\text{PK-PD}}$, perhaps due to missingness in the data, which also results in the GRUs generalizing worse than SSM models (see Table 7.4).

To further probe the performance of $\mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}$, we conduct a control experiment against the Mixture of Experts parameterization in $\textbf{SSM}_{\text{PK-PD}}$ vs $\textbf{SSM}_{\text{MOE}}$ (see Table 7.4 for likelihood results). We find that the inductive bias in each *domain expert* plays a role in ensuring that $\mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}$ outperforms the vanilla Mixture of Experts architecture.

We are also interested in studying the absolute negative log likelihood and predictive capacity of the models. In Figure 7-5a), we use importance sampling to estimate the marginal negative log likelihood of $\textbf{SSM}_{\text{Linear}}$ and $\textbf{SSM}_{\text{PK-PD}}$ for each covariate across all time points. Namely, we utilize the following estimator,

$$p(\mathbf{X}) \approx \frac{1}{S}\sum_{s=1}^{S}\frac{p(\mathbf{X}|\mathbf{Z}^{(s)})p(\mathbf{Z}^{(s)})}{q(\mathbf{Z}^{(s)}|\mathbf{X})}, \tag{7.11}$$

akin to what is used in (Rezende *et al.* , 2014). $\textbf{SSM}_{\text{PK-PD}}$ has lower negative log likelihood compared to $\textbf{SSM}_{\text{Linear}}$ for several covariates, including neutrophil count, albumin, BUN, serum IgM and serum lambda. This result is corroborated with the generated samples in Figure 7-6, which often show that the PK-PD model qualitatively does better at capturing IgM dynamics compared to the Linear model. In general, although there is a large degree of overlap in the estimates of the likelihood under the two models for some features, it is reassuring to see that $\textbf{SSM}_{\text{PK-PD}}$ does explain vital markers like serum IgM and serum Lambda (which are often used by doctors to measure progression for specific kinds of patients), better than the baseline.

In Figure 7-5b), c), and d), we show the L1 error of $\textbf{SSM}_{\text{PK-PD}}$ and $\textbf{SSM}_{\text{Linear}}$ when predicting future values of each covariate. We do so under three different conditioning strategies: 1) condition on 6 months of patient data, and predict 1 year into the future; 2) condition on 6 months of patient data, and predict 2 years into the future; 3) condition on 2 years of patient data, and predict 1 year into the future. Observing 1) and 2) (Figure 7-5b) and c)), we see that prediction quality expectedly degrades when trying to forecast longer into the future. However, the amount of data conditioned on does not seem to affect the L1 error, as the SSM models do well in predicting 1 year into the future (see Figure 7-5b) and d)).

Ablation studies

Table 7.3: **Ablation experiments on the synthetic and ML-MMRF datasets**: **Top):** We study the effect of adding each domain expert module to $\textbf{SSM}_{\text{PK-PD}}$. We report held-out bounds on negative log likelihood. **Bottom):** In this experiment, we study the effect of varying the tunable parameters of the domain expert modules in the SSM models vs keeping them fixed.

Dataset	**Held-out NELBO**	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
ML-MMRF	linear	69.19	72.89	71.13	65.85	62.53
	linear + log-cell	67.33	73.47	71.48	68.01	61.04
	linear + log-cell + te	60.11	70.50	64.45	64.63	55.52

Dataset	**SSM Linear**	**SSM PK-PD (fixed params)**	**SSM PK-PD (varying params)**
ML-MMRF	71.46	62.25	63.04
Synthetic	52.25 +/- 0.04	47.03 +/- 0.02	44.76 +/- 0.01

We report two ablation experiments in Table 7.3.

In Table 7.3 (top), we assess the effect of adding each domain expert module to $\textbf{SSM}_{\text{PK-PD}}$ on held-out negative log likelihood (upper bound). We see that the LC module gives a modest improvement, while the addition of the TE module gives most of the improvements.

In Table 7.3(bottom), we show the effect of fixing all tunable parameters in the domain expert modules vs allowing them to vary over the state space dimension of the SSM. On the synthetic data, varying the parameters yields a measurable improvement in generalization, while doing so on the multiple myeloma data does not yield the same improvement.

Next, we examine how the different models perform on the ML-MMRF data in terms of held-out log likelihood. In Table 7.4, we report held-out negative log likelihoods (or bounds on them for the SSM models) across each fold of the multiple myeloma data. These results anchor the relative pairwise comparisons depicted in Table 7.1,7.2 to absolute likelihood measures. We see that $\mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}$ yields improvements in generalization across all five folds for FOMMs and SSMs. While we do not see these gain in the RNN, we note that the RNN models overall do worse at modeling the ML-MMRF data than the SSMs. In Table 7.5, we show the full set of pairwise comparisons over all five folds of the multiple myeloma data for reference relative to

Table 7.4: **Generalization on held-out data in ML-MMRF:** Lower is better. For the FOMM and RNN models, we report negative log-likelihood. For the SSM models, we report upper bounds on the negative log-likelihood.

Dataset	**Held-out Neg Log Likelihood**	**FOMM Linear**	**FOMM Nonlinear**	**FOMM PK-PD**	**FOMM MOE**	**RNN**	**RNN PK-PD**
Multiple Myeloma	Mean	92.80	97.53	**90.26**	97.26	**89.89**	99.98
	Fold 1	90.81	93.83	87.70	90.51	88.61	98.33
	Fold 2	98.84	103.00	98.54	106.53	93.59	102.62
	Fold 3	98.68	109.56	95.10	109.35	99.17	111.15
	Fold 4	89.58	96.29	88.06	96.74	86.83	98.74
	Fold 5	86.07	84.98	81.89	83.18	81.25	89.05

Dataset	**Held-out Neg Log Likelihood**	**SSM Linear**	**SSM PK-PD**	**SSM Nonlinear**	**SSM MOE**
Multiple Myeloma	Mean	71.46	**63.04**	70.58	68.80
	Fold 1	70.35	60.11	70.87	69.47
	Fold 2	77.99	70.50	74.75	73.98
	Fold 3	73.77	64.45	73.31	73.24
	Fold 4	70.23	64.63	70.64	65.02
	Fold 5	64.96	55.52	63.34	62.31

the limited subset we showcased in Table 7.2.

Table 7.5: **Pairwise comparison of models trained on ML-MMRF**: Higher is better. Each number is the fraction (with std. dev.) of held out patients for which the model that used $\mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}$ has a lower negative log-likelihood (or bound on it) than a model in the same family that uses a different IEF. We report fractions for each fold in ML-MMRF.

Dataset		**FOMM PK-PD** vs. **FOMM Linear**	**FOMM PK-PD** vs. **FOMM NL**	**FOMM PK-PD** vs. **FOMM MOE**
Multiple Myeloma	Mean	0.792 (0.405)	0.668 (0.457)	0.510 (0.490)
	Fold 1	0.813 (0.390)	0.606 (0.489)	0.404 (0.490)
	Fold 2	0.753 (0.432)	0.732 (0.443)	0.665 (0.472)
	Fold 3	0.773 (0.419)	0.835 (0.371)	0.464 (0.499)
	Fold 4	0.799 (0.401)	0.623 (0.485)	0.573 (0.495)
	Fold 5	0.824 (0.381)	0.544 (0.498)	0.420 (0.494)

Dataset		**SSM PK-PD** vs. **SSM Linear**	**SSM PK-PD** vs. **SSM NL**	**SSM PK-PD** vs. **SSM MOE**
Multiple Myeloma	Mean	0.776 (0.414)	0.750 (0.431)	0.706 (0.454)
	Fold 1	0.793 (0.405)	0.798 (0.402)	0.725 (0.446)
	Fold 2	0.778 (0.415)	0.701 (0.456)	0.670 (0.470)
	Fold 3	0.742 (0.437)	0.749 (0.434)	0.742 (0.438)
	Fold 4	0.764 (0.425)	0.734 (0.442)	0.673 (0.469)
	Fold 5	0.801 (0.390)	0.767 (0.423)	0.720 (0.449)

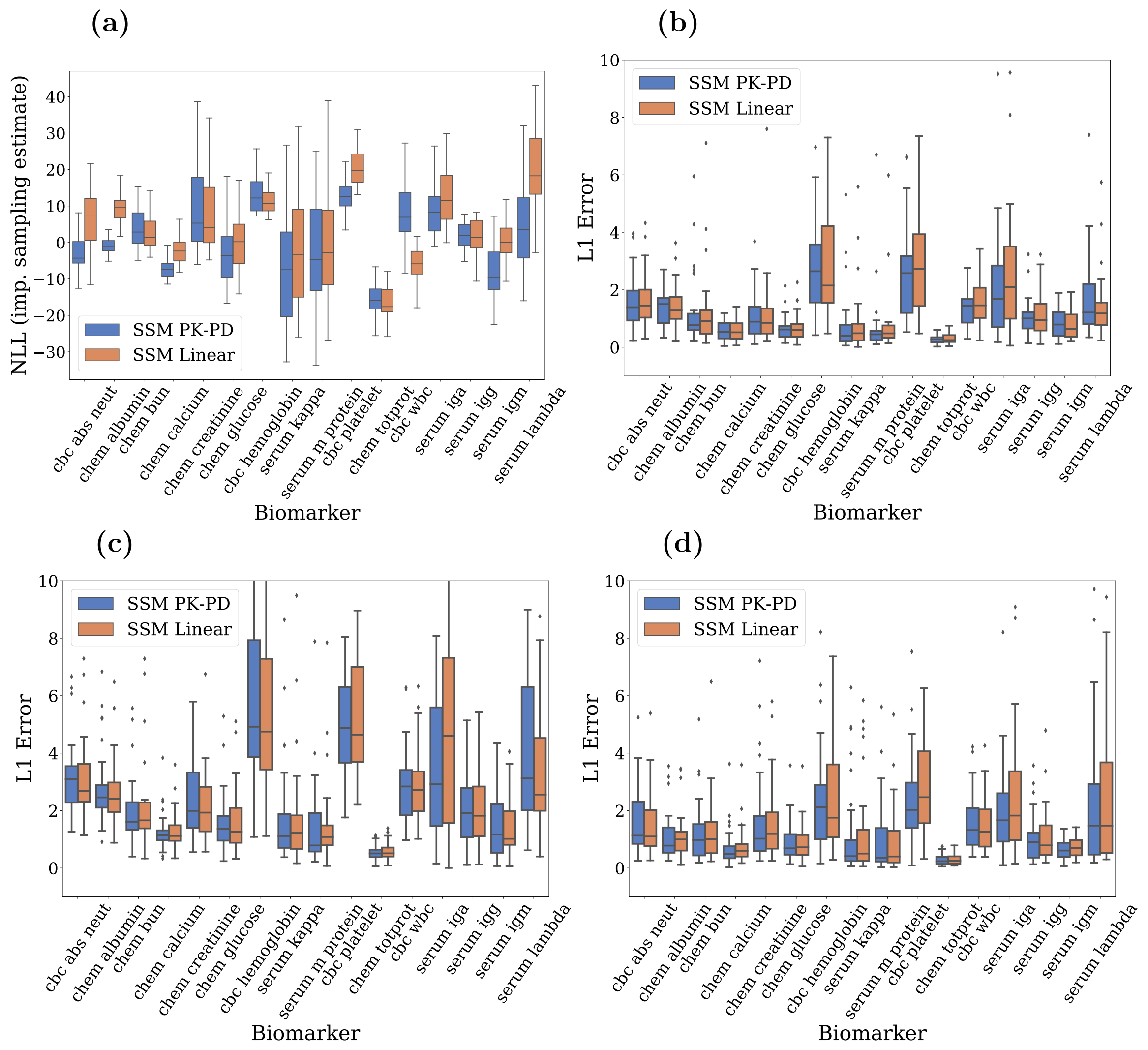


Figure 7-5: **a) NLL estimates via importance sampling**: We estimate the NLL of $\mathbf{SSM}_{\text{PK-PD}}$ and $\mathbf{SSM}_{\text{Linear}}$ for each feature, summed over all time points and averaged over all patients. **b) Condition on 6 months, forward sample 1 year:** We show L1 prediction error for forward samples over a 1 year time window conditioned on 6 months of patient data. At each time point, we compute the L1 error with the observed biomarker and sum these errors (excluding predictions for missing biomarker values) over the prediction window. We employ this procedure for each patient. **c) Condition on 6 months, sample forward 2 years:** We report L1 error for forward samples over a 2 year window conditioned on 6 months of patient data. **d) Condition on 2 years, sample forward 1 year:** Finally, we report L1 error for forward samples over a 1 year time window conditioned on 2 years of patient data.

7.5.2 Qualitative Analysis

Ancestral sampling

Figure 7-4(a) shows samples from three SSMs trained on synthetic data. $\mathbf{SSM}_{\text{PK-PD}}$ captures treatment response accurately while $\mathbf{SSM}_{\text{Linear}}$ does not register that the effect of treatment can persist over time. What role does the local clock (in Section 7.3.1) play? We perform an ablation study on SSMs where the local clock in U_t, used by $\mathbb{PK}\text{-}\mathbb{PD}_{\mathbb{IEF}}$, is set to a constant. Without clocks (PKPD w/o lc), the model does not capture the onset or persistence of treatment response. Figure 7-4(d) shows the average of five ancestral samples from $\mathbf{SSM}_{\text{Linear}}$ and $\mathbf{SSM}_{\text{PK-PD}}$ trained on ML-MMRF. We track five biomarkers that characterize myeloma. $\mathbf{SSM}_{\text{PK-PD}}$ better captures the evolution of biomarkers conditioned on treatment, particularly of serum IgA, where $\mathbf{SSM}_{\text{Linear}}$ mistakenly predicts the value will be steady. For serum lambda and IgG, the PK-PD model predicts the dip and rise in the lab values, while the linear model does not.

In the samples described above, we conditioned on the patient's baseline covariates and longitudinal treatments. We now visualize sampling from the model *after* inferring the latent representations of patients up to a point in time that we condition on. Let C denote the point in time until which we condition on patient data and F denote the number of timesteps that we sample forward into the future. We limit our analysis to the subset of patients for which $C + F <= T$ where T is the maximum number of time steps for which we observe patient data. The following samples we display are obtained as a consequence of averaging over five different samples, each of which is generated (for the SSM) as follows:

$$\begin{aligned}
&Z \sim q_\phi(Z_C|Z_{C-1}, X_{1:C}, U_{0:C-1})\\
&Z_k \sim p_\theta(Z_k|Z_{k-1}, U_{k-1}, B) \quad k = C+1, \ldots, C+F\\
&X_k \sim p_\theta(X_k|Z_k) \quad k = C+1, \ldots, C+F
\end{aligned} \tag{7.12}$$

We study the following strategies for simulating patient data from the models.

1. Condition on 6 months of patient data, and then sample forward 2 years,

2. Condition on 1 year of a patient data and then sample forward 1 year,

3. Condition on the data coinciding with a patient's first line therapy and then

forward sample until the end of their third line therapy.

In Figure 7-6, we show additional samples from $\mathbf{SSM}_{\text{PK-PD}}$ when conditioning on differing amounts of data. Overall, in all three cases, $\mathbf{SSM}_{\text{PK-PD}}$ models treatment response better than a linear baseline. For 1. (Figure 7-6a)), we see that $\mathbf{SSM}_{\text{PK-PD}}$ correctly captures that the serum IgA value remains steady while $\mathbf{SSM}_{\text{Linear}}$ predicts an upward trend. For 2. (Figure 7-6b)), $\mathbf{SSM}_{\text{PK-PD}}$ does well in modeling down-trends, as in serum IgA and serum IgM. For 3. (Figure 7-6c)), we similarly see that $\mathbf{SSM}_{\text{PK-PD}}$ captures the down-trending serum IgA and serum IgM during the second line therapy.

Interpreting what the model has learned about multiple myeloma

Do the models learn known clinical relationships between interventions and observations? On $\mathbf{SSM}_{\text{PK-PD}}$, we analyze this via the sensitivity function $\nabla_{U_{t-1}}\mathbb{E}_{Z_t}(X_t|Z_t)$. This presents another use case wherein gradients of a deep generative model may be used for In Figure 7-7a) and b) show how changes in two combination therapies, Lenalidomide, Bortezomib, Dexamethasone (RVD) and Bortezomib, Dexamethasone (VD) respectively are associated with changes in clinical lab markers. (a) RVD is associated with a decrease in hemoglobin and platelet values, two known side effects of the treatment (Kumar *et al.* , 2012). (b) We observe a diminished effect of VD on hemoglobin, platelets and creatinine. Indeed, VD is given in favor of RVD when trying to avoid side effects (Jacobus *et al.* , 2016).

Figure 7-4(c) visualizes the gates $\sigma(\boldsymbol{\delta})$ from $\mathbf{SSM}_{\text{PK-PD}}$ trained on ML-MMRF. The highest weighted component is the treatment exponential model followed by the log-cell kill model for many of the latent state dimensions. This result tells us (a) that no single domain expert is responsible for the dynamics of all the latent dimensions and (b) the treatment exponential IEF appears to have the largest weights across several dimensions.

Knowing that α_{1t} in the treatment exponential IEF drives much of the variation in representation, we perform TSNE (Maaten & Hinton, 2008) on each held-out patient's high-dimensional α_{1t} at two time points in Figure 7-4(b). This analysis shows how variation in α_{1t}, and consequently the dimensions of the latent representation, are driven by treatment. Early on, the representations segregate by lenalidomine (a first-line therapy), whereas for patients who make it through three years of treatment, the line of therapy drives the representation. The $\mathbb{PK}$-$\mathbb{PD}_{\mathbb{IEF}}$ thus admits a hierarchical decomposition of information that aids in interpreting the model's dynamics.

In Figure 7-8, we perform TSNE (Maaten & Hinton, 2008) on each held-out patient's high-dimensional α_{1t} vector (obtained from $\mathbf{SSM}_{\text{PK-PD}}$ trained on ML-MMRF) at *multiple* time points, expanding on the two time points that were shown in Figure 7-4. Overall, we gain a richer understanding of how the variation in α_{1t} is driven by treatment. As we saw before, the representations segregate by presence of lenalidomide in initial therapy. Later, they segregate by line of therapy; finally, most patients are taken off treatment.

In Figure 7-7c), we show the how treatments in U_t play a role in α_{1t} by visualizing the weights of the linear model that maps from treatment signal to α_1. This result showcases how this parameter varies as a function of treatment.

An important point to note in these plots is that as T increases, the number of patients decreases due to right censoring in the ML-MMRF dataset.

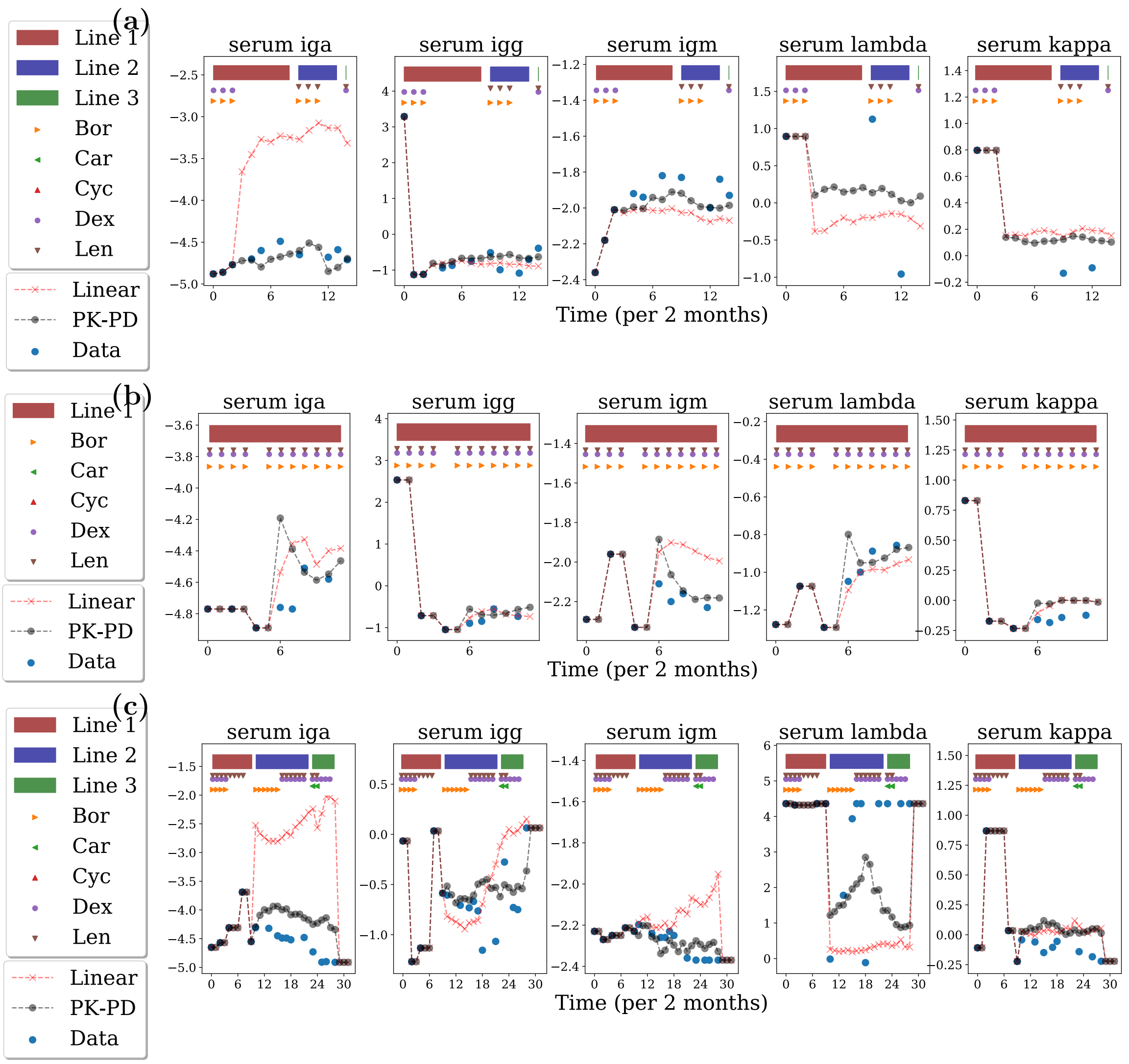


Figure 7-6: **Samples from learned SSM models with differing conditioning strategies**: We visualize samples from $\mathbf{SSM}_{\text{PK-PD}}$ (o) and $\mathbf{SSM}_{\text{linear}}$ (x). Each row corresponds to a single patient, whereas each column represents a different biomarker for that patient. **a)**: We condition on 6 months of patient data and forward sample 2 years. **b)**: We condition on 1 year of patient data and forward sample 1 year. **c)**: We condition on data corresponding to the patient's first line of therapy and then forward sample the extent of their second and third line therapies. The blue circles denote ground truth, and the markers above the trajectories represent treatments prescribed across time.

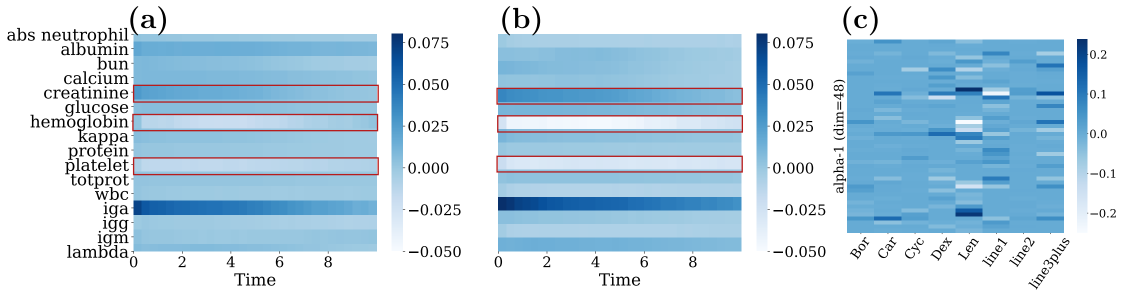


Figure 7-7: **a),b) Heatmaps showing directional derivative of expected longitudinal values:** Here, we depict two heatmaps showing the directional derivatives of the expected longitudinal data with respect to VD (**a)**) and RVD (**b)**), two common first line therapies in multiple myeloma. Red boxes surround hemoglobin, creatinine, and platelet count, covariates that display the most differences between the two therapies over time. This analysis was done on $\mathbf{SSM}_{\text{PK-PD}}$. **c) Weights on the linear model that maps treatment to α_1:** We visualize the weight matrix of the linear function that maps the treatment signal to α_1, which varies across the state space dimension, in $\mathbf{SSM}_{\text{PK-PD}}$.

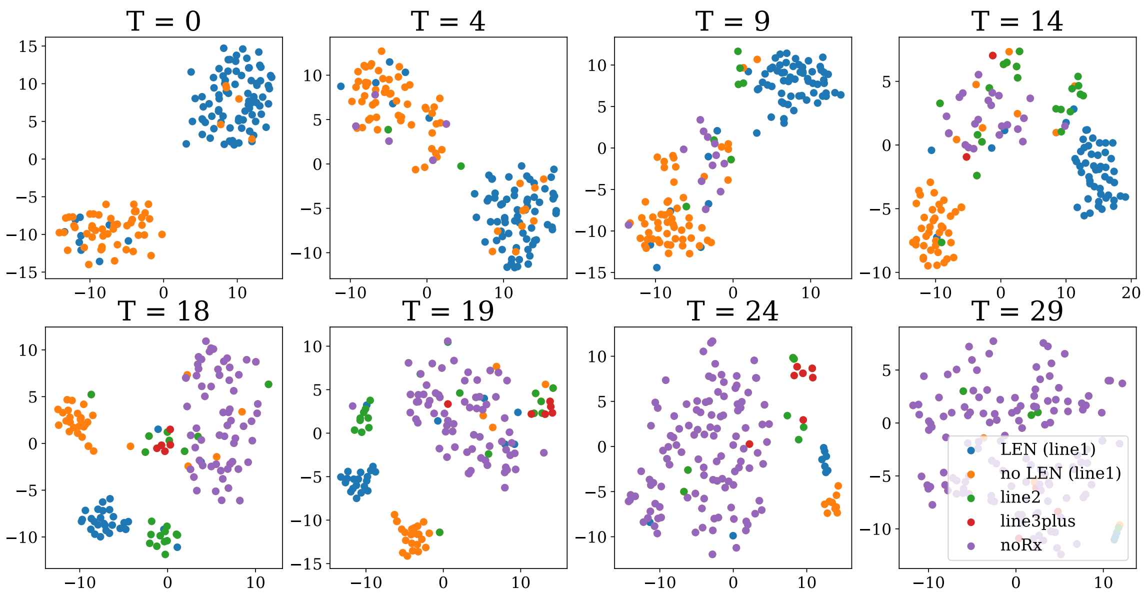


Figure 7-8: α_{1t} **Visualizations:** We visualize the TSNE representations of each held-out patient's α_1 parameter (in TE module used in trained $\mathbf{SSM}_{\text{PK-PD}}$) over multiple time points.

7.6 Discussion

$\mathbb{PK}$-$\mathbb{PD}_{\mathbb{IEF}}$ uses domain knowledge from pharmacology, makes use of the properties of interventions found in clinical data of chronically ill patients, and in doing so, improves generalization of representation-learning based models in the low data regime. $\mathbb{PK}$-$\mathbb{PD}_{\mathbb{IEF}}$ admits introspection and captures known clinical information about treatment effects. As Bica *et al.* (2020) note, there is very little work in using pharmacology to design priors for machine learning models.

Approaches such as (Albers *et al.* , 2012) make use of domain knowledge in the form of physiological models to track serum glucose dynamics from electronic health record data. In a similar vein, albeit in the context of representation learning, we instantiate knowledge from pharmacology in deep generative models and study the statistical ramifications of doing so (improved generalization in the low data regime).

An interesting question is whether deep generative models, such as the ones we design herein, may be used to tackle problems in fields such as quantitative systems pharmacology (Helmlinger *et al.* , 2019; Jusko, 2013; Fleisher *et al.* , 2017). The $\mathbf{SSM}_{\text{PK-PD}}$ may be seen as an adaptation of Bayesian pharmacokinetic models (Lenert *et al.* , 1992) to representation learning. If data were no barrier, *and* interpretability of the models parameters is not paramount, then the $\mathbf{SSM}_{\text{PK-PD}}$ can present practitioners with new approach for building new simulation based pipelines to determine effective drug doses using deep generative models (Hutchinson *et al.* , 2019).

It is worth pausing to reflect upon why the use of functions derived from PK-PD models make sense within representation learning frameworks. One hypothesis would be that through trial and error, these are functions that which commonly capture the range of variation in time due to the prescription of treatment. Another hypothesis could be that the representations learn the marker for disease burden, such as tumor volume, that the PK-PD model was designed to model. The latter hypothesis, if true, would be remarkable since tumor volume could be broadly useful as a prognostic marker of disease burden but is often (a) difficult to measure and (b) rarely observed in electronic health record data and therefore difficult to regress onto. To test such a hypothesis however, we would need access to a large dataset of patient records and linked clinical trial data where tumor burden is measured; such data may prove challenging to come by.

The work herein may be seen as a hybrid between the data intensive approach typically adopted by deep learning where the goal is to learn correlation between random variable from scratch, and the approach made use of in knowledge-based systems (Szolovits, 1986, 1982; Szolovits *et al.* , 1988; Patil & Schwartz, 1982) where relationships between random variables are prescribed. The issue with the former is that in data poor regimes, learning functional relationships is hard; the issue with the latter is that it can be brittle and sensitive to potential model misspecification. By grounding the functional forms of the functions in a knowledge based system, whilst still making use of data to guide the learning of relationships, we seek to obtain the benefits of both paradigms.

Chapter 8

Latent Representations of Privileged Information

Chapter 6 introduced Deep Markov Models and Chapter 7 introduced intervention effect functions, which when used in Deep Markov Models, improved generalization of deep generative models trained on clinical data. An important question to ask at this juncture is what can good unsupervised models of sequential, clinical data be used for?

A motivating example we consider is the problem of building risk stratification tools for rare diseases. In such problems, we are data limited due to the frequency by which the disease manifests in the general population. In this chapter, we show that if one has access to a model that can uncover patterns in time-varying data, then, by making use of the inferred patterns, we can reduce the sample complexity of supervised learning. Our approach opens a new set of algorithmic techniques to learn risk prediction models from high-dimensional data when labelled data is scarce.

8.1 Introduction

Accurate models for predicting patient risk can have a large impact on clinical care and practice. For diseases with no known cure, risk prediction models can guide ongoing care and help clinicians and patients plan future therapy development (Razavian *et al.* , 2015; Chen & Asch, 2017). In the selection of patient cohorts for clinical trials, risk stratification tools can ensure a new drug's effect is validated on a diverse group of people through cohort selection (Shivade *et al.* , 2013). A common approach to patient risk stratification is to collect labelled patient data prior to treatment (baseline data) and to regress onto an outcome of interest such as death or progression-event for a disease. Approaches based on deep neural networks along these lines can work well, and success stories abound (Gulshan *et al.* , 2016; Razavian *et al.* , 2015). But such models are data-hungry. What happens when data is scarce?

In this work, we make use of *privileged information* (Vapnik & Vashist, 2009): information (features) available during training but not available for prediction at test time. In clinical settings, privileged data is often available but is rarely used for task of risk stratification. For example, such information can take the form of prescribed medications and patient treatment response as measured in terms of longitudinal biomarkers relevant to the disease. This work aims to achieve two goals: (a) to build models that capture the progression of a disease as observed in privileged data, and (b) to learn representations from such models that contain information vital to building accurate risk stratification models when data is scarce.

It is known that patients respond differently to the same medication. This heterogeneity in response to medication can be driven by a patient's underlying genetics as well as their past medical history. The key assumption that we will make here is that characterizing this heterogeneity can yield insight into disease progression and aid prediction of patient outcomes. To operationalize these insights, we propose the Privileged Information Variational Autoencoder (PIVAE), a deep generative model which uses a latent variable to model the statistical variation in treatment effect that remains constant across time. The model predicts patient outcomes of interest using the latent variable while conditioning on baseline patient covariates.

Why should we expect gains from using privileged information? In the low-data regime, there can be a high degree of uncertainty in the decision boundary for a prediction task from baseline data alone. Intuitively, a representation of post-treatment observations may decrease uncertainty by providing a different view of the prediction problem and consequently improve accuracy. Building on learning using privileged information (LuPI), our work serves as a case-study for how clinical domain knowledge can be utilized to make deep learning in healthcare practical in the low data regime.

In this chapter we introduce the Privileged Information VAE (PIVAE), a conditional deep generative model designed to capture statistical patterns in the effect of treatment (or control signals) on a time-varying, multi-variate longitudinal sequences. We use the PIVAE to form a representation of privileged data, which can be used to improve predictive performance for outcomes of interest in the low-data regime.

8.2 Setup

We begin by presenting background on learning risk prediction and learning using privileged information.

Survival analysis. Survival analysis (Cox, 2018) is a popular tool used used to estimate the time to a patient outcome of interest Y conditional on some covariates X when the outcomes are censored (or unobserved). Censoring occurs if an patient outcome is unknown due to incomplete information or if the patient leaves before the end of the study. The survival analysis literature spans the gamut of non-parametric models such as the Kaplan-Meier model (Kaplan & Meier, 1958), semi-parametric models such as the the Cox-proportional hazards model (Cox, 1972) and fully parameteric models such as the Weibull distribution (Klein & Moeschberger, 2006). To learn parameteric survival models via maximum likelihood we denote a binary indication of censorship as C. In this setting, it is common to combine a log survival function sf for censored events with a model log likelihood of observed events using the binary censorship variable C (Klein & Moeschberger, 2006) and maximize:

$$\log p(Y|X, C) = (1 - C) \log p(Y|X) + C \log \mathrm{sf}(Y|X) \tag{8.1}$$

Learning using privileged information (LuPI). In most machine learning problems, the training data used for model development, the validation data used for hyperparameter tuning, and test data for evaluation all comprise the same set of covariates and labels. With LUPI, at training time, we have access to *privileged* information not available at validation or test time. Prior work in support vector machine classification has shown that privileged information such as slack variables or related correcting functions can improve learning(Vapnik & Vashist, 2009). Similar results have been seen for problems in multi-class learning (Wang *et al.* , 2018).

8.3 Privileged Information Variational Autoencoder

We seek a risk stratification tool for a single disease of interest. Such tools are often built by using pre-treatment data (i.e. data about a patient prior to undergoing therapy) to regress onto an outcome that characterizes patient risk (such as the time to death). For rare diseases, this learning problem will typically lie in the low-data regime which limits the use of more data-hungry non-linear models.

Fortunately, at training time, in addition to pre-treatment data, we often have *post-treatment* data available. We will assume the latter is in the form of longitudinal trajectories of treatments and their subsequent effects on bio-markers that track disease progression. Such patient trajectories are often available when building risk prediction models from Electronic Medical Record (EMR), registry or claims data. This is *privileged information* that we will leverage to build our risk prediction models.

The central hypothesis of this work is twofold: first, that we can learn a representation of privileged information which characterizes the progression of disease; and second, that the representation provides a *different view* of the pre-treatment data that is easier to correlate with patient outcomes.

But what principle guides the representation we seek? It is well known that there is heterogeneity in the way a disease manifests itself and that a disease that we refer to by a single name, could comprise many different sub-types. For example, Ahlqvist *et al.* (2018) describes five different subgroups within diabetes and demonstrates that patient outcomes vary by subgroup. We use privileged information to uncover this latent subgroup. We seek a representation such that closeness in representation space corresponds to similarity in progression patterns, and, we hypothesize, patient outcomes of interest.

(Privileged) longitudinal data, U, X: For patients in the *training* set, we assume that we have access to a sequence of multivariate interventions $U = (U_1, \ldots, U_{T-1})$ and multivariate, post-treatment biomarkers $X = (X_1, \ldots, X_T)$ for $T > 0$. The biomarkers may be any combination of real-valued (e.g. laboratory measurements of proteins), categorical (e.g. responses to survey data) or binary-valued (indications of comorbidities). The PIVAE will seek to build representations, Z, of these (privileged) time-varying biomarkers.

Baseline data, B, corresponds to patient features prior to the start of therapy, used in the prediction task—these are the only data available at test time. The set B

will include co-morbidities, age, gender, demographics, genetic features. We assume it is a statistic that when conditioned on, renders any future outcome independent of any previous medical history in the patient's record. We will also assume that baseline biomarkers X_0 and first therapy prescribed U_0 form part of the pre-treatment covariates available for risk-stratification.

Patient outcomes, Y: We pose risk stratification as the problem of estimating the probability, using a suitable model, of a clinical outcome of interest, given their baseline features. The outcome may be binary (will the patient survive for a year), ordinal (will the patient survive for one, two or three years) or real-valued (time-to-adverse event). Y can either be censored $C = 1$ or observed $C = 0$.

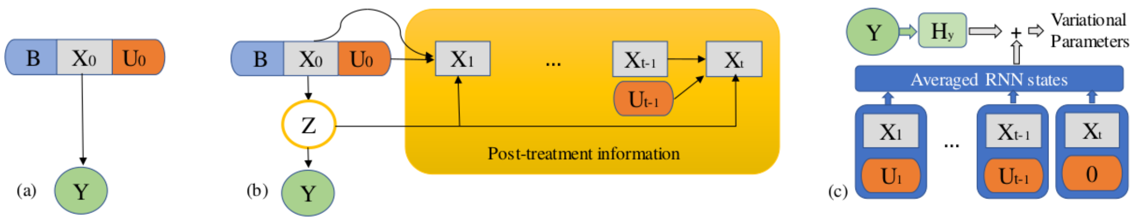


Figure 8-1: **Learning with post-treatment information:** (a) prediction of outcomes from baseline data only. (b) the Privileged Information Variational Autoencoder (PIVAE) (c) the PIVAE's inference network.

We typically regress onto Y using B at training time to learn $p(Y \mid B, X_0, U_0)$ and use the resulting function for test time prediction (Figure 8-1 (a)).

In Figure 8-1 (b), we visualize the Privileged Information Variational Autoencoder, whose generative process we now describe:

$$Z \sim p(Z|B, X_0, U_0; \theta_1);\ Y \sim p(Y|Z; \theta_2);$$
$$X_t \sim p(X_t; X_{t-1}, U_{t-1}, Z; \theta_3)\ t = \{1, \ldots, T\} \tag{8.2}$$

Z is a latent variable that serves as a summary statistic for all post-treatment information whose prior depends on the pre-treatment set of variables (B, X_0, U_0). The outcome Y is a function of Z (and optionally the pre-treatment variables).

Learning and Prediction: We maximize the likelihood of $Y, X_1, \ldots, X_T$ given $B, X_0, U_0, \ldots, U_T$. For simplicity, we denote the set $X_1, \ldots, X_T$ as $X_{1:T}$ and $U_1, \ldots, U_T$

as $U_{1:T}$. For a single patient:

$$\begin{aligned}
&\log p(X_{1:T}, Y|B, X_0, U_{0:T}, C) \\
&= \log \int_Z p(X_{1:T}, Y, Z|B, X_0, U_{0:T}) \\
&= \log \int_Z p(X_{1:T}|Z, X_0, U_{0:T}) p(Z|B, X_0, U_0) p(Y|Z, C) \\
&= \log \int_Z q(Z|Y, X_{1:T}, U_{1:T}; \phi) \\
&\quad \prod_{t=1}^{T} p(X_t|X_{t-1}, Z, U_{t-1}) \frac{p(Z|B, X_0, U_0)}{q(Z|Y, X_{1:T}, U_{1:T}; \phi)} p(Y|Z, C) \\
&\geq \mathbb{E}_{q(Z|Y, X_{1:T}, U_{1:T}; \phi)} \left[\sum_{t=1}^{T} \log p(X_t|X_{t-1}, Z, U_{t-1}) \right. \\
&\left. + \log p(Y|Z, C) \right] \\
&- \mathrm{KL}(q(Z|Y, X_{1:T}, U_{1:T}; \phi) || p(Z|B, X_0, U_0)) \qquad (8.3)
\end{aligned}$$

where $\log p(Y|Z, C)$ is estimated as in Equation 8.1.

The inference network for the model is depicted in Figure 8-1 (c). A recurrent neural network uses the concatenation of treatments and interventions in addition to a representation of the outcome to parameterize the variational distribution $q(Z|Y, X_{1:T}, U_{1:T}; \phi)$.

At test time, $X_1, \ldots, X_T, U_1, \ldots, U_T$ are unobserved and we approximate the prediction function $P(Y|B, U_0, X_0) = \int_Z p(Y|Z) p(Z|B, U_0, X_0)$ for prediction.

Overview: It is worth giving pause to why the PIVAE is a good fit for the problem at hand.

Our first goal was to learn a representation of privileged information. The PIVAE does this by modeling the likelihood of privileged data as a function of the latent variable Z. At training time, the privileged data are observed and probabilistic inference (via an inference network) is used to infer Z by maximizing a lower bound on the log likelihood of the observed data. At prediction time, the privileged information is unobserved. However, as leaves in a Bayesian network, they may be ignored and a Monte-Carlo approximation to the prior-predictive (in this case, a marginalization of latent variable Z) is used to predict the outcome Y.

Our second goal was to build a model wherein the representation learned yielded insights into the progression of a disease. Patients respond differently to treatments over time. The PIVAE hypothesizes that there is structure in the response to therapy as observed in the longitudinal biomarkers and that we can uncover disease subtypes by using a deep generative model to characterize variation in the response to therapy.

Parameterizations: We make the following choices for the conditional densities in Equation 8.2.

Prior conditional $p(Z; B, X_0, U_0, \theta_1)$*:* $Z \sim \mathcal{N}(\text{softmax}(W_h[B; X_0; U_0] + b_h) * W_{\mu_p}; \Sigma_p)$ where $[A; B]$ denotes the concatenation of A and B. The prior mean is computed as a weighted sum of learned *protoype* means in W_{μ_p}.

Outcome conditional $p(Y|Z; \theta_2)$*:* If the prediction task is a regression, we parameterize the outcome as a linear function of Z: $Y \sim \mathcal{N}(W_y Z + b; \Sigma_y)$. When the event is time-to-death (positive, real-valued number), we use a log normal distribution: i.e. $\log Y \sim \mathcal{N}(W_y Z + b; \Sigma_y)$.

Longitudinal data $p(X_t; f(X_{t-1}, U_{t-1}, Z; \theta_3))$*:* For the data considered here, all the longitudinal biomarkers are real-valued. We therefore model the biomarkers at each point in time as:

$$X_t \sim \mathcal{N}(X_{t-1} + \text{MLP}(Z; \theta_3)^T U_{t-1}; \Sigma_x) \tag{8.4}$$

8.4 Evaluation

We first study the PIVAE in the context of a synthetic setting designed to mimic our application of interest.

8.4.1 Synthetic Data

Each patient has a six dimensional baseline state (B), the first two components of which are visualized in Figure 8-2 (left). At training time, we have access to two biomarkers across time (X) (the privileged information). The task is to predict each patient's time-to-death (Y). We assume that there are four distinct subtypes in the disease (denoted as a categorical random variable S, here) and that each patient belongs to a single subtype. A patient's subtype will depend on which orthant of the two-dimensional plane the *first two components* of the patient's six-dimensional baseline data lie in (seen in the different markers and colours in Figure 8-2 (left)).

The subtype determines the pattern followed by the biomarkers (denoted X_1, X_2). Time-to-death is a noisy function of each patient's subtype as described in Equation 8.5.

The treatment vector at time t, U_t, for each patient includes the time elapsed from the start of therapy, denoted by t_s, a one hot encoding of the line of therapy, and a binary variable that is 1 if a treatment is given at the current time point or has been given previously and 0 otherwise.

This is identical to the synthetic dataset used in Chapter 7. However, here the goal is to do good supervised learning whilst treating the longitudinal data as privileged information.

This dataset has the following properties that merit its use our study. First, subtype is inferrable from both baseline data and (privileged) longitudinal information. Second, time-to-death (outcome) is a nonlinear function of baseline data but is a simple linear function of the subtype.

$$\begin{aligned} &B_{1\ldots,6} \sim \mathcal{N}(0; I), \\ &Y = \\ &\begin{cases} 3 + \mathcal{N}(0, 0.5), B_1 \geq 0, B_2 \geq 0 \\ 7 + \mathcal{N}(0, 0.5); B_1 \geq 0, B_2 < 0 \\ 9 + \mathcal{N}(0, 0.5); B_1 < 0, B_2 \geq 0 \\ 5 + \mathcal{N}(0, 0.5); B_1 < 0, B_2 < 0 \end{cases} \end{aligned} \tag{8.5}$$

8.4.2 Evaluation

We compare the following models for this predictive task: *Linear* denotes a linear parameterization of $p(Y|B, X_0, U_0)$, *Random Forest* denotes a random forest regression for $p(Y|B, X_0, U_0)$. Since we know that the true regression function given baseline data can be effectively represented by half-spaces, this is a very strong baseline for this task. *Chained* assumes oracle access to the ground-truth subtype S for each patient and parameterizes $p(Y|B, X_0, U_0) = g(Y|f(S|B, X_0, U_0))$ where g is a linear function from subtype onto outcome, and f is a random-forest that regresses onto subtype

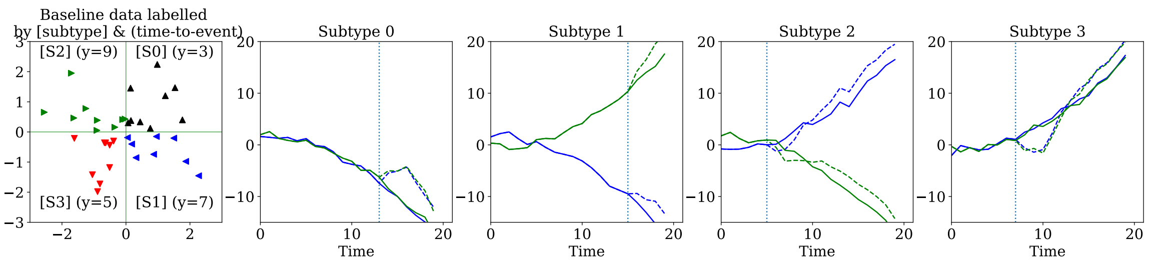


Figure 8-2: **Visualizing synthetic data:** Left: A visualization of patient's baseline data (coloured and marked by patient subtype). Each quadrant is annotated with [subtype] (time-to-death). Right four plots: For patients from each of the subtypes, an example of their longitudinal trajectories. The solid lines are trajectories had there been no treatment, while the dotted lines over time represent trajectories with treatment response. The dashed vertical line represents the therapy given at a particular point in time.

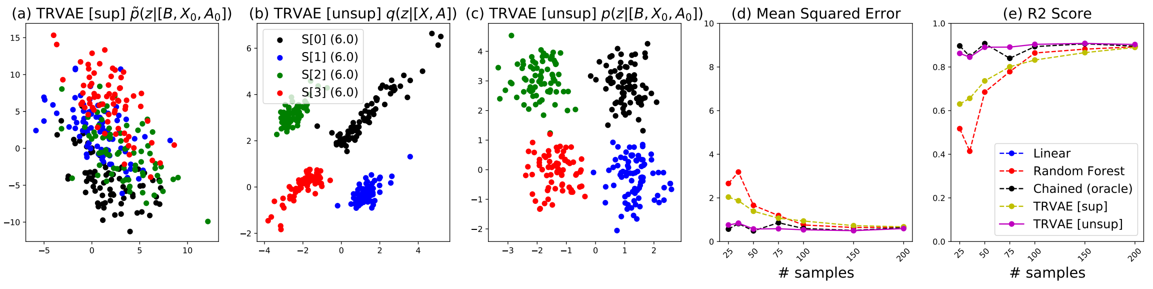


Figure 8-3: **Synthetic (held-out) data:** (a) depicts the delta distribution implied by Z under a supervised PIVAE while (b) $\sum_n q(Z_n|X, U)$, (c) $\sum_n p(Z_n|B, X_0, U_0)$ visualize the corresponding distributions from an unsupervised PIVAE. (d), (e) visualize different accuracy metrics comparing the PIVAE to various baselines.

from baseline data. This is an approximation to the best achievable regression model. *PIVAE [supervised]* (denoted with [sup]) corresponds to learning the PIVAE in a fully supervised manner. i.e. via minimizing a Maximum A-Posteriori approximation to $\mathbb{E}_{p(Z|B,X_0,U_0)}[-\log p(Y|Z, C)]$. *PIVAE [unsupervised]* (denoted with [unsup]) learns by maximizing Equation 8.2.

How do the different methods compare to one another? We evaluate performance on mean square error (MSE) and R^2 (coefficient of determination) as a function of the number of training samples while keeping fixed the number of held-out data at 300 points. We conduct a hyperparameter search over the number of latent dimension $(4, 10)$, number of hidden dimensions in $p(Y|Z, C)$ $(10, 50)$ and learning rates $(1e-3, 8e-3)$ identically for PIVAE [sup] and PIVAE [unsup] and selected the best model using mean square error on held-out data as the metric.

In Figure 8-3 (d,e) we see that the linear models, as expected, perform poorly (since the outcome is not linear in the input). (Note that the MSE for the linear models is >30, so therefore are not shown in the plot). Both random forests and PIVAE [sup] do well when the number of training samples exceeds one-hundred but their performance degrades in the low-data regime. In contrast, PIVAE [unsup], matches the performance (in MSE and R^2) of an oracle regression model with access to the true underlying subtype.

What advantage does using of privileged information confer upon PIVAE [unsup] relative to PIVAE [sup] in the small data regime? To answer this question we train, on 25 patients, a supervised and unsupervised version of the PIVAE where Z is two-dimensional. When PIVAE [sup] is via a MAP approximation to

$$\mathbb{E}_{p(Z|B,X_0,U_0)}[-\log p(Y|Z,C)]$$

, we obtain a delta-distribution $\tilde{p}(Z|B,X_0,U_0))$. In Figure 8-3 (a) we visualize this distribution evaluated on a held-out set. Although the PIVAE [sup] has begun to separate out patients with varying outcome times, learning about subtype structure, in the low-data regime, is difficult from baseline data alone. The same representation $p(Z|B,X_0,U_0)$ for PIVAE [unsup] is plotted in Figure 8-3 (c) where we see that the model has successfully learned to map from baseline data onto four distinct regions corresponding to subtype – why has this happened? Note that the learning signal for $p(Z|B,X_0,U_0)$ is derived from the minimization of $\text{KL}(q(Z|Y,X_{1:T},U_{1:T};\phi)||p(Z|B,X_0,U_0))$ (Equation 8.2). We plot the aggregate posterior distribution in Figure 8-3 (b) which reveals that the inference network has used privileged information to uncover the latent subtype. The minimization of KL divergence consequently transfers this knowledge onto $p(Z|B,X_0,U_0)$, allowing the model to generalize effectively at test time. Note that the functions used in PIVAE [sup] and PIVAE [unsup] are identical in their structure and number of parameters, what differs is that at training time, the privileged information is used to construct a view on data which when leveraged by the latter allows it to generalize better. This, in effect, validates the utility of privileged information in the small data regime.

To inspect what the PIVAE [unsup] (trained on 25 patients) learns, in Figure 8-4 (a) we visualize values of h_{global} in matrix form. Across all patients within a subtype, we average the estimates of h_{global}, binarize the result and insert it into a row of the matrix visualized in the plot. We find that the global structure learned by the model corresponds to the four patterns of variation exhibited by the biomarkers for each subtype. We forward sample the longitudinal data from the model and visualize the

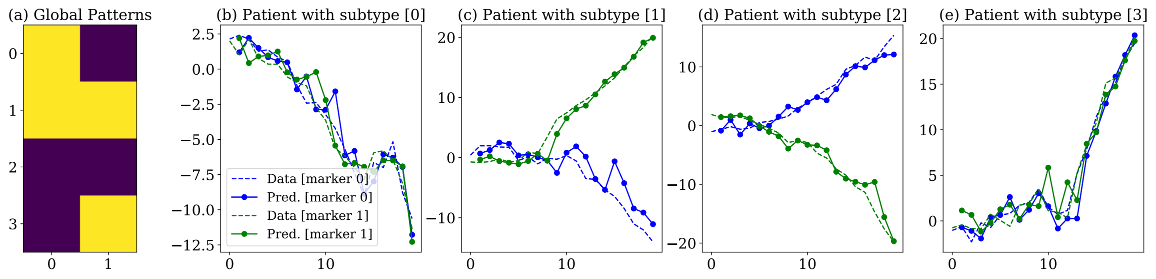


Figure 8-4: **Visualizing the learned model** (a) Visualization of h_{global} (each row corresponds to the averaged, binarized h_{global} of a patient within a subtype); (b, c, d, e) for a single patient from each subtype, we sample the patient's biomarker from the generative model (conditioned on their baseline data), where we see a good fit to the ground truth

results in Figure 8-4 (b, c, d, e) where we validate that it forms a reasonable fit to the ground truth for both biomarkers.

8.5 Related work

Our work is inspired by learning with privileged information (Vapnik & Vashist, 2009), who impose constraints on slack variables (available as privileged information) to improve the generalization of support vector machines (SVMs). The underlying principle espoused is the judicious use of additional information at training time to improve test-time performance. However, rather than using privileged information to modify the parameters of an existing classifier, we use generative models to capture the content of privileged information into a representation that when conditioned on, improves test-time generalization (Sohn *et al.* , 2015). Lopez-Paz *et al.* (2015) unify model distillation and learning with privileged information with a framework called generalized distillation. In the framework, the teacher typically predicts the outcome using privileged information and the student uses the teacher's prediction estimates to improve learning. In our work, we rely on a generative model's latent representation to have the capacity to be a good teacher.

Unlike semi-supervised learning (SSL) with deep generative models (Kingma *et al.* , 2014) , we have *more* information for *all* training points compared to SSL for which we have unlabelled data at training time. In a similar vein, multi-task learning (MTL) Caruana (1997) seeks to find a common representation to capture the similarity between multiple prediction tasks (in this case, the prediction of longitudinal trajectories and survival outcomes). However, MTL typically seeks to solve multiple prediction tasks at test time.

There is a rich history of jointly modeling longitudinal data and clinical outcomes (Wu *et al.* , 2012). Unsupervised sequential models have been used to model diseases like Chronic Kidney disease (Futoma *et al.* , 2016; Wang *et al.* , 2014b) and modeling mobile health drug-use (Dempsey *et al.* , 2017). In addition to modeling sequential data, Ranganath *et al.* (2015) use a time-varying latent variable to parameterize the hazard function while predicting outcomes for heart patients. Schulam & Saria (2016) propose a conditional Bayesian network that models the progression of a single biomarker as a function of other observed data. They use latent variables as proxies for the observed set of biomarkers. Our work differs from theirs in our explicit desire to use heterogeneity in treatment effect as the means by which we uncover patterns in data.

Gabler *et al.* (2009) discuss the importance of explicitly accounting for treatment heterogeneity, particularly in the context of designing and evaluating clinical trials. Smolenski *et al.* (2017) study the heterogeneity in the context of patients treated for depression via video-conference. In child psychology, Mertens *et al.* (2017) study heterogeneity of response to therapy for problematic behavior. To the best of our knowledge, we are not aware of other work that uses deep generative models to capture treatment heterogeneity.

8.6 Discussion

This chapter proposes the Privileged Information Variational Autoencoder. The model captures representations of subtype in longitudinal patient data while correcting for the effect of treatments. We show how the model's latent representations can serve both as a diagnostic tool to understand how disease behave as well as improve the predictive performance of risk prediction tools.

Although our method is most useful at providing predictive gains in the low-data regime, there are limitations to the model and care must be taken in its use for building risk-prediciton models. First, within the prediction problem must exist a degree of correlation between the variation in privileged information and the outcome of interest. In such scenarios, the information captured in the latent variable be correlated to the outcome and provide a kind of supervision to the risk-prediction model than the outcome alone. Second, designing good deep generative models of privileged information often requires domain knowledge for the problem at hand. Validating the approach by building a new risk prediction model on a real world dataset is an important direction for future work.

Chapter 9

Conclusion

We are constantly discovering new ways to measure phenomena occurring at a variety of scales in the human body. As we do so, our notions of healthy and diseased changes. While our understanding of the mechanisms that drive change over time in the human body is constantly improving, there is much we do not know. In the absence of detailed mechanistic knowledge about the data generating processes that drive, we conjecture in this thesis that deep generative models, given sufficient data, may prove a capable surrogate as a computational modeling tool for clinical questions of interest.

This thesis develops supervised, and unsupervised learning algorithms for models of high-dimensional data designed to tackle some of the challenges that arise in the context of healthcare. We return to the challenges highlighted in Chapter 1 and outline how some of the work in this thesis addresses them.

Heterogeneity, sparsity, missingness, and high-dimensionality: This thesis makes use of latent variable deep generative models to capture patterns in high-dimensional data. When the data comes from long-tailed distributions, as they often do for problems in healthcare, deep generative models may underfit. In Chapter 4 (Krishnan *et al.* , 2018) we investigate and remedy this pathology. Being able to fit these models well means that we can make use of the innovations in Chapter 3 to investigate the parameters of the model and understand the correlations that exist among features in high-dimensional data.

Temporal data: As diseases progress in a patient they manifest changes in clinical observations which then prompt changes in downstream treatment. Deep markov models (Krishnan *et al.* , 2017), in Chapter 6, are a flexible model family that practitioners can use for unsupervised learning of such data. The black-box variational learning algorithm we derive can be scaled up to learn DMMs from millions of datapoints through the use of GPUs.

Limited mechanistic knowledge: When modeling patient data from rare diseases, DMMs may overfit. In Chapter 7, we combine ideas from pharmacology with deep learning and design new neural architectures for use in the conditional probability distributions of DMMs. We show that this judicious use of domain knowledge improves the generalization of DMMs when data is scarce.

Dataset sizes: In Chapter 8 we show how deep generative models may be used to capture salient structure in post-treatment, privileged information and in doing so learn representations that can reduce the sample complexity of risk stratification models that make predictions using pre-treatment data. Chapter 5 shows how to fine-tune deep generative models with a little bit of supervision so that datapoints that are similar have similar latent representations.

9.1 Future directions for deep generative modeling

Inference as prediction The idea of posing an optimization problem, such as probabilistic inference in a graphical model, as prediction has roots that go back at least to the wake-sleep algorithm (Hinton *et al.* , 1995). However, there remain many questions of a statistical nature that arise from such a transformation. Statistical learning theory tells us about how well classifiers generalize when applied to unseen data. Little is known about whether such results may be extended to characterize the generalization of inference networks. For amortized variational inference to find footing within the statistical modeling workflow, we need ways to quantify the sample complexity, and generalization of the coupled systems comprising the deep generative model and the inference network. Such results will be necessary to trust the predictions obtained from inference networks when deployed to tackle real-world problems in domains such as healthcare.

Disentangled representation learning and identifiability The ability of deep generative models to model complex log likelihoods has led researchers to question whether this class of models can identify factors of variation in a dataset. In the context of MNIST, a disentangled model will map digit identity to some subset of the latent variables and digit style to others.

Several studies (Siddharth *et al.* , 2017; Kingma *et al.* , 2014) make use of supervision to control the information content captured by different latent variables – this is a powerful approach when supervision is available. Others claim that disentanglement may be obtained in a purely unsupervised manner, usually via some form of regularization on the inference network (Higgins *et al.* , 2016; Kim & Mnih, 2018; Chen *et al.* , 2018). However, the above studies leave open a crucial question: does there exist a coupling of model and learning algorithm that guarantees disentanglement *independent of the dataset*? Fortunately, and unsurprisingly, Locatello *et al.* (2019) show that in the absence of assumptions about the inductive biases of the generative model or the datasets that it is trained on, disentangled representation learning is impossible. Where does that leave us?

One of the most promising directions for future research towards the goal of disentangled representation learning is building deep generative models with *identifiable latent representations*. Classical identifiability is a statistical property of a model under which it is possible to uniquely determine the model parameters after observing an infinite number of observations. However deep generative models typically rely on conditional probability distributions defined using overparameterized neural networks rendering the unique identification of parameters improbable. Indeed one of the rationales behind the successes of neural networks is that overparameterization is crucial to learning such models via stochastic gradient based methods(Allen-Zhu *et al.* , 2019). But if we cannot identify the parameters of the model, perhaps we may identify the latent representations uniquely given infinite data. One of the pioneering works towards this end is that of (Khemakhem *et al.* , 2020) who make use of results from nonlinear independent component analysis (ICA) (Hyvarinen & Morioka, 2017; Hyvärinen & Pajunen, 1999; Hyvarinen *et al.* , 2019) to derive identifiability results for variational autoencoders. The extension of such results towards Deep Markov Models presents an intriguing opportunity to learn identifiable non-linear state space models where we can uniquely determine low-dimensional patient trajectories for high-dimensional, time-varying patient data.

Deep generative models with tractable likelihoods In this thesis, we made use of latent variable models for unsupervised learning. Furthermore, we assumed that latent variables had a lower dimensionality than the data. Dinh *et al.* (2016) present an alternative approach to unsupervised learning with latent variables. They assume the latent variable has the same dimensionality of the data, and the generative process comprises iterated, parameteric, transformations, each of which is constrained to be volume preserving. The resulting model has a tractable and differentiable log-likelihood and one can perform *exact* probabilistic inference by inverting each of the transformations in the generative model to obtain the posterior distribution. Although such models yield competitive results on image datasets, it would be interesting to study their utility on tabular datasets such as those found in electronic health record data.

The use of parameteric, volume preserving transformations has seen use not just in density estimation, but also in variational inference. Indeed, one of the canonical ways in which the complexity of the inference network may be improved is via normalizing flows. We refer the reader to (Kobyzev *et al.* , 2019; Papamakarios *et al.* , 2019) for a comprehensive review of such methods.

9.2 Future directions for machine learning in healthcare

In addition to the above methodological directions for future work, much remains to be done before we can answer some of the most pressing questions posed by clinical informatics.

Multi-scale, multi-modal deep generative models Over the next decade, medical institutions will collect a large amount of fine-grained data about the human body across different scales. At the micro scale, the collection of RNA and DNA sequencing data will give us a view into the cellular health of a patient. At the macro scale, the compilation of diagnosis codes, medical images and procedures will offer insights into the health of a patient's organ system. At the population scale, infection counts and community wellness data aggregated by non-governmental agencies will characterize the general health of large groups of people. Building hierarchical models of data at multiple scales and multiple modalities is a rich area for innovative applications of deep generative models (Shi *et al.* , 2019; Wu & Goodman, 2018; Wang *et al.* , 2014a), and new algorithms for inference and learning.

Success in this arena can inform solutions to new prediction problems that cannot solely be answered using data at a single scale. For example, a model looking to predict how likely a patient is to contract a life threatening C.difficile infection might require a patient's medical history at the macro scale, as well as the prevailing rates of infection in the hospital ward at the population scale.

Inductive biases for clinical data Hierarchical models are only part of the solution to decision making with multi-modal, multi-scale data. The other part of the solution lies in developing new parameterizations for the conditional probability distributions used in deep generative models. Many problems in healthcare will be fundamentally data limited, either due to the nature of the disease or due to socio-technical constraints on data access. Good inductive biases can prove crucial in building models that generalize well in the low-data regime. In Chapter 7, we saw the dramatic improvements obtained from the use of a judiciously chosen intervention effect function. Coming up with good neural architectures for deep neural networks that parameterize models of data such as lab results, x-rays, DNA and immunomics will no doubt require the expertise of pharmacologists, radiologists, geneticists and immunologists.

Disease progression Disease progression modeling (Cook & Bies, 2016) encompasses the use of discrete (Sukkar *et al.* , 2012) or continuous-time (Wang *et al.* , 2014b) statistical models to uncover patterns in longitudinal patient data. The ability of deep generative models to model data from millions of patients, each one with a potentially high-dimensional set of covariates, means that we are no longer computationally limited in which diseases we choose to study. Making use of sequential deep generative models to subtype low-dimensional patient trajectories (while correcting for variation in biomarkers due to treatments) can give clinicians insights into strata that exist within their patient populations. Characterizing the strata may then reveal known or unknown aspects of the disease, or give clinicians new ways to group patients.

Causal Inference There are several opportunities wherein the innovations within this thesis can play a role towards tackling problems of a causal nature. The one we highlight is the use of deep generative models as structural equation models (Pearl, 2012). Under the appropriate conditions where the sequential causal effect is identifiable (Pearl *et al.* , 2009), the DMM may be viewed as a structural equation model and be used to ask counterfactual queries of how patients behave under various treatment plans. Such a tool can give clinicians the ability to gauge the success of a chosen longitudinal treatment plan not just in terms of the primary biomarkers used to characterize disease burden, but also in terms of biomarkers that characterize related comorbidities.

Clinical Deployment On a more humbling note, while there many *known unknowns* in the application of machine learning for problems in healthcare, there are a far greater number of *unknown unknowns*. The careful deployment, study and characterization of decision support tools powered by machine learning is vital to shed light on what problems remain to be characterized before patients and doctors can make use of the insights learned from data.

Appendix A

Model configurations

We detail the model configurations used in the experiments within Chapter 5. We present the architectures in a format used by Keras (Chollet *et al.* , 2015).

A.0.1 Pinwheel Dataset

Encoder: $p(z|x)$**:**

- $x \rightarrow$ Dense(20, 'relu')
- $h_1 \rightarrow$ Dense(20, 'relu') $\rightarrow h_2$
- $h_2 \rightarrow$ Dense(1) $\rightarrow \mu$
- $h_2 \rightarrow$ Dense(1) $\rightarrow \log \Sigma$

Decoder: $p(x|z)$**:**

- $z \rightarrow$ Dense(20, 'relu')
- $h_1 \rightarrow$ Dense(20, 'relu')
- $h_2 \rightarrow$ Dense(2) $\rightarrow \mu_{\text{obs}}$

Reasoning Model: $p(z|Q)$**:**

- $\{x_1, \ldots, x_Q\} \to p(z|x)$ **(Elementwise)**

- $\{[\mu_1, \log \Sigma_1], \ldots, [\mu_Q, \log \Sigma_Q]\}$
  $\to$ PermutationEquivariant(20,'elu')

- $\{h_1^1, \ldots, h_Q^1\} \to$ PermutationEquivariant(20,'elu')

- $\{h_1^2, \ldots, h_Q^2\} \to$ PermutationInvariant(1) $\to \mu$

- $\{h_1^2, \ldots, h_Q^2\} \to$ PermutationInvariant(1) $\to \log \Sigma$

A.0.2 MiniImagenet Dataset

Embedding Network $f(x) \to x'$**:**

- $x \to$ ResNet18 (He *et al.* , 2016) Conv Layers (see below) $\to h_1$
- $h_1 \to$ AveragePooling $\to x'$

Encoder: $p(z|x')$**:**

- $x' \to$ Dense(512, 'relu') $\to h_1$
- $h_1 \to$ Dense(128, 'linear') $\to \mu$
- $h_1 \to$ Dense(128, 'linear') $\to \sigma$

Decoder: $p(x'|z)$**:**

- $z \to$ Dense(512, 'relu') $\to h_1$

- $h_1 \to$ Dense(256, 'linear') $\to \mu_{obs}$

Reasoning Model: $p(z|Q)$**:**

- $\{x_1, \ldots, x_Q\} \rightarrow p(z|x)$ **(Elementwise)**
- $\{[\mu_1, \log \Sigma_1], \ldots, [\mu_Q, \log \Sigma_Q]\}$
  $\rightarrow$ PermutationEquivariant(2048,'linear')
- $\{h_1^2, \ldots, h_Q^2\} \rightarrow$ PermutationInvariant(128) $\rightarrow \mu$
- $\{h_1^2, \ldots, h_Q^2\} \rightarrow$ PermutationInvariant(128) $\rightarrow \log \Sigma$

Training Details:

We take $|Qs|= 1, |Q_{ns}|= 5$, learning rate $= 5e - 5$.

A.0.3 MNIST Dataset

Encoder: $p(z|x)$**:**

- $x \rightarrow$ Flatten() $\rightarrow h_1$
- $h_1 \rightarrow$ Dense(500, 'relu') $\rightarrow h_2$
- $h_2 \rightarrow$ Dense(500, 'relu') $\rightarrow h_3$
- $h_3 \rightarrow$ Dense(2) $\rightarrow \mu$
- $h_3 \rightarrow$ Dense(2) $\rightarrow \sigma$

Decoder: $p(x|z)$**:**

- $z \rightarrow$ Dense(500, 'relu') $\rightarrow h_1$
- $h_1 \rightarrow$ Dense(784, 'sigmoid') $\rightarrow h_2$
- $h_2 \rightarrow$ Reshape((28,28)) $\rightarrow \mu$

Reasoning Model: $p(z|Q)$**:**

- $\{x_1, \ldots, x_Q\} \rightarrow p(z|x)$ **(Elementwise)**
- $\{[\mu_1, \log \Sigma_1], \ldots, [\mu_Q, \log \Sigma_Q]\}$
  $\rightarrow$ PermutationEquivariant(20,'relu')
- $\{h_1^2, \ldots, h_Q^2\} \rightarrow$ PermutationInvariant(2) $\rightarrow \mu$
- $\{h_1^2, \ldots, h_Q^2\} \rightarrow$ PermutationInvariant(2) $\rightarrow \log \Sigma$

Training Details: We take $|Qs|= 5, |Q_{ns}|= 5$, learning rate $= 1e - 4$.

Appendix B

Model configurations

We detail the model configurations used in the experiments within Chapter 6.

In each instance, we detail the model architecture used in the generative model (comprising a transition and emission function) as well as the model used for posterior inference. “L” denotes a linear layer with the values in parenthesis denoting the dimensions of the transformation. “NL” denotes the application of a non-linearity (specified in the caption). Since parameters in the model are shared across time, we describe the architecture at a single time-step. The row denoted “Inference” is the architecture used for performing inference. In the case of the bidirectional RNN, we concatenate the outputs from the forward and reverse RNN to perform prediction of the posterior means and log covariances. The recognition network predicts the posterior mean and log-covariance. The two quantities are predicted with a shared base network feeding into a separate final linear layer (i.e the last linear layer in the row “Inference” is different for the function used to predict the posterior mean and the posterior log-covariance). Square braces indicate a vector concatenation operation.

Table B.1 describes the model architecture used in the synthetic experiments. We detail the architectures used for inference in “MF-LR” and “ST-LR”. The other inference algorithms involve different structures in the LSTM-RNN module but are otherwise identical. The “combiner function” is detailed by the mapping from $[h_t; z_{t-1}] \rightarrow \mu_q$.

Table B.2 describes the architecture used for the polyphonic dataset and in Table B.3, we describe the architecture used for experiments on medical data.

Table B.1: Synthetic Experiments (ReLU was used as the non-linearity)

Inference (MF-LR)	x_t →LNL120→LNL2020 →LNL2020 →bLSTM22020 →LNL4020 →LNL2020 →LIN201 →μ_q or $\log \sigma_q^2$
Inference (ST-LR)	x_t →LNL120→LNL2020 →LNL2020 →bLSTM22020 →LNL4020 →h_t $[h_t; z_{t-1}]$ →LNL20+120 →LNL2020→LIN201 →μ_q or $\log \sigma_q^2$
Emission	Fixed
Transition	Fixed

Table B.2: Polyphonic Experiments (Tanh was used as the non-linearity).

Inference	x_t →LNL88200→LNL200200 →LNL200200 →LSTM2200200 →LIN200200 →μ_q or $\log \sigma_q^2$
Emission	Z→LNL200200 →LNL200200 →LNL200200 →LIN200200 →Sigmoid
Transition (μ_p)	Z→LNL200200 →LNL200200 →LIN200200
Transition ($\log \sigma_p^2$)	Z→LNL200200 →LNL200200 →LIN200200

Table B.3: Medical Experiments (Tanh was used as the non-linearity). We describe the "E:NL-T:NL" model. The observations were 48 dimensional of which there were 4 lab indicators that we treat separately to perform do-calculus.

Inference	$[x_t; u_t]$ →LNL48+8200→LNL200300 →LNL200200 →bLSTM2200200 →LIN40020 →μ_q or $\log \sigma_q^2$
Emission (Lab Indicators i_t)	z_t →LNL20200 →LIN2004 →Sigmoid
Emission (Lab Values, Diagnosis Codes)	$[z_t; i_t]$ →LNL20+4200 →LIN20044 →Sigmoid
Transition (μ_p)	$[z_t; u_t]$ →LNL20+8200 →LNL200200 →LIN20020
Transition ($\log \sigma_p^2$)	Fixed with dimension 20 (Sampled from Uniform(-1,1)

Bibliography

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

Ahlqvist, Emma, Storm, Petter, Käräjämäki, Annemari, Martinell, Mats, Dorkhan, Mozhgan, Carlsson, Annelie, Vikman, Petter, Prasad, Rashmi B, Aly, Dina Mansour, Almgren, Peter, *et al.* . 2018. Novel subgroups of adult-onset diabetes and their association with outcomes: a data-driven cluster analysis of six variables. *The Lancet Diabetes & endocrinology*, **6**(5), 361–369.

Albers, David J, Hripcsak, George, & Schmidt, Michael. 2012. Population physiology: leveraging electronic health record data to understand human endocrine dynamics. *PLoS One*, **7**(12), e48058.

Allen-Zhu, Zeyuan, Li, Yuanzhi, & Liang, Yingyu. 2019. Learning and generalization in overparameterized neural networks, going beyond two layers. *Pages 6158–6169 of: Advances in neural information processing systems.*

Almeida, Felipe, & Xexéo, Geraldo. 2019. Word embeddings: A survey. *arXiv preprint arXiv:1901.09069.*

Anandkumar, Animashree, Hsu, Daniel, & Kakade, Sham M. 2012. A method of moments for mixture models and hidden Markov models. *Pages 33–1 of: Conference on Learning Theory.*

Archer, Evan, Park, Il Memming, Buesing, Lars, Cunningham, John, & Paninski, Liam. 2015. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367.*

Baeza-Yates, Ricardo, Ribeiro-Neto, Berthier, *et al.* . 1999. *Modern information retrieval.* Vol. 463. ACM press New York.

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

Bar-Hillel, Aharon, Hertz, Tomer, Shental, Noam, & Weinshall, Daphna. 2005. Learning a mahalanobis metric from equivalence constraints. *JMLR*.

Bauer, Matthias, Rojas-Carulla, Mateo, Bartłomiej Świątkowski, Jakub, Schölkopf, Bernhard, & Turner, Richard E. 2017. Discriminative k-shot learning using probabilistic models. *arXiv preprint arXiv:1706.00326*.

Baydin, Atılım Günes, Pearlmutter, Barak A, Radul, Alexey Andreyevich, & Siskind, Jeffrey Mark. 2017. Automatic differentiation in machine learning: a survey. *The Journal of Machine Learning Research*, **18**(1), 5595–5637.

Bayer, Justin, & Osendorfer, Christian. 2014. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*.

Bengio, Yoshua, Ducharme, Réjean, Vincent, Pascal, & Jauvin, Christian. 2003. A neural probabilistic language model. *JMLR*.

Bica, Ioana, Alaa, Ahmed M, Lambert, Craig, & van der Schaar, Mihaela. 2020. From real-world patient data to individualized treatment effects using machine learning: Current and future methods to address underlying challenges. *Clinical Pharmacology & Therapeutics*.

Bingham, Eli, Chen, Jonathan P, Jankowiak, Martin, Obermeyer, Fritz, Pradhan, Neeraj, Karaletsos, Theofanis, Singh, Rohit, Szerlip, Paul, Horsfall, Paul, & Goodman, Noah D. 2019. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, **20**(1), 973–978.

Blei, David M, Ng, Andrew Y, & Jordan, Michael I. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, **3**(Jan), 993–1022.

Bojarski, Mariusz, Del Testa, Davide, Dworakowski, Daniel, Firner, Bernhard, Flepp, Beat, Goyal, Prasoon, Jackel, Lawrence D, Monfort, Mathew, Muller, Urs, Zhang, Jiakai, *et al.* . 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Boulanger-Lewandowski, Nicolas, Bengio, Yoshua, & Vincent, Pascal. 2012. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*.

Bowman, Samuel R, Vilnis, Luke, Vinyals, Oriol, Dai, Andrew M, Jozefowicz, Rafal, & Bengio, Samy. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Breiman, Leo. 2001. Statistical modeling: The two cultures. *Statistical Science*.

Briegel, Thomas, & Tresp, Volker. 1999. Fisher scoring and a mixture of modes approach for approximate inference and learning in nonlinear state space models. *Pages 403–409 of: Advances in Neural Information Processing Systems.*

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

Butler, A., Hoffman, P., Smibert, P., Papalexi, E., & Satija, R. 2018. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature biotechnology 36.5.*

Caruana, Rich. 1997. Multitask learning. *Machine learning*, **28**(1), 41–75.

Chakraborty, Bibhas. 2013. *Statistical methods for dynamic treatment regimes.* Springer.

Che, Zhengping, Purushotham, Sanjay, Li, Guangyu, Jiang, Bo, & Liu, Yan. 2018a. Hierarchical deep generative models for multi-rate multivariate time series. *Pages 784–793 of: International Conference on Machine Learning.*

Che, Zhengping, Purushotham, Sanjay, Cho, Kyunghyun, Sontag, David, & Liu, Yan. 2018b. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, **8**(1), 1–12.

Chen, Jonathan H, & Asch, Steven M. 2017. Machine learning and prediction in medicine—beyond the peak of inflated expectations. *The New England journal of medicine*, **376**(26), 2507.

Chen, Ricky TQ, Li, Xuechen, Grosse, Roger B, & Duvenaud, David K. 2018. Isolating sources of disentanglement in variational autoencoders. *Pages 2610–2620 of: Advances in Neural Information Processing Systems.*

Chen, Xi, Kingma, Diederik P, Salimans, Tim, Duan, Yan, Dhariwal, Prafulla, Schulman, John, Sutskever, Ilya, & Abbeel, Pieter. 2016. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731.*

Chen, Xinxiong, Liu, Zhiyuan, & Sun, Maosong. 2014. A Unified Model for Word Sense Representation and Disambiguation. *In: EMNLP.*

Chiappa, Silvia, Racaniere, Sébastien, Wierstra, Daan, & Mohamed, Shakir. 2017. Recurrent environment simulators. *arXiv preprint arXiv:1704.02254.*

Choi, Edward, Bahadori, Mohammad Taha, Schuetz, Andy, Stewart, Walter F, & Sun, Jimeng. 2016a. Doctor ai: Predicting clinical events via recurrent neural networks. *Pages 301–318 of: Machine Learning for Healthcare Conference.*

Choi, Edward, Bahadori, Mohammad Taha, Sun, Jimeng, Kulas, Joshua, Schuetz, Andy, & Stewart, Walter. 2016b. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. *Pages 3504–3512 of: Advances in Neural Information Processing Systems.*

Choi, Youngduck, Yi-I Chiu, Chill, & Sontag, David. 2016c. Learning Low-Dimensional Representations of Medical Concepts. *In: AMIA*.

Chollet, François, *et al.* . 2015. *Keras*. `https://github.com/keras-team/keras`.

Chung, Junyoung, Gulcehre, Caglar, Cho, KyungHyun, & Bengio, Yoshua. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Chung, Junyoung, Kastner, Kyle, Dinh, Laurent, Goel, Kratarth, Courville, Aaron C, & Bengio, Yoshua. 2015. A recurrent latent variable model for sequential data. *Pages 2980–2988 of: Advances in neural information processing systems*.

Church, Kenneth Ward, & Hanks, Patrick. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*.

Collins, Michael, Dasgupta, Sanjoy, & Schapire, Robert E. 2001. A generalization of principal component analysis to the exponential family. *In: NIPS*.

Collobert, Ronan, & Weston, Jason. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. *In: ICML*.

Collobert, Ronan, Bengio, Samy, & Mariéthoz, Johnny. 2002. *Torch: a modular machine learning software library*. Tech. rept. Idiap.

Cook, Sarah F, & Bies, Robert R. 2016. Disease progression modeling: key concepts and recent developments. *Current pharmacology reports*, **2**(5), 221–230.

Cox, David R. 1972. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, **34**(2), 187–202.

Cox, David Roxbee. 2018. *Analysis of survival data*. Routledge.

Cremer, Chris, Li, Xuechen, & Duvenaud, David. 2018. Inference Suboptimality in Variational Autoencoders. *Pages 1078–1086 of: International Conference on Machine Learning*.

Cybenko, George. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, **2**(4), 303–314.

De Vine, Lance, Zuccon, Guido, Koopman, Bevan, Sitbon, Laurianne, & Bruza, Peter. 2014. Medical semantic similarity with a neural language model. *In: CIKM*.

Dempsey, Walter H, Moreno, Alexander, Scott, Christy K, Dennis, Michael L, Gustafson, David H, Murphy, Susan A, & Rehg, James M. 2017. iSurvive: An Interpretable, Event-time Prediction Model for mHealth. *Pages 970–979 of: International Conference on Machine Learning*.

Dinh, Laurent, Sohl-Dickstein, Jascha, & Bengio, Samy. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

Doyle, Orla M, Westman, Eric, Marquand, Andre F, Mecocci, Patrizia, Vellas, Bruno, Tsolaki, Magda, Kłoszewska, Iwona, Soininen, Hilkka, Lovestone, Simon, Williams, Steve CR, *et al.* . 2014. Predicting progression of Alzheimer's disease using ordinal regression. *PloS one*, **9**(8).

Du, Yilun, & Narasimhan, Karthic. 2019. Task-Agnostic Dynamics Priors for Deep Reinforcement Learning. *Pages 1696–1705 of: International Conference on Machine Learning.*

Duckworth, D. 2016. *Kalman filter, kalman smoother, and em library for python.*

Dziugaite, Gintare Karolina, Roy, Daniel M, & Ghahramani, Zoubin. 2015. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906.*

Edwards, Harrison, & Storkey, Amos. 2016. Towards a neural statistician. *arXiv preprint arXiv:1606.02185.*

Engel, Jesse, Hoffman, Matthew, & Roberts, Adam. 2018. Latent Constraints: Learning to Generate Conditionally from Unconditional Generative Models. *In: ICLR.*

Erhan, Dumitru, Bengio, Yoshua, Courville, Aaron, & Vincent, Pascal. 2009. Visualizing higher-layer features of a deep network.

Evain, Simon, & Benzekry, Sebastien. 2016. Mathematical modeling of tumor and metastatic growth when treated with sunitinib.

Fabius, Otto, & van Amersfoort, Joost R. 2014. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581.*

Fano, Robert M. 1949. *The transmission of information.* The MIT Press.

Fefferman, Charles, Mitter, Sanjoy, & Narayanan, Hariharan. 2016. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, **29**(4), 983–1049.

Finkelstein, Lev, Gabrilovich, Evgeniy, Matias, Yossi, Rivlin, Ehud, Solan, Zach, Wolfman, Gadi, & Ruppin, Eytan. 2001. Placing search in context: The concept revisited. *In: WWW.*

Finlayson, Samuel G, LePendu, Paea, & Shah, Nigam H. 2014. Building the graph of medicine from millions of clinical narratives. *Scientific data.*

Finn, Chelsea, Abbeel, Pieter, & Levine, Sergey. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *ICML.*

Fleisher, Brett, Brown, Ashley N, & Ait-Oudhia, Sihem. 2017. Application of pharmacometrics and quantitative systems pharmacology to cancer therapy: The example of luminal a breast cancer. *Pharmacological Research*, **124**, 20–33.

Fraccaro, Marco, Sønderby, Søren Kaae, Paquet, Ulrich, & Winther, Ole. 2016. Sequential neural models with stochastic layers. *Pages 2199–2207 of: Advances in neural information processing systems.*

Futoma, Joseph, Sendak, Mark, Cameron, C Blake, & Heller, Katherine. 2016. Scalable Modeling of Multivariate Longitudinal Data for Prediction of Chronic Kidney Disease Progression. *arXiv preprint arXiv:1608.04615.*

Gabler, Nicole B, Duan, Naihua, Liao, Diana, Elmore, Joann G, Ganiats, Theodore G, & Kravitz, Richard L. 2009. Dealing with heterogeneity of treatment effects: is the literature up to the challenge? *Trials*, **10**(1), 43.

Gan, Zhe, Li, Chunyuan, Henao, Ricardo, Carlson, David E, & Carin, Lawrence. 2015. Deep temporal sigmoid belief networks for sequence modeling. *Pages 2467–2475 of: Advances in Neural Information Processing Systems.*

Gao, Yuanjun, Archer, Evan W, Paninski, Liam, & Cunningham, John P. 2016. Linear dynamical neural population models through nonlinear embeddings. *Pages 163–171 of: Advances in neural information processing systems.*

Geng, Changran, Paganetti, Harald, & Grassberger, Clemens. 2017. Prediction of treatment response for combined chemo-and radiation therapy for non-small cell lung cancer patients using a bio-mathematical model. *Scientific reports*, **7**(1), 1–12.

Ghahramani, Zoubin, & Heller, Katherine A. 2005. Bayesian sets. *In: NIPS.*

Ghahramani, Zoubin, & Roweis, Sam T. 1999. Learning nonlinear dynamical systems using an EM algorithm. *Pages 431–437 of: Advances in neural information processing systems.*

Gibson, WA. 1960. Nonlinear factors in two dimensions. *Psychometrika*, **25**(4), 381–392.

Glorot, Xavier, & Bengio, Yoshua. 2010. Understanding the difficulty of training deep feedforward neural networks. *In: AISTATS.*

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, & Bengio, Yoshua. 2014. Generative adversarial nets. *Pages 2672–2680 of: Advances in neural information processing systems.*

Grassberger, C, & Paganetti, H. 2016. Methodologies in the modeling of combined chemo-radiation treatments. *Physics in Medicine & Biology*, **61**(21), R344.

Graves, Alex, Wayne, Greg, & Danihelka, Ivo. 2014. Neural Turing Machines. *arXiv preprint arXiv:1410.5401.*

Gregor, Karol, Danihelka, Ivo, Graves, Alex, Rezende, Danilo Jimenez, & Wierstra, Daan. 2015. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623.*

Gu, Shixiang, Ghahramani, Zoubin, & Turner, Richard E. 2015. Neural adaptive sequential monte carlo. *Pages 2629–2637 of: Advances in Neural Information Processing Systems.*

Gulshan, V, Peng, L, Coram, M, & et al. 2016. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, **316**(22), 2402–2410.

Halpern, Yoni, & Sontag, David. 2013. Unsupervised Learning of Noisy-Or Bayesian Networks. *Page 272 of: Uncertainty in Artificial Intelligence.* Citeseer.

Harper, F Maxwell, & Konstan, Joseph A. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS).*

He, Junxian, Spokoyny, Daniel, Neubig, Graham, & Berg-Kirkpatrick, Taylor. 2019. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534.*

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, & Sun, Jian. 2016. Deep residual learning for image recognition. *In: CVPR.*

Helmlinger, Gabriel, Sokolov, Victor, Peskov, Kirill, Hallow, Karen M, Kosinsky, Yuri, Voronova, Veronika, Chu, Lulu, Yakovleva, Tatiana, Azarov, Ivan, Kaschek, Daniel, *et al.* . 2019. Quantitative Systems Pharmacology: An Exemplar Model-Building Workflow With Applications in Cardiovascular, Metabolic, and Oncology Drug Development. *CPT: pharmacometrics & systems pharmacology*, **8**(6), 380–395.

Hernán, Miguel, & Robins, Jamie. 2020. *Causal Inference: What If.* Boca Raton: Chapman & Hall/CRC.

Higgins, Irina, Matthey, Loic, Pal, Arka, Burgess, Christopher, Glorot, Xavier, Botvinick, Matthew, Mohamed, Shakir, & Lerchner, Alexander. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework.

Hinton, Geoffrey E, & Salakhutdinov, Ruslan R. 2009. Replicated softmax: an undirected topic model. *Pages 1607–1614 of: Advances in Neural Information Processing Systems.*

Hinton, Geoffrey E, Dayan, Peter, Frey, Brendan J, & Neal, Radford M. 1995. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, **268**(5214), 1158–1161.

Hjelm, R Devon, Cho, Kyunghyun, Chung, Junyoung, Salakhutdinov, Russ, Calhoun, Vince, & Jojic, Nebojsa. 2016. Iterative Refinement of Approximate Posterior for Training Directed Belief Networks. *In: NIPS.*

Hoffman, Matthew D, Blei, David M, Wang, Chong, & Paisley, John. 2013. Stochastic variational inference. *The Journal of Machine Learning Research*, **14**(1), 1303–1347.

Honkela, Antti, & Valpola, Harri. 2004. Variational learning and bits-back coding: an information-theoretic view to Bayesian learning. *IEEE transactions on Neural Networks*, **15**(4), 800–810.

Hsu, Daniel, Kakade, Sham M, & Zhang, Tong. 2012. A spectral algorithm for learning hidden Markov models. *Journal of Computer and System Sciences*, **78**(5), 1460–1480.

Hu, Yifan, Koren, Yehuda, & Volinsky, Chris. 2008. Collaborative filtering for implicit feedback datasets. *Pages 263–272 of: 2008 Eighth IEEE International Conference on Data Mining.* Ieee.

Huang, Eric H., Socher, Richard, Manning, Christopher D., & Ng, Andrew Y. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. *In: ACL.*

Hutchinson, Lucy, Steiert, Bernhard, Soubret, Antoine, Wagg, Jonathan, Phipps, Alex, Peck, Richard, Charoin, Jean-Eric, & Ribba, Benjamin. 2019. Models and Machines: How Deep Learning Will Take Clinical Pharmacology to the Next Level. *CPT: pharmacometrics & systems pharmacology*, **8**(3), 131–134.

Hyvärinen, Aapo, & Pajunen, Petteri. 1999. Nonlinear independent component analysis: Existence and uniqueness results. *Neural networks*, **12**(3), 429–439.

Hyvarinen, Aapo, Sasaki, Hiroaki, & Turner, Richard. 2019. Nonlinear ICA using auxiliary variables and generalized contrastive learning. *Pages 859–868 of: The 22nd International Conference on Artificial Intelligence and Statistics.*

Hyvarinen, AJ, & Morioka, Hiroshi. 2017. Nonlinear ICA of temporally dependent stationary sources. Proceedings of Machine Learning Research.

Iyyer, Mohit, Manjunatha, Varun, Boyd-Graber, Jordan, & Daumé III, Hal. 2015. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. *In: ACL.*

Jaakkola, Tommi S, & Haussler, David. 2007. Exploiting generative models in discriminative classifiers. *In: NIPS.*

Jacobs, Robert A, Jordan, Michael I, Nowlan, Steven J, & Hinton, Geoffrey E. 1991. Adaptive mixtures of local experts. *Neural computation*, **3**(1), 79–87.

Jacobus, SJ, Rajkumar, S Vincent, Weiss, M, Stewart, Alexander Keith, Stadtmauer, EA, Callander, NS, Dreosti, Lydia M, Lacy, MQ, & Fonseca, Rafael. 2016. Randomized phase III trial of consolidation therapy with bortezomib–lenalidomide–Dexamethasone (VRd) vs bortezomib–dexamethasone (Vd) for patients with multiple myeloma who have completed a dexamethasone based induction regimen. *Blood cancer journal*, **6**(7), e448–e448.

Jankowiak, Martin, & Obermeyer, Fritz. 2018. Pathwise Derivatives Beyond the Reparameterization Trick. *Pages 2235–2244 of: International Conference on Machine Learning.*

Järvelin, K., & Kekäläinen, J. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, **20**(4), 422–446.

Jeffreys, Harold. 1998. *The Theory of Probability.* OUP Oxford.

Jernite, Yacine, Halpern, Yonatan, & Sontag, David. 2013. Discovering hidden variables in noisy-or networks using quartet tests. *Pages 2355–2363 of: Advances in Neural Information Processing Systems.*

Johnson, Matthew, Duvenaud, David K, Wiltschko, Alex, Adams, Ryan P, & Datta, Sandeep R. 2016. Composing graphical models with neural networks for structured representations and fast inference. *Pages 2946–2954 of: Advances in neural information processing systems.*

Jones, Christopher S. 2006. A nonlinear factor analysis of S&P 500 index option returns. *The Journal of Finance*, **61**(5), 2325–2363.

Jordan, Michael I, & Jacobs, Robert A. 1994. Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, **6**(2), 181–214.

Jordan, Michael I, Ghahramani, Zoubin, Jaakkola, Tommi S, & Saul, Lawrence K. 1999. An introduction to variational methods for graphical models. *Machine learning*, **37**(2), 183–233.

Jusko, William J. 2013. Moving from basic toward systems pharmacodynamic models. *Journal of pharmaceutical sciences*, **102**(9), 2930–2940.

Jutten, C. 2003. Advances in nonlinear blind source separation. *Pages 245–256 of: 4th Int. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003), Nara.*

Kaplan, Edward L, & Meier, Paul. 1958. Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, **53**(282), 457–481.

Khemakhem, Ilyes, Kingma, Diederik, Monti, Ricardo, & Hyvarinen, Aapo. 2020. Variational autoencoders and nonlinear ica: A unifying framework. *Pages 2207–2217 of: International Conference on Artificial Intelligence and Statistics.*

Kim, Hyunjik, & Mnih, Andriy. 2018. Disentangling by Factorising. *Pages 2649–2658 of: International Conference on Machine Learning.*

Kim, Yoon, Wiseman, Sam, Miller, Andrew, Sontag, David, & Rush, Alexander. 2018. Semi-Amortized Variational Autoencoders. *Pages 2678–2687 of: International Conference on Machine Learning.*

Kingma, Diederik P, & Ba, Jimmy. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, Diederik P., & Welling, Max. 2014. Auto-encoding variational bayes. *In: Proceedings of the International Conference on Learning Representations (ICLR)*, vol. 2.

Kingma, Diederik P, Mohamed, Shakir, Rezende, Danilo Jimenez, & Welling, Max. 2014. Semi-supervised learning with deep generative models. *In: Advances in Neural Information Processing Systems*.

Klein, Christoph A. 2009. Parallel progression of primary tumours and metastases. *Nature Reviews Cancer*, **9**(4), 302–312.

Klein, John P, & Moeschberger, Melvin L. 2006. *Survival analysis: techniques for censored and truncated data*. Springer Science & Business Media.

Kobyzev, Ivan, Prince, Simon, & Brubaker, Marcus A. 2019. Normalizing flows: Introduction and ideas. *arXiv preprint arXiv:1908.09257*.

Koller, Daphne, Friedman, Nir, & Bach, Francis. 2009. *Probabilistic graphical models: principles and techniques*. The MIT Press.

Koutnik, Jan, Greff, Klaus, Gomez, Faustino, & Schmidhuber, Juergen. 2014. A Clockwork RNN. *Pages 1863–1871 of: International Conference on Machine Learning*.

Krishnan, Rahul G, Shalit, Uri, & Sontag, David. 2017. Structured Inference Networks for Nonlinear State Space Models. *In: AAAI*.

Krishnan, Rahul G, Liang, Dawen, & Hoffman, Matthew. 2018. On the challenges of learning with inference networks on sparse, high-dimensional data. *In: Proceedings of the Twenty-first Conference on Artificial Intelligence and Statistics*.

Krizhevsky, Alex, Sutskever, Ilya, & Hinton, Geoffrey E. 2012. Imagenet classification with deep convolutional neural networks. *Pages 1097–1105 of: Advances in neural information processing systems*.

Kumar, Shaji, Flinn, Ian, Richardson, Paul G, Hari, Parameswaran, Callander, Natalie, Noga, Stephen J, Stewart, A Keith, Turturro, Francesco, Rifkin, Robert, Wolf, Jeffrey, *et al.* . 2012. Randomized, multicenter, phase 2 study (EVOLUTION) of combinations of bortezomib, dexamethasone, cyclophosphamide, and lenalidomide in previously untreated multiple myeloma. *Blood, The Journal of the American Society of Hematology*, **119**(19), 4375–4382.

Lake, Brenden M, Salakhutdinov, Ruslan R, & Tenenbaum, Josh. 2013. One-shot learning by inverting a compositional causal process. *In: NIPS*.

Landauer, Thomas K, Foltz, Peter W, & Laham, Darrell. 1998. An introduction to latent semantic analysis. *Discourse processes.*

Larochelle, Hugo, & Murray, Iain. 2011. The neural autoregressive distribution estimator. *Pages 29–37 of: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics.*

Larochelle, Hugo, Bengio, Yoshua, Louradour, Jérôme, & Lamblin, Pascal. 2009. Exploring strategies for training deep neural networks. *Journal of machine learning research*, **10**(Jan), 1–40.

Lawrence, Neil D. 2004. Gaussian process latent variable models for visualisation of high dimensional data. *Pages 329–336 of: Advances in neural information processing systems.*

LeCun, Yann. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/.*

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, & Haffner, Patrick. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), 2278–2324.

LeCun, Yann, Bengio, Yoshua, & Hinton, Geoffrey. 2015. Deep learning. *nature*, **521**(7553), 436–444.

Lee, Wonyeol, Yu, Hangyeol, & Yang, Hongseok. 2018. Reparameterization gradient for non-differentiable models. *Pages 5553–5563 of: Advances in Neural Information Processing Systems.*

Lenert, Leslie A, Lurie, Jon, Sheiner, Lewis B, Coleman, Robert, Klostermann, Heidrun, & Blaschke, Terrence F. 1992. Advanced computer programs for drug dosing that combine pharmacokinetic and symbolic modeling of patients. *Computers and biomedical research*, **25**(1), 29–42.

Lewis, David D, Yang, Yiming, Rose, Tony G, & Li, Fan. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, **5**(Apr), 361–397.

Li, Chongxuan, Zhu, Jun, Shi, Tianlin, & Zhang, Bo. 2015a. Max-margin deep generative models. *Pages 1837–1845 of: Advances in neural information processing systems.*

Li, Yujia, Swersky, Kevin, & Zemel, Rich. 2015b. Generative moment matching networks. *Pages 1718–1727 of: International Conference on Machine Learning.*

Lim, Bryan. 2018. Forecasting treatment responses over time using recurrent marginal structural networks. *Pages 7483–7493 of: Advances in Neural Information Processing Systems.*

Lipton, Zachary C, Kale, David C, Elkan, Charles, & Wetzell, Randall. 2015. Learning to diagnose with LSTM recurrent neural networks. *arXiv preprint arXiv:1511.03677*.

Locatello, Francesco, Bauer, Stefan, Lucic, Mario, Raetsch, Gunnar, Gelly, Sylvain, Schölkopf, Bernhard, & Bachem, Olivier. 2019. Challenging common assumptions in the unsupervised learning of disentangled representations. *Pages 4114–4124 of: international conference on machine learning*.

Lopez-Paz, David, Bottou, Léon, Schölkopf, Bernhard, & Vapnik, Vladimir. 2015. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*.

Lucas, James, Tucker, George, Grosse, Roger B, & Norouzi, Mohammad. 2019. Don't Blame the ELBO! A Linear VAE Perspective on Posterior Collapse. *Pages 9403–9413 of: Advances in Neural Information Processing Systems*.

Maaten, Laurens van der, & Hinton, Geoffrey. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, **9**(Nov), 2579–2605.

MacKay, David JC, & Mac Kay, David JC. 2003. *Information theory, inference and learning algorithms*. Cambridge university press.

Maddison, Chris J, Mnih, Andriy, & Teh, Yee Whye. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Marlin, Benjamin M, & Zemel, Richard S. 2009. Collaborative prediction and ranking with non-random missing data. *Pages 5–12 of: Proceedings of the third ACM conference on Recommender systems*.

Mertens, Esther CA, Deković, Maja, Asscher, Jessica J, & Manders, Willeke A. 2017. Heterogeneity in response during multisystemic therapy: Exploring subgroups and predictors. *Journal of abnormal child psychology*, **45**(7), 1285–1295.

Miao, Yishu, Yu, Lei, & Blunsom, Phil. 2016. Neural Variational Inference for Text Processing. *In: International Conference on Machine Learning (ICML)*.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, & Dean, Jeff. 2013a. Distributed representations of words and phrases and their compositionality. *Pages 3111–3119 of: Advances in neural information processing systems*.

Mikolov, Tomas, Chen, Kai, Corrado, Greg, & Dean, Jeffrey. 2013b. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mishra, Nikhil, Rohaninejad, Mostafa, Chen, Xi, & Abbeel, Pieter. 2018. Meta-learning with temporal convolutions. *ICLR*.

Mnih, Andriy, & Gregor, Karol. 2014. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*.

Mohamed, Shakir, & Lakshminarayanan, Balaji. 2016. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*.

Mohamed, Shakir, Rosca, Mihaela, Figurnov, Michael, & Mnih, Andriy. 2019. Monte carlo gradient estimation in machine learning. *arXiv preprint arXiv:1906.10652*.

Mohan, Karthika, & Pearl, Judea. 2018. Graphical models for processing missing data. *arXiv preprint arXiv:1801.03583*.

Multiple Myeloma Research Foundation & others. 2011. Relating clinical outcomes in multiple myeloma to personal assessment of genetic profile (CoM-Mpass). *Clinical Trials website.* `https://clinicaltrials.gov/ct2/show/NCT01454297`.

Munkhdalai, Tsendsuren, & Yu, Hong. 2017. Meta networks. *ICML*.

Murphy, Kevin P. 2012. *Machine Learning: A Probabilistic Perspective*. MIT Press.

Neal, Radford M. 1993. *Probabilistic inference using Markov chain Monte Carlo methods*. Department of Computer Science, University of Toronto Toronto, ON, Canada.

Ning, Xia, & Karypis, George. 2011. Slim: Sparse linear methods for top-n recommender systems. *Pages 497–506 of: 2011 IEEE 11th International Conference on Data Mining*. IEEE.

Norris, Megan, & Lecavalier, Luc. 2010. Evaluating the use of exploratory factor analysis in developmental disability psychological research. *Journal of autism and developmental disorders*, **40**(1), 8–20.

Norton, Larry. 2014. Cancer log-kill revisited. *American Society of Clinical Oncology Educational Book*, **34**(1), 3–7.

Oh, Junhyuk, Guo, Xiaoxiao, Lee, Honglak, Lewis, Richard L, & Singh, Satinder. 2015. Action-conditional video prediction using deep networks in atari games. *Pages 2863–2871 of: Advances in neural information processing systems*.

Oord, Aaron van den, Kalchbrenner, Nal, & Kavukcuoglu, Koray. 2016a. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*.

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

Pang, Bo, & Lee, Lillian. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *In: ACL*.

Papamakarios, George, Nalisnick, Eric, Rezende, Danilo Jimenez, Mohamed, Shakir, & Lakshminarayanan, Balaji. 2019. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*.

Paszke, Adam, Gross, Sam, Chintala, Soumith, Chanan, Gregory, Yang, Edward, DeVito, Zachary, Lin, Zeming, Desmaison, Alban, Antiga, Luca, & Lerer, Adam. 2017. Automatic differentiation in pytorch.

Paszke, Adam, Gross, Sam, Massa, Francisco, Lerer, Adam, Bradbury, James, Chanan, Gregory, Killeen, Trevor, Lin, Zeming, Gimelshein, Natalia, Antiga, Luca, Desmaison, Alban, Kopf, Andreas, Yang, Edward, DeVito, Zachary, Raison, Martin, Tejani, Alykhan, Chilamkurthy, Sasank, Steiner, Benoit, Fang, Lu, Bai, Junjie, & Chintala, Soumith. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Pages 8024–8035 of:* Wallach, H., Larochelle, H., Beygelzimer, A., dAlche Buc, F., Fox, E., & Garnett, R. (eds), *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc.

Patil, Ramesh S., Peter Szolovits, & Schwartz, William B. 1982. Modeling knowledge of the patient in acid-base and electrolyte disorders. *Pages 345—348 of: Artificial Intelligence in Medicine.*

Pearl, Judea. 1998. Graphical models for probabilistic and causal reasoning. *Pages 367–389 of: Quantified representation of uncertainty and imprecision.* Springer.

Pearl, Judea. 2009. *Causality.* Cambridge university press.

Pearl, Judea. 2012. *The causal foundations of structural equation modeling.* Tech. rept. University of California, Los Angeles, Department of Computer Science.

Pearl, Judea, *et al.* . 2009. Causal inference in statistics: An overview. *Statistics surveys*, **3**, 96–146.

Pennington, Jeffrey, Socher, Richard, & Manning, Christopher D. 2014. Glove: Global Vectors for Word Representation. *In: EMNLP.*

Raiko, Tapani, & Tornio, Matti. 2009. Variational Bayesian learning of nonlinear hidden state-space models for model predictive control. *Neurocomputing*, **72**(16-18), 3704–3712.

Raiko, Tapani, Tornio, Matti, Honkela, Antti, & Karhunen, Juha. 2006. State inference in variational Bayesian nonlinear state-space models. *Pages 222–229 of: International Conference on Independent Component Analysis and Signal Separation.* Springer.

Ranganath, Rajesh, Gerrish, Sean, & Blei, David M. 2013. Black box variational inference. *arXiv preprint arXiv:1401.0118.*

Ranganath, Rajesh, Perotte, Adler J, Elhadad, Noémie, & Blei, David M. 2015. The Survival Filter: Joint Survival Analysis with a Latent Time Series. *In: Uncertainty in Artificial Intelligence.*

Ravi, Sachin, & Larochelle, Hugo. 2016. Optimization as a model for few-shot learning. *In: ICLR.*

Razavian, Narges, Blecker, Saul, Schmidt, Ann Marie, Smith-McLallen, Aaron, Nigam, Somesh, & Sontag, David. 2015. Population-level prediction of type 2 diabetes from claims data and analysis of risk factors. *Big Data*, **3**(4), 277–287.

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

Rezende, Danilo Jimenez, Mohamed, Shakir, & Wierstra, Daan. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082.*

Rubin, Donald B. 1974. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology*, **66**(5), 688.

Rudolph, Maja R, Ruiz, Francisco JR, Mandt, Stephan, & Blei, David M. 2016. Exponential Family Embeddings. *In: NIPS.*

Sachan, Devendra Singh, Xie, Pengtao, Sachan, Mrinmaya, & Xing, Eric P. 2017. Effective use of bidirectional language modeling for transfer learning in biomedical named entity recognition. *arXiv preprint arXiv:1711.07908.*

Salakhutdinov, Ruslan, & Larochelle, Hugo. 2010. Efficient learning of deep Boltzmann machines. *Pages 693–700 of: Proceedings of the thirteenth international conference on artificial intelligence and statistics.*

Salimans, Tim, Kingma, Diederik, & Welling, Max. 2015. Markov chain monte carlo and variational inference: Bridging the gap. *Pages 1218–1226 of: International Conference on Machine Learning.*

Salimans, Tim, Karpathy, Andrej, Chen, Xi, & Kingma, Diederik P. 2017. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517.*

Scholz, Jonathan, Levihn, Martin, Isbell, Charles, & Wingate, David. 2014. A physics-based model prior for object-oriented mdps. *Pages 1089–1097 of: International Conference on Machine Learning.*

Schön, Thomas B, Wills, Adrian, & Ninness, Brett. 2011. System identification of nonlinear state-space models. *Automatica*, **47**(1), 39–49.

Schulam, Peter, & Saria, Suchi. 2016. Integrative analysis using coupled latent variable models for individualizing prognoses. *The Journal of Machine Learning Research*, **17**(1), 8244–8278.

Schulam, Peter, & Saria, Suchi. 2017. Reliable decision support using counterfactual models. *Pages 1697–1708 of: Advances in Neural Information Processing Systems.*

Schwab, Patrick, Linhardt, Lorenz, Bauer, Stefan, Buhmann, Joachim M, & Karlen, Walter. 2019. Learning counterfactual representations for estimating individual dose-response curves. *arXiv preprint arXiv:1902.00981.*

Sedhain, Suvash, Menon, Aditya Krishna, Sanner, Scott, & Braziunas, Darius. 2016. On the effectiveness of linear models for one-class collaborative filtering. *In: Thirtieth AAAI Conference on Artificial Intelligence.*

Shachter, Ross D. 2013. Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). *arXiv preprint arXiv:1301.7412*.

Shi, Yuge, Siddharth, N, Paige, Brooks, & Torr, Philip. 2019. Variational mixture-of-experts autoencoders for multi-modal deep generative models. *Pages 15718–15729 of: Advances in Neural Information Processing Systems*.

Shivade, Chaitanya, Raghavan, Preethi, Fosler-Lussier, Eric, Embi, Peter J, Elhadad, Noemie, Johnson, Stephen B, & Lai, Albert M. 2013. A review of approaches to identifying patient phenotype cohorts using electronic health records. *Journal of the American Medical Informatics Association*, **21**(2), 221–230.

Siddharth, Narayanaswamy, Paige, Brooks, Van de Meent, Jan-Willem, Desmaison, Alban, Goodman, Noah, Kohli, Pushmeet, Wood, Frank, & Torr, Philip. 2017. Learning disentangled representations with semi-supervised deep generative models. *Pages 5925–5935 of: Advances in Neural Information Processing Systems*.

Silva, Ricardo. 2016. Observational-interventional priors for dose-response learning. *Pages 1561–1569 of: Advances in Neural Information Processing Systems*.

Silver, David, Huang, Aja, Maddison, Chris J, Guez, Arthur, Sifre, Laurent, Van Den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, *et al.* . 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, **529**(7587), 484.

Slee, Vergil N. 1978. The International classification of diseases: ninth revision (ICD-9). *Annals of Internal Medicine*.

Smolenski, Derek J, Pruitt, Larry D, Vuletic, Simona, Luxton, David D, & Gahm, Gregory. 2017. Unobserved heterogeneity in response to treatment for depression through videoconference. *Psychiatric rehabilitation journal*, **40**(3), 303.

Snell, Jake, Swersky, Kevin, & Zemel, Richard. 2017. Prototypical networks for few-shot learning. *In: NIPS*.

Socher, Richard, Perelygin, Alex, Wu, Jean Y, Chuang, Jason, Manning, Christopher D, Ng, Andrew Y, Potts, Christopher, *et al.* . 2013. Recursive deep models for semantic compositionality over a sentiment treebank. *In: EMNLP*.

Sohn, Kihyuk, Lee, Honglak, & Yan, Xinchen. 2015. Learning structured output representation using deep conditional generative models. *Pages 3483–3491 of: Advances in neural information processing systems*.

Soleimani, Hossein, Subbaswamy, Adarsh, & Saria, Suchi. 2017. Treatment-response models for counterfactual reasoning with continuous-time, continuous-valued interventions. *In: 33rd Conference on Uncertainty in Artificial Intelligence, UAI 2017*. AUAI Press Corvallis.

Sønderby, Casper Kaae, Raiko, Tapani, Maaløe, Lars, Sønderby, Søren Kaae, & Winther, Ole. 2016a. How to train deep variational autoencoders and probabilistic ladder networks. *In: 33rd International Conference on Machine Learning (ICML 2016).*

Sønderby, Casper Kaae, Raiko, Tapani, Maaløe, Lars, Sønderby, Søren Kaae, & Winther, Ole. 2016b. Ladder variational autoencoders. *Pages 3738–3746 of: Advances in neural information processing systems.*

Spearman, Charles. 1904a. "General Intelligence," objectively determined and measured. *The American Journal of Psychology.*

Spearman, Charles. 1904b. "General Intelligence" Objectively Determined and Measured. *American Journal of Psychology*, **15**(2), 201–293.

Sukkar, Rafid, Katz, Elyse, Zhang, Yanwei, Raunig, David, & Wyman, Bradley T. 2012. Disease progression modeling using hidden Markov models. *Pages 2845–2848 of: Annual International Conference of the IEEE Engineering in Medicine and Biology Society.* IEEE.

Szolovits, Peter. 1982. Artificial intelligence methods for medical expert systems. *In: In Proc. Amer. Med. Informatics Assn. Congress . AMIA.*

Szolovits, Peter. 1986. Knowledge-based systems: A survey. *Pages 339–352 of: On Knowledge Base Management Systems.* Springer.

Szolovits, Peter, Patil, Ramesh S, & Schwartz, William B. 1988. Artificial intelligence in medical diagnosis. *Annals of internal medicine*, **108**(1), 80–87.

Team, The Theano Development, Al-Rfou, Rami, Alain, Guillaume, Almahairi, Amjad, Angermueller, Christof, Bahdanau, Dzmitry, Ballas, Nicolas, Bastien, Frédéric, Bayer, Justin, Belikov, Anatoly, *et al.* . 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688.*

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints.*

Tipping, Michael E, & Bishop, Christopher M. 1999. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **61**(3), 611–622.

Toyer, Sam, Cherian, Anoop, Han, Tengda, & Gould, Stephen. Human pose forecasting via deep markov models. *Pages 1–8 of: 2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA).* IEEE.

Tran, Dustin, Ranganath, Rajesh, & Blei, David. 2016. The variational Gaussian process. *In: International Conference on Representation Learning.*

Valpola, Harri, & Karhunen, Juha. 2002. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural computation*, **14**(11), 2647–2692.

Vapnik, Vladimir, & Vashist, Akshay. 2009. A new learning paradigm: Learning using privileged information. *Neural networks*, **22**(5-6), 544–557.

Vincent, Pascal, Larochelle, Hugo, Bengio, Yoshua, & Manzagol, Pierre-Antoine. 2008. Extracting and composing robust features with denoising autoencoders. *Pages 1096–1103 of: Proceedings of the 25th international conference on Machine learning.*

Vinyals, Oriol, Blundell, Charles, Lillicrap, Tim, Wierstra, Daan, *et al.* . 2016. Matching networks for one shot learning. *Pages 3630–3638 of: Advances in Neural Information Processing Systems.*

Wan, Eric A, & Nelson, Alex T. 1997. Dual Kalman filtering methods for nonlinear prediction, smoothing and estimation. *Pages 793–799 of: Advances in neural information processing systems.*

Wan, Eric A, & Van Der Merwe, Rudolph. 2000. The unscented Kalman filter for nonlinear estimation. *Pages 153–158 of: Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000.* IEEE.

Wang, Bo, Mezlini, Aziz M, Demir, Feyyaz, Fiume, Marc, Tu, Zhuowen, Brudno, Michael, Haibe-Kains, Benjamin, & Goldenberg, Anna. 2014a. Similarity network fusion for aggregating data types on a genomic scale. *Nature methods*, **11**(3), 333.

Wang, Shangfei, Chen, Shiyu, Chen, Tanfang, & Shi, Xiaoxiao. 2018. Learning with privileged information for multi-label classification. *Pattern Recognition*, **81**, 60–70.

Wang, Shengjie, Mohamed, Abdel-rahman, Caruana, Rich, Bilmes, Jeff, Plilipose, Matthai, Richardson, Matthew, Geras, Krzysztof, Urban, Gregor, & Aslan, Ozlem. 2016. Analysis of deep neural networks with extended data jacobian matrix. *Pages 718–726 of: International Conference on Machine Learning.*

Wang, Xiang, Sontag, David, & Wang, Fei. 2014b. Unsupervised learning of disease progression models. *Pages 85–94 of: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining.*

Warner, JH, & Sampaio, C. 2016. Modeling Variability in the Progression of Huntington's Disease A Novel Modeling Approach Applied to Structural Imaging Markers from TRACK-HD. *CPT: pharmacometrics & systems pharmacology*, **5**(8), 437–445.

Wasserman, Larry. 2013. *All of statistics: a concise course in statistical inference.* Springer Science & Business Media.

Watter, Manuel, Springenberg, Jost, Boedecker, Joschka, & Riedmiller, Martin. 2015. Embed to control: A locally linear latent dynamics model for control from raw images. *Pages 2746–2754 of: Advances in neural information processing systems.*

Webb, Stefan, Golinski, Adam, Zinkov, Rob, Siddharth, N, Rainforth, Tom, Teh, Yee Whye, & Wood, Frank. 2018. Faithful inversion of generative models for effective amortized inference. *Pages 3070–3080 of: Advances in Neural Information Processing Systems.*

West, Jeffrey, & Newton, Paul K. 2017. Chemotherapeutic dose scheduling based on tumor growth rates provides a case for low-dose metronomic high-entropy therapies. *Cancer research*, **77**(23), 6717–6728.

Wu, Lang, Liu, Wei, Yi, Grace Y, & Huang, Yangxin. 2012. Analysis of longitudinal and survival data: joint modeling, inference methods, and issues. *Journal of Probability and Statistics*, **2012**.

Wu, Mike, & Goodman, Noah. 2018. Multimodal generative models for scalable weakly-supervised learning. *Pages 5575–5585 of: Advances in Neural Information Processing Systems.*

Wu, Yao, DuBois, Christopher, Zheng, Alice X, & Ester, Martin. 2016. Collaborative denoising auto-encoders for top-n recommender systems. *Pages 153–162 of: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining.*

Xu, Yanbo, Xu, Yanxun, & Saria, Suchi. 2016. A Bayesian nonparametric approach for estimating individualized treatment-response curves. *Pages 282–300 of: Machine Learning for Healthcare Conference.*

Yala, Adam, Schuster, Tal, Miles, Randy, Barzilay, Regina, & Lehman, Constance. 2019. A deep learning model to triage screening mammograms: a simulation study. *Radiology*, **293**(1), 38–46.

Yu, Chun-Nam John, & Joachims, Thorsten. 2009. Learning structural svms with latent variables. *In: ICML.* ACM.

Zaheer, Manzil, Kottur, Satwik, Ravanbakhsh, Siamak, Poczos, Barnabas, Salakhutdinov, Ruslan R, & Smola, Alexander J. 2017. Deep sets. *Pages 3391–3401 of: Advances in Neural Information Processing Systems.*

Zhi-Xuan, Tan, Soh, Harold, & Ong, Desmond C. 2020. Factorized inference in Deep Markov Models for incomplete multimodal time series. *In: Association for the Advancement of Artificial Intelligence.*

Zhu, Jun, & Xing, Eric P. 2009. Maximum entropy discrimination Markov networks. *Journal of Machine Learning Research*, **10**(Nov), 2531–2569.