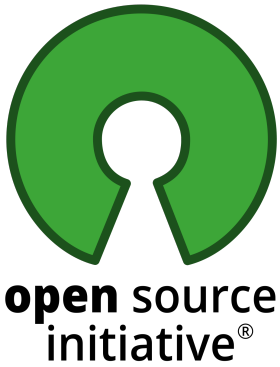
Probabilistic Modeling of Structure in Science: Statistical Physics to Recommender Systems

Jaan Altosaar

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of Physics
Advisors: David Blei and Shivaji Sondhi

June 2020



- This document: https://github.com/altosaar/thesis
- Chapter 3: https://github.com/altosaar/hierarchical-variational-models-physics
- Chapter 4: https://github.com/altosaar/rankfromsets
- Chapter 5: https://github.com/altosaar/proximity_vi

Abstract

Applied machine learning relies on translating the structure of a problem into a computational model. This arises in applications as diverse as statistical physics and food recommender systems. The pattern of connectivity in an undirected graphical model or the fact that datapoints in food recommendation are unordered collections of features can inform the structure of a model. First, consider undirected graphical models from statistical physics like the ubiquitous Ising model. Basic research in physics requires scalable simulations for comparing the behavior of a model to its experimental counterpart. The Ising model consists of binary random variables with local connectivity; interactions between neighboring nodes can lead to long-range correlations. Modeling these correlations is necessary to capture physical phenomena such as phase transitions. To mirror the local structure of these models, we use flow-based convolutional generative models that can capture long-range correlations. Combining flow-based models designed for continuous variables with recent work on hierarchical variational approximations enables the modeling of discrete random variables. Compared to existing variational inference methods, this approach scales to statistical physics models with millions of correlated random variables and uses 100 times fewer parameters. Just as computational choices can be made by considering the structure of an undirected graphical model, model construction itself can be guided by the structure of individual datapoints. Consider a recommendation task where datapoints consist of unordered sets, and the objective is to maximize top-K recall, a common recommendation metric. Simple results show that a classifier with zero worst-case error achieves maximum top-K recall. Further, the unordered structure of the data suggests the use of a permutation-invariant classifier for statistical and computational efficiency. We evaluate such a classifier on human dietary behavior data, where every meal is an unordered collection of ingredients, and find that it outperforms probabilistic matrix factorization methods. Finally, we show that building problem structure into an approximate inference algorithm improves the accuracy of probabilistic modeling methods.

Acknowledgements

HUNDREDS of people have contributed to this thesis expedition. I am grateful for Acknowledgements, to reflect on the village it takes to train a Ph.D.

Shivaji Sondhi—thank you for your enduring support. Shivaji first answered my sophomoric email when I was searching for a summer internship in college, and years later took me under his wing after two advisers moved their labs out of Princeton during my first year of graduate school.

To David Blei, thank you for taking a chance on me. In our first research meeting I asked Dave what the word prior means, that I had been hearing it in the lab's reading group for the past few months. I wondered out loud whether it might be important for me to learn about. Without missing a beat Dave suggested I pick up Bishop's textbook. The same spirit of non-judgmental openness and freedom has enabled me to carve my own path through graduate school, hewn from Dave's support as a role model in thinking and writing clearly.

To Rajesh Ranganath: thank you for picking me up by the nape and pointing me in the right direction innumerable times and at personal cost; for believing in me when I did not, and gently showing me that I stand a chance despite the odds. As but one example, Rajesh had an unimaginably compassionate response to my errors when writing our first paper together, sleep-deprived. He interpreted my blunders as mere confounders in lieu of ignorance, and accurately inferred an underlying serious health condition. The subsequent sleep apnea diagnosis at a young age changed my outlook on life and research. I could at last retain and understand information without rote memorization, and my systolic blood pressure dropped a solid twenty. In the few counterfactuals where I redo grad school with enthusiastic consent, I get to re-meet Rajesh in them all—he fills the void of unambiguously altruistic, heroic characters in our lives. Thank you for modeling thinking of and doing for others (especially when they are not around), for indefatigable craic and curiosity under duress; for boldly asking why.

Thank you to my Columbia family: Aaron Schein, Keyon Vafa, Scott Linderman, Kriste Krstovski, Stephan Mandt, Stella Lianou, Kui Tang, Robert Florida, Dustin Tran, James McInerney, Wesley Tansey, Jackson Loper, Daisy Nguyen, Dawen Liang, Alp Kucukelbir, Victor Veitch, Liping Liu, Thibault Sellam, Adji Dieng, Ali Mehmani, Ghada Almashaqbeh, Yunhao Tang, Yixin Wang, Giannis Karamanolakis, Oscar Chang, Olivia Winn, Francisco J. Rodríguez Ruiz, Da Tang, Iñigo Urteaga, Ghazal Fazelnia, Christian Naesseth, Drim Stokhuijzen, and Gonzalo Esteban Mena.

To Kyle Cranmer, your generous enthusiasm and appetite for science is unmatched, and your warm welcome made the New York University stop in my thesis journey enriching and productive. To other colleagues and friends at

NYU, thank you for the impromptu hangouts, whiteboards, and celebrations: I acknowledge Will Whitney, Nikita Nangia, Zach Martin, Raghav Singal, Michael Albergo, Min Jae Song, Mukund Sudarshan, David Brandfonbrener, Alfredo Canziani, Cinjon Resnick, Xintian Han, Mark Goldstein, Aahlad Manas Puli, and Ilya Kostrikov.

At Google and DeepMind, I am grateful to Eugene Brevdo and Andriy Mnih for mentoring me during unforgettable internships. Any hint of reliable experimentation and code used for this thesis is a direct reflection of what you taught me. And in the South Bay during my time at Google, I am grateful to Tiiu Jürvetson for being the most graceful, generous roommate a lowly intern could ask for.

Thank you to my collaborators Kexin Huang, Jingwei Zhang, Andrew James Mercer-Taylor, Ethan Benjamin, Rohan Bansal, Tony Paek, Anjishnu Kumar, Abhishek Bhatia, Drishan Arora, Shixiang Shane Gu, and Eamon Bell.

Cristina Alvarez, Christina DeLa Cruz, Sol and José at Big Sister Cleaners, Julio and Raoul at Rapid Park Industries, Robert Rafulowitz, Aimée Zambrana, Rayfield Gibson, Sharnice Ottley, and José Luis Ortiz: you were often the only (joyous!) human contact on many long days entering and exiting otherwise nameless buildings and streets.

Through Princeton I am grateful to have met many kindred spirits, including Jamal Williams, Sam Ritter, Gecia Bravo Hermsdorff, Lee Gunderson, Maxim Zaslavsky, Tom Clayton, Kevin Crowley, Siddharth Mishra-Sharma, Andrew Hein, Adam Pellegrini, Tim Treuer, Mark Ioffe, Vanessa Lehner, Richard Da, Kelvin Mei, DJ Strouse, Vladimir Kirilin, Angela Radulescu, Jon Berliner, Andrew Hartnett, Allison Chaney, Zack Dulberg, Alan Morningstar, and Tejal Bhamre. Our long walks around campus and the world are like fireworks; I am grateful for your stamina for conversating, gregarious rabbit holes, and your readiness to disagree for hours about weakly-held opinions.

To Princeton staff Josko Plazonic, Bill Wichser, Kate Brosowsky, Barbara Mooring, Jessica Heslin, Mariangela Lisanti, Jim Olsen, Bob Austin, Herman Verlinde—you help set a high-water mark for a supportive, collegial work environment. Any staff was willing to help at the slightest instigation, even when I was scared to ask. It makes a difference.

For making New York City feel like home I am indebted to Toby Shorin, Alessandra Poblador, Bharat Srikishan, Elisa Muyl, Bernard Itaka Lebrón, Jeffrey Seeley, Amy McDermott, Genevieve Fried, Luc Cary, Marina Krygin, Andra Mihali, Kariina Altosaar, Jessica Forde, Will Geary, Louise Contino, Alexandra Rone Lang, Tessa Bonduelle, Laurent Charlin, Clarissa Moliterno, Hiie Saumaa, Maja Rudolph, Gustav Kalm, James Shen, Sarth Calhoun, Ruth Tupe, Sophie Kleber, Becca Dittrich, and the forró family.

To Hanif Jetha, Kyle Saikaley, Maryse Thomas, Bohdan Kulchytskyy, Michael King, Fredrik Folkestad, Briana Salas, Francis Megna, Colin Raffel, Sarah Hagi, Christine Li, Ben Poole, Patrick Murumi Njoroge, Raul Altosaar, Laura Cooper Hall, David Dohan, Jasmine Collins, Dylan Cable, David Aasen, Jakob Foerster, Sergey Bartunov, Tyeler Matsuo, Éva Altosaar, Natasha Jaques, Vanessa Chazelle, Faris Haddad, Aleksi Altosaar, Courtney Yusuf, Rebecca Choong-Wilkins, Nick Greaves-Tunnell, Janine, Peter, and Justin Cary, Georgie Rubens, Solomon Sonya, and Cathy Ji—thank you for reminding me from near and far about what is important in life.

To my parents: Illimar, for always picking up the phone and infectious enthusiasm for the world; Tiiu, for assuaging rational and irrational panics in ways only a mother can.

Finally, I recognize Caroline Martin: if this is but a crisis with you, I would rejoice in submitting to considerably greater mid-pandemic quarantines, job hunts, and theses.

Contents

Chapter 1

Introduction

From the development of novel antibiotics (Stokes et al., 2020) to cataloging sources of light in the night sky (Regier et al., 2019), many domains in science can benefit from applied machine learning methods. However, the utility of such methods hinges on building the structure of a problem—knowledge about the data or task—into a machine learning solution. Whether the setting is statistical physics or recommender systems, an off-the-shelf machine learning method can serve as a starting point. But performance is sacrificed when a method cannot be customized to the specifics of an applied scientific problem. This thesis focuses on probabilistic modeling, where what is known about a problem can be molded into assumptions about a probability distribution. We develop probabilistic modeling methods that use the structure of a problem to yield meaningful solutions in the study of models with large numbers of random variables in statistical physics systems and recommender systems. In tandem, this thesis develops an algorithm to improve the accuracy of approximations to probabilistic models, through the use of the structure of a probability model during optimization. By both building scalable probabilistic modeling methods tailored to answer scientific questions, and improving flexible probabilistic modeling methods themselves, we highlight the reciprocal relationship between these aims.

One example of an applied problem in statistical physics is the study of probable configurations of atoms in a material. Simulating a material to find likely configurations of atoms with statistical physics models can be expensive, but designing materials with improved properties is valuable (Schmidt et al., 2019). The computational cost of these simulations for studying statistical physics models can be reduced by doing math, for example in analytical calculations to develop approximations or to incorporate knowledge of how neighboring atoms interact into simulations (Swendsen and Wang, 1987). A challenge in studying statistical physics models is balancing problem-specific customization with the result-

ing computational savings. Machine learning techniques applied to statistical physics systems can be used to develop generic methods that exploit problem structure for better performance. These methods can be re-used across models, saving practitioners time.

Where statistical physics concerns probable configurations of interacting atoms, recommender systems find items a user is likely to interact with (Koren et al., 2009). For example, humans eat. A meal recommender system can predict which meals someone is likely to consume. Such a recommendation model might inform its predictions using the history of meals a user has eaten, namely which foods comprise those meals. A property of this type of data is that items (meals) are associated with unordered collections of attributes (sets of foods). This means that the number of possible meals a user might consume is very large. Existing methods for this type of data either cannot scale to large numbers of datapoints, or fail to accurately predict which items a user is likely to consume. This highlights the need to imbue a recommendation model with both properties of the data (such as meals represented as unordered sets) and the goals of the recommendation problem, or accurate prediction of which items a user will consume.

Both statistical physics models and recommender systems can be framed as probability models, the former as probable configurations of atoms, and the latter as probable items users may consume. Probabilistic modeling relies on inferring the parameters of a probability model using knowledge about the structure of the problem. For example, knowledge in the form of data regarding which meals someone has eaten can inform the predictions of a recommendation model; or, knowledge of how neighboring atoms interact in a material can be used in a probabilistic model of that material.

Practitioners that work with probability models seek probabilistic inferences. For example, the goals of such inferences include computing probabilities, summing over the random variables in a probability model, or finding likely configurations of random variables. Common inference methods are Markov Chain Monte Carlo (Metropolis et al., 1953), variational inference (Blei et al., 2017), and maximum likelihood estimation (Bishop, 2006). This thesis uses variational inference and maximum likelihood estimation, as both algorithms can be scaled to large probability models (Hoffman et al., 2013; Robbins and Monro, 1951). In particular, the variational inference algorithm can aid probabilistic inference in interacting systems of random variables found in statistical physics models. However, variational inference is sensitive to the initial choice of parameters governing the probability of the random variables under study. The accuracy of inferences of likely configurations of random variables can suffer, depending on this choice of initial parameters. Inference algorithms such as variational inference are utile in applied domains insofar as their performance is independent of the initial choice of parameters.

This thesis is organized as follows. Chapter 2 introduces probabilistic models and gives examples of their use in statistical physics and recommender systems. We also review two approaches for statistical inference in probability models: variational inference and maximum likelihood estimation. Chapter 3 develops and applies variational inference methods for statistical physics models. We show that exploiting the structure of a statistical physics model in a variational inference method is advantageous. This work was presented in Altosaar et al. (2019). Chapter 4 develops probability models for recommending items with sets of attributes and is based on Altosaar et al. (2020). Similar to variational methods in physics, accounting for the structure of the problem helps: probability models that represent set-valued datapoints and the goals of the recommendation task are accurate and scalable. Chapter 5 develops proximity variational inference (PVI) based on Altosaar et al. (2018). PVI is an inference algorithm that is imbued with information about a probability distribution we wish to infer. In this case, the structure of the problem is information about a probability distribution, which is used to inform an algorithm for fitting a probability model. We show how this enables PVI to obtain accurate solutions. Finally, Chapter 6 reviews how knowledge about a problem is useful in building probabilistic models for science solutions. We apply the PVI algorithm developed in Chapter 5 to the statistical physics setting of Chapter 3 and the recommender systems application of Chapter 4. This highlights that methods development goes hand-in-hand with the aims of applied probabilistic modeling. We close with a discussion of extensions of this line of work.

Chapter 2

Background

This chapter describes probabilistic models and probabilistic inference, taking as examples models from statistical physics and recommender systems.

2.1 Probabilistic Models

Probability models assign probability to configurations of random variables. The random variables in a probability model might correspond to observed variables in a physical system, or to latent properties representing patterns in data collected from the world, or a combination of both. To define a probability model, it is necessary to specify the density p of a collection of random variables $\mathbf{z}$. We focus on probabilistic models $p(\mathbf{z})$ where relationships between random variables can be encoded as edges in a graph, or probabilistic graphical models (Jordan, 2004).

2.1.1 Example: Ising Model

For example, consider a model used in statistical physics: the Ising model. The Ising model can be used to model interactions between atoms in a material (Henelius et al., 2016) to study how the material behaves in different conditions, paving the way toward material design. This probabilistic model has binary random variables z_n with density

$$p(\mathbf{z}; \beta) = \frac{\exp(-\beta E(\mathbf{z}))}{\mathcal{Z}}. \tag{2.1}$$

The semicolon in Equation (2.1) denotes that the model has a parameter β, representing the reciprocal temperature of the system of random variables (a physical

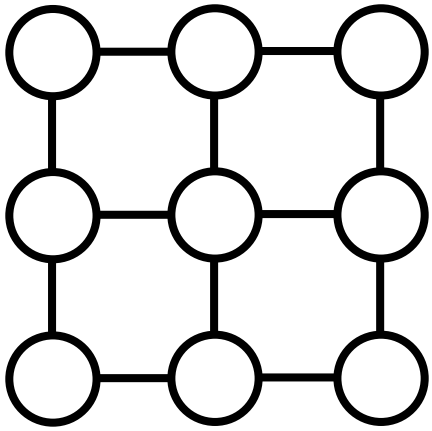


Figure 2.1: The Ising model is a probabilistic model used in statistical physics. The nodes in this probabilistic graphical model represent random variables, and the edges in the graph represent relationships between neighboring random variables. In this Ising model there are nine random variables variables $\mathbf{z} = \{z_1, z_2, \ldots, z_9\}$ represented by nodes and the edges connecting two nodes indicate that those random variables interact in the energy function of the model $E(\mathbf{z})$.

quantity). The energy function $E(\mathbf{z})$ encodes the relationships between random variables, and $\mathcal{Z}$, the normalizing constant, ensures that this probability distribution sums to one over all configurations of random variables (Chandler and Wu, 1987). The energy function of the Ising model is[1]

$$E(\mathbf{z}) = -\frac{1}{2}\sum_{i,j} J_{ij} z_i z_j - H \sum_i z_i \,. \tag{2.2}$$

The interaction strength J_{ij} defines the interactions between random variables. In a simple Ising model, only nearest neighbors interact, so J_{ij} is nonzero if the random variables z_i and z_j are neighbors. The parameter H increases or decreases the energy in proportion to the values of the random variables z_i; we give its physical interpretation later.

The Ising model can be represented as a probabilistic graphical model, shown in Figure 2.1. Two variables z_i and z_j interact (changing the value of one leads to a change in probability of the other) only if they share an edge in the graph. This representation works in conjuction with the density in Equation (2.1), as the presence of an edge in the graph corresponds to two variables interacting in the energy function E. In this model, the energy function (and hence graph) is such that only neighboring random variables interact.

[1]Bold letters can denote collections of random variables $\mathbf{z} = \{z_1, z_2, ..., z_N\}$, or vectors, depending on the context.

The Ising model can be used to study physical systems such as magnetic materials, where interactions between atoms can be encoded into the interaction strength J_{ij}. The interactions between random variables encoded in this manner contain the necessary information to model the properties of a material. In modeling a material, the random variables $\mathbf{z}$ can be referred to as spins. Spin is a type of angular momentum carried by particles comprising atoms, and such angular momentum causes a magnetic field. Although the random variables $\mathbf{z}$ are binary, taking on values of -1 and $+1$, they can be re-scaled to the magnetic strength of the atoms in a particular material of interest if comparison to experimental data is required. The parameter H can be interpreted as the magnitude of an external magnetic field that interacts with the magnetic strength and orientation of every atom (Chandler and Wu, 1987).

To see how well an Ising model mirrors a physical material, a property such as magnetization can be measured in the material, and calculated using the model. Magnetization is the average orientation of the magnetic strength of every atom or random variable in the material,

$$M(\mathbf{z}) = \frac{1}{N} \sum_{i=1}^{N} z_i \,. \tag{2.3}$$

By measuring the magnetization M and computing its value in the Ising model, a practitioner can deduce how accurately the model reproduces experimental data. For example, if an Ising model with nearest neighbors ($J_{ij} \neq 0$ if i neighbors j) does not accurately reproduce the magnetization of a physical material, it may be necessary to include second-nearest neighbor effects ($J_{ij} \neq 0$ if i and j are connected by a path of length at most two).

Another example of a quantity that can be measured experimentally and computed in a probabilistic model is the thermodynamic free energy F,

$$F = -\frac{1}{\beta} \log \mathcal{Z} \,. \tag{2.4}$$

The free energy of a system relates to the amount of energy that can be extracted from a system by its surroundings. For example, the free energy of a protein is used to understand its stability, and can be measured by the amount of energy needed to destroy its structure by denaturing it (Stone, 2013). In modeling a magnetic material or biological material, the free energy can be derived from the normalizing constant $\mathcal{Z}$ (Chandler and Wu, 1987).

	Attributes									User
Items	Pizza	Eggs	Taco	Salad	Avocado	Chicken	Sardines	Beer	Coffee	1
Morning Pizza	•	•							•	•
Dinner Pizza	•						•	•		
Small Salad				•	•		•			
Big Salad		•		•	•	•			•	•
Taco			•		•	•			•	
Fish Taco			•				•			

Table 2.1: Example binary classification data. A user consumes meals (rightmost column), and meals have attributes (table on the left. Meals are represented as datapoints x_n with covariates being foods in the meals. The goal of a binary classifier trained on this data is to predict which meals a user will consume, or which datapoints (x_n, y_n) have a label $y_n = 1$. An accurate classifier will information about which covariates are shared across positive or negative labeled datapoints; for example, this user consumes meals that include eggs and coffee.

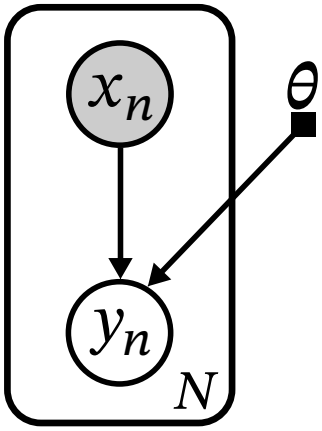


Figure 2.2: Binary classification is a probabilistic modeling technique used in recommender systems. Observed random variables are denoted by shaded nodes, and directed edges in the graph indicated conditional dependence on a node's parents. There are N independent, identically distributed observations (y_n, x_n); the rectangular plate denotes repetition of nodes and edges. The model predicts a binary response y_n using parameters θ and covariates x_n.

2.1.2 Example: Binary Classification

Another example of a probabilistic model is a binary classifier (Bishop, 2006), represented as a graphical model in Figure 2.2. Consider N datapoints of the form (x_n, y_n) consisting of covariates x_n and binary responses y_n. As illustrated in Table 2.1, the covariates x_n might represent information about items such as foods in a meal, and y_n may indicate whether a single user ate a meal with those foods. A binary classifier would then classify whether the user would eat a new meal $\hat{x}_n$ based on its constituent foods.

A binary classifier is defined using a regression function f with parameters θ. The logistic function σ applied to the regression function defines the probability model for a binary classifier,

$$p(y_n \mid x_n; \theta) = \frac{\exp\left(\sigma(f(x_n; \theta)) \cdot y_n\right)}{\mathcal{Z}}. \tag{2.5}$$

The logistic function constrains the output of f to the unit interval, and $\mathcal{Z}$ is again the normalizing constant. The regression function f uses information about a datapoint to classify whether the response y_n is positive. An example of a regression function is an inner product, defined by

$$f(x_n; \theta) = \theta^\top x_n\,, \tag{2.6}$$

which corresponds to logistic regression (Bishop, 2006). Alternatively, a more flexible model can be built using a deep neural network (LeCun et al., 2015).

2.2 Inference

In a probability model, computing—or, inferring—properties of the probability distribution is a central task. One inference problem is to ascertain likely configurations of random variables. Another is to compute the sum of a probability distribution over a set of random variables, for example, to compute the normalizing constant (Jordan, 2004).

2.2.1 Computing Likely Configurations of Random Variables

In the study of a probability model such as a binary classifier in Equation (2.5), one question of interest is: for a set of observations (x_n, y_n), what is a likely value of θ? Maximum likelihood estimation is one way to answer this question (Bishop, 2006).

A probability distribution like $p(\mathbf{y} \mid \mathbf{x}; \theta)$ is also known as a likelihood function. It defines the likelihood of a random variable $\mathbf{y}$ conditional on the value of data $\mathbf{x}$, with the current setting of the parameters θ. The maximum likelihood estimate of the parameters of this probability model for the data $(\mathbf{x}, \mathbf{y})$ is given by

$$\theta^* = \arg\max_{\theta} p(\mathbf{y} \mid \mathbf{x}; \theta)\,. \tag{2.7}$$

This maximum likelihood estimate of the parameters θ^* can be computed using stochastic optimization if the data is large (Robbins and Monro, 1951).

2.2.2 Computing the Normalizing Constant

The second central inference task in probabilistic modeling is summing a probability model over a set of random variables. One example of this is computing the normalizing constant $\mathcal{Z}$. This inference problem requires computing a sum: the normalizing constant ensures a probability distribution sums to 1 over values the random variables can take.

Consider computing the normalizing constant for the binary classifier in Equation (2.5). To compute the normalizing constant $\mathcal{Z}$ for this probability model, we can sum over the binary values the random variable y_n can take,

$$1 = \sum_{y_n \in \{0,1\}} \frac{\exp\left(\sigma(f(x_n; \theta)) \cdot y_n\right)}{\mathcal{Z}} \tag{2.8}$$

$$\Rightarrow \mathcal{Z} = \sum_{y_n \in \{0,1\}} \exp\left(\sigma(f(x_n; \theta) \cdot y_n)\right) \tag{2.9}$$

$$\mathcal{Z} = 1 + \exp\left(\sigma(f(x_n; \theta))\right) . \tag{2.10}$$

Inference of the normalizing constant $\mathcal{Z}$ is straightforward in this probability model. The random variable y_n is binary, so there are only two terms in the sum needed to compute the normalizing constant.

Next, consider computing the normalizing constant or partition function for the Ising model in Equation (2.1). The random variables z_n in this model also take on binary values. The partition function is computed by summing over all the values associated with all random variables in the system, $\mathbf{z} = \{z_1, \ldots, z_N\}$:

$$1 = \sum_{z_1 \in \{-1,+1\}} \ldots \sum_{z_N \in \{-1,+1\}} \frac{\exp(-\beta E(\mathbf{z}))}{\mathcal{Z}} \tag{2.11}$$

$$\Rightarrow \mathcal{Z} = \sum_{z_1 \in \{-1,+1\}} \ldots \sum_{z_N \in \{-1,+1\}} \exp(-\beta E(\mathbf{z})) . \tag{2.12}$$

There are N binary-valued random variables and 2^N terms in the sum required to compute the partition function, so inference in the Ising model is difficult. For Ising models used to study materials, the partition function is intractable to compute for most model sizes practitioners want to study and compare to physical realizations.

One way to address the issue of an intractable partition function is with sampling methods, such as Markov chain Monte Carlo (Metropolis et al., 1953). These algorithms enable inference by simulating likely configurations of random variables. These samples of likely configurations are used to approximate quantities of interest such as the partition function. But, Markov chain Monte Carlo methods are difficult to scale to probabilistic models with large numbers of correlated ran-

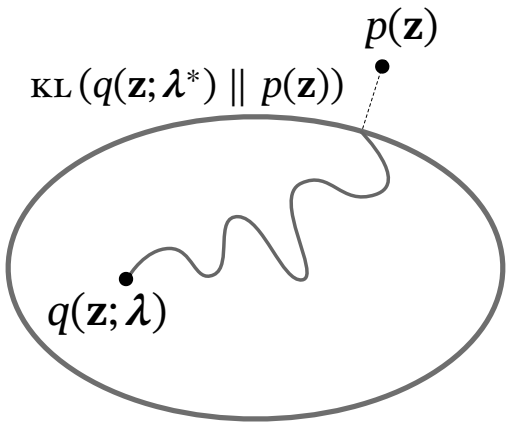


Figure 2.3: Variational inference finds the member of the variational family closest to the target distribution. The oval in the cartoon represents the space of variational approximations $q(\mathbf{z};\boldsymbol{\lambda})$, and the goal of variational inference is to find variational parameters $\boldsymbol{\lambda}^*$ that yield an approximation close to the target probability model $p(\mathbf{z})$. One way to measure the distance between a variational approximation and the target probability distribution is with the Kullback-Leibler (KL) divergence.

dom variables. In this thesis, we instead use variational inference, an approximate inference algorithm that relies on optimization instead of sampling.

2.3 Variational Inference

Instead of working with a probability model $p(\mathbf{z})$ directly, variational inference (VI) posits a family of distributions $q(\mathbf{z};\boldsymbol{\lambda})$ indexed by parameters $\boldsymbol{\lambda}$ (Blei et al., 2017). The goal of variational inference (VI) is to find the closest member of the variational family q to the target distribution p. The algorithm consists of varying the parameters $\boldsymbol{\lambda}$ to improve the quality of the approximation, as illustrated in Figure 2.3. One way to measure the distance between the variational approximation and the target distribution is with the Kullback-Leibler (KL) divergence, or relative entropy (MacKay, 2003; Ranganath, 2018).

The intractable partition function in $p(\mathbf{z})$ appears in the KL divergence VI uses to assess distance,

$$\mathrm{KL}\,(q(\mathbf{z};\boldsymbol{\lambda}) \,||\, p(\mathbf{z})) = \mathbb{E}_q[\log q(\mathbf{z};\boldsymbol{\lambda})] - \mathbb{E}_q[\log p(\mathbf{z})] \tag{2.13}$$

But it is possible to derive an objective function that does not depend on the partition function, starting from the KL divergence. Taking the Ising model in Equa-

tion (2.1) as an example,

$$\mathrm{KL}\left(q(\mathbf{z};\boldsymbol{\lambda}) \,\|\, p(\mathbf{z})\right) = \mathbb{E}_q[\log q(\mathbf{z};\boldsymbol{\lambda})] - \mathbb{E}_q[\log p(\mathbf{z})] \tag{2.14}$$

$$\mathrm{KL}\left(q(\mathbf{z};\boldsymbol{\lambda}) \,\|\, p(\mathbf{z})\right) = \mathbb{E}_q[\log q(\mathbf{z};\boldsymbol{\lambda})] - \mathbb{E}_q[-\beta E(\mathbf{z}) - \log \mathcal{Z}] \tag{2.15}$$

$$\log \mathcal{Z} = \mathbb{E}_q[-\beta E(\mathbf{z})] - \mathbb{E}_q[\log q(\mathbf{z};\boldsymbol{\lambda})] + \mathrm{KL}\left(q(\mathbf{z};\boldsymbol{\lambda}) \,\|\, p(\mathbf{z})\right) \tag{2.16}$$

$$\Rightarrow \log \mathcal{Z} \geq \mathcal{L}(\boldsymbol{\lambda}) \coloneqq \mathbb{E}_q[-\beta E(\mathbf{z})] - \mathbb{E}_q[\log q(\mathbf{z};\boldsymbol{\lambda})]\,. \tag{2.17}$$

This lower bound $\mathcal{L}$ on the log normalizing constant is also called the evidence lower bound (ELBO), and serves as the objective function for VI. In deriving this lower bound from Equation (2.16) to Equation (2.17), we used the fact that the KL is greater than or equal to zero. To show this fact, we start from Jensen's inequality for a convex function f, or

$$f(\mathbb{E}[\mathbf{z}]) \leq \mathbb{E}[f(\mathbf{z})]\,. \tag{2.18}$$

The logarithm in the KL is concave, so its negative is convex. We apply Jensen's inequality to the negative KL in Equation (2.13):

$$-\mathrm{KL}\left(q(\mathbf{z}) \,\|\, p(\mathbf{z})\right) = \mathbb{E}_q\left[\log \frac{p(\mathbf{z})}{q(\mathbf{z})}\right] \tag{2.19}$$

$$\leq \log \mathbb{E}_q\left[\frac{p(\mathbf{z})}{q(\mathbf{z})}\right] \tag{2.20}$$

$$= \log \int q(\mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})} d\mathbf{z} \tag{2.21}$$

$$= \log \int p(\mathbf{z}) d\mathbf{z} \tag{2.22}$$

$$= 0\,. \tag{2.23}$$

This shows that the KL is greater than or equal to zero (Cover and Thomas, 2012).

The left-hand-side in Equation (2.17) does not change as the variational parameters $\boldsymbol{\lambda}$ are varied in $\mathcal{L}(\boldsymbol{\lambda})$. In words, maximizing the lower bound $\mathcal{L}(\boldsymbol{\lambda})$ is equivalent to minimizing the KL divergence between the variational approximation and target probability model.

2.3.1 Example: Mean Field Variational Inference in the Ising model

To demonstrate VI, we use the Ising model described in Section 2.1.1 with probability distribution $p(\mathbf{z})$ defined in Equation (2.1) and energy function $E(\mathbf{z})$ in Equation (2.2). Inspecting the intractable partition function of the Ising model can help construct a variational family $q(\mathbf{z};\boldsymbol{\lambda})$ to approximate the Ising model.

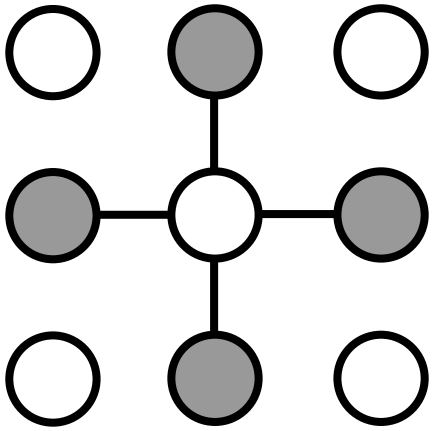


Figure 2.4: The structure of an Ising model can inform variational approximations. This graphical model illustrates a Markov blanket in the Ising model of Figure 2.1. The Markov blanket of a node is the set of nodes whose values need to be fixed to render a node independent of the rest of the graph. In an Ising model, only neighboring random variables interact and therefore comprise the Markov blanket of a node. Here, the Markov blanket of the central node are shaded, indicating that their values are fixed. The missing edges between the peripheral nodes indicate that the central node is independent of the rest of the graph, conditional on its Markov blanket.

The Ising model partition function in Equation (2.12) is intractable because the sums do not decompose by random variables: every sum must be carried out in order, because the result of the Nth sum over the random variable z_N depends on the results of the sums over the previous $N-1$ random variables. This is because of interactions between dependent random variables. The first term in the energy function of the Ising model represents nearest neighbor interactions, $z_i z_j$, and is graphically equivalent to the links between nearest neighbors in Figure 2.1.

However, the second term in the Ising energy function in Equation (2.2), $H \sum_i z_i$, does decompose by random variable. Physically, this corresponds to a magnetic field applied to the system as a whole, so every random variable is subject to the same force. Mathematically, there is an outer sum over every configuration of random variables, and in this term the results of the summation over a variable z_i do not affect the summation over another variable z_j. So this magnetic field term can be evaluated for systems with many random variables.

The structure of the Ising model energy function and corresponding graphical model can be used to build a variational approximation $q(\mathbf{z}; \boldsymbol{\lambda})$ as follows. If the second term of the Ising model energy function does not lead to an intractable partition function due to every random variable being subject to a magnetic field, one can construct a variational approximation by extending this physical intuition and developing the concept of a 'mean field'. Consider the central random

variable z_i in Figure 2.1. Fixing the values of its nearest neighbors renders this random variable independent of the rest of the graph as shown in Figure 3.2. The nearest neighbors of the central random variable can then be interpreted as giving rise to a magnetic field. The strength of this magnetic field is unknown, so we can define this unknown strength as a variational parameter δH that we will infer using VI. This mean field is additive to the external magnetic field H applied to the system as a whole, so the energy function for the central random variable z_i under this mean field assumption can be written

$$E_{\text{MF}}(z_i; \delta H) = \delta H z_i + H z_i \,. \tag{2.24}$$

Note that we have replaced the interaction term $J_{ij} z_i z_j$ in the Ising model energy function in Equation (2.2) by the mean field δH. The mean field assumption is that term can approximate the effects of neighboring nodes (Chandler and Wu, 1987). If we repeat this argument for every node in the graph, we arrive at the mean field energy function

$$E_{\text{MF}}(\mathbf{z}; \delta H) = -(H + \delta H) \sum_{i=1}^{N} z_i \,. \tag{2.25}$$

The above construction starting from the mean field assumption corresponds to the variational approximation with density

$$q(\mathbf{z}; \beta, \delta H) = \prod_{i=1}^{N} \frac{\exp(-\beta E_{\text{MF}}(z_i; \delta H))}{\mathcal{Z}_{\text{MF}}} \,, \tag{2.26}$$

and we see that the variational parameter λ is simply the mean field strength δH. The mean field variational approximation corresponds to a fully factorized probability distribution where every random variable is independent (Wainwright and Jordan, 2008). This is a useful property, as the partition function is tractable in this mean field variational approximation: we can compute the partition function for every random variable by itself. The partition function for a single random variable z_i under the mean field assumption is straightforward,

$$\mathcal{Z}_{\text{MF, i}} = \sum_{z_i \in \{-1, +1\}} \exp(-\beta(H + \delta H) z_i) \tag{2.27}$$

$$= 2 \cosh(\beta(H + \delta H)) \,, \tag{2.28}$$

and the partition function for the variational approximation for all variables is simply $\mathcal{Z}_{\text{MF}} = \mathcal{Z}_{\text{MF, i}}^N$. Similarly, the average of a random variable under the varia-

tional distribution is readily computed as

$$\begin{aligned}\mathbb{E}_{q(z_i)}[z_i] &= \sum_{z_i \in \{-1,+1\}} \frac{z_i \exp(-\beta E_{\text{MF}}(z_i; \delta H))}{\mathcal{Z}_{\text{MF, i}}} \\ &= \sum_{z_i \in \{-1,+1\}} \frac{z_i \exp(-\beta(H + \delta H) z_i)}{2\cosh(\beta(H + \delta H))} \\ &= -\tanh(\beta(H + \delta H))\,.\end{aligned} \tag{2.29}$$

Now that we have constructed a variational family for the Ising model, we can proceed with the VI algorithm. The next step is writing down and maximizing the lower bound on the log partition function to minimize the KL between our approximating distribution and model.

The lower bound on the log partition function $\mathcal{L}(\delta H)$ in Equation (2.17) becomes

$$\mathcal{L}(\delta H) = \mathbb{E}_q[-\beta E(\mathbf{z})] - \mathbb{E}_q[\log q(\mathbf{z}; \delta H)] \tag{2.30}$$

$$= \mathbb{E}_q\left[-\frac{1}{2}\beta \sum_{i,j} J_{ij} z_i z_j - \beta H \sum_i z_i\right] - \mathbb{E}_q\left[-\beta(H + \delta H)\sum z_i\right] + \log \mathcal{Z}_{\text{MF}} \tag{2.31}$$

$$= \mathbb{E}_q\left[-\frac{1}{2}\beta \sum_{i,j} J_{ij} z_i z_j + \beta \delta H \sum_i z_i\right] + \log \mathcal{Z}_{\text{MF}}\,, \tag{2.32}$$

and we can take the expectation inside the sum using the fact that the mean field variational distribution is fully factorized, so

$$\mathcal{L}(\delta H) = -\frac{1}{2}\beta \sum_{i,j} J_{ij} \mathbb{E}_{q(z_i)}[z_i]\mathbb{E}_{q(z_j)}[z_j] + \beta \delta H \sum_i \mathbb{E}_{q(z_i)}[z_i] + \log \mathcal{Z}_{\text{MF}}\,. \tag{2.33}$$

$$\tag{2.34}$$

In the first term, recall that two random variables z_i and z_j have the same distribution under the mean field assumption, and that every variable interacts with its four nearest neighbors in the Ising model. The lower bound on the log partition function then becomes

$$\mathcal{L}(\delta H) = -\frac{1}{2}\beta 4JN \mathbb{E}_{q(z_i)}[z_i]^2 + \beta N \delta H \mathbb{E}_{q(z_i)}[z_i] + \log \mathcal{Z}_{\text{MF}}\,. \tag{2.35}$$

The next step in the VI algorithm is maximizing this lower bound, to minimize the KL divergence between the variational approximation and the model. Taking the derivative with respect to δH and suppressing the subscript of the expecta-

tion operator, we get

$$\frac{\partial \mathcal{L}(\delta H)}{\partial \delta H} = N\beta(-4J\mathbb{E}[z_i]\partial_{\delta H}\mathbb{E}[z_i] + \mathbb{E}[z_i] + \delta H \partial_{\delta H}\mathbb{E}[z_i]) + N\beta \tanh(\beta(H + \delta H)) \,. \tag{2.36}$$

Next, setting this derivative to zero and cancelling out terms (and using Equation (2.29)) leads to

$$\begin{aligned} 0 &= -4J\mathbb{E}[z_i]\partial_{\delta H}\mathbb{E}[z_i] + \delta H \partial_{\delta H}\mathbb{E}[z_i]) && (2.37)\\ \Rightarrow \delta H \partial_{\delta H}\mathbb{E}[z_i]) &= 4J\mathbb{E}[z_i]\partial_{\delta H}\mathbb{E}[z_i] && (2.38)\\ \Rightarrow \delta H^* &= 4J\mathbb{E}[z_i] \,. && (2.39) \end{aligned}$$

This shows that under a mean field assumption, the variational parameter that maximizes the lower bound on the log partition function—and hence minimizes the KL divergence between the approximation and model—is proportional to the mean field around any node in the system. The structure of the model informs our choice of variational approximation.

The quality of the variational approximation $q(\mathbf{z}; \beta, \delta H^*)$ from VI can be assessed in several ways. For example, the magnetization M or the free energy F can be calculated using the variational approximation, and these values can be compared to Markov Chain Monte Carlo simulations in small systems. This can be viewed as a type of predictive check for a VI algorithm (Blei, 2014). However, the development of theoretical guarantees to assess the quality of variational approximations found with VI is an open area of research (Wang and Blei, 2019). Practitioners must currently empirically evaluate the quality of variational approximations according to the task at hand, as we do in Chapters 3 and 5.

2.3.2 Variational Inference Originated in Statistical Physics

Previously, we derived a variational approximation to the Ising model by making a mean field assumption. That the language of physics is used in machine learning algorithms such as VI is no coincidence. In fact, Feynman (1972) and Feynman (2018) derives the Gibbs-Bogoliubov-Feynman (GBF) inequality for use in a variational principle for approximating intractable partition functions using mean field assumptions. Consider a model with energy function E and partition function $\mathcal{Z}$, and a mean field variational approximation with energy function E_{MF} (and corresponding partition function $\mathcal{Z}_{\text{MF}}$). Then the GBF inequality reads (Feynman, 1972; Feynman, 2018)

$$\mathcal{Z} \geq \mathcal{Z}_{\text{MF}} \exp\left(-\beta \left\langle E - E_{\text{MF}} \right\rangle_{\text{MF}}\right) . \tag{2.40}$$

In physics, bra-ket notation is used to denote expectations. For example, expectations with respect to Equation (2.26) are written $\langle \cdot \rangle_{\text{MF}}$. Rewriting the GBF with statistics notation for the expectation $\mathbb{E}_q[\,\cdot\,]$ yields

$$\mathcal{Z} \geq \mathcal{Z}_{\text{MF}} \exp\left(-\beta \mathbb{E}_q[E - E_{\text{MF}}]\right). \tag{2.41}$$

Taking the logarithm, we recover the lower bound on the log partition function

$$\begin{aligned} \log \mathcal{Z} &\geq \mathbb{E}_q[-\beta E] - \mathbb{E}_q[-\beta E_{\text{MF}}] + \log \mathcal{Z}_{\text{MF}} && (2.42)\\ &= \mathbb{E}_q[-\beta E] - \mathbb{E}_q[\log q_{\text{MF}}(\mathbf{z}; \boldsymbol{\lambda})] && (2.43)\\ &= \mathcal{L}(\boldsymbol{\lambda}). && (2.44) \end{aligned}$$

This is identical to the log partition function lower bound in Equation (2.17). Hoffman et al. (2013) review the historical roots of the variational principle in its machine learning incarnation.

To complete the connection to machine learning, we relate this log partition function lower bound to the evidence lower bound studied in the VI literature (Blei et al., 2017). A probabilistic model of data might have the following process for generating data $\mathbf{x}$ using prior information in latent variables $\mathbf{z}$:

$$\begin{aligned} \mathbf{z} &\sim p(\mathbf{z}) \\ \mathbf{x} &\sim p(\mathbf{x} \mid \mathbf{z}) \end{aligned}$$

The posterior distribution of this model is computed using Bayes' rule,

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z})}{p(\mathbf{x})}.$$

The model evidence $p(\mathbf{x})$ is the partition function of the posterior. Calculating the partition function is what makes posterior inference difficult, as it requires integration over the latent variables $\mathbf{z}$,

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z},$$

and the latent variables $\mathbf{z}$ are typically high-dimensional, such as the number of random variables in an Ising model. But VI can be used to approximate this intractable integral. The lower bound on the log partition function becomes the evidence lower bound (ELBO):

$$\begin{aligned} \log p(\mathbf{x}) &\geq \mathcal{L}(\boldsymbol{\lambda}) && (2.45)\\ \mathcal{L}(\boldsymbol{\lambda}) &= \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z})] - \mathbb{E}_q[\log q(\mathbf{z}; \boldsymbol{\lambda})]. && (2.46) \end{aligned}$$

An example of a latent variable model without data is the Ising model—in this case, the data is an empty set, $\mathbf{x} = \{\}$. In this case $\mathcal{L}(\lambda)$ is a lower bound on the log partition function as we derived in Equation (2.17) and identical to the GBF inequality.

2.4 Conclusion

We reviewed probability models and gave examples of their use in statistical physics and recommender systems. The task of inference is central to working with probability models; we described variational inference and maximum likelihood estimation. The following chapters address the issue of building the structure of a problem into a performant probability model, whether that structure concerns the connectivity in a statistical physics model, the structure of datapoints in a recommender system, or information about a variational approximation useful in an optimization algorithm for this approximation.

Chapter 3

Hierarchical Variational Models for Statistical Physics

As probabilistic modeling finds widespread use in science, it is necessary to adapt machine learning tools to the specifics of a scientific domain. In this chapter we focus on probabilistic models used in statistical physics. Whether such models are used in molecular dynamics simulations for finding drug candidates for disease (Shamay et al., 2018) or simulating solid state systems for materials design (Schmidt et al., 2019), the scalability of machine learning methods is a bottleneck for progress. Simulations need to be run for longer timescales and in larger systems, and must leverage knowledge about the physical system under study to yield accurate predictions. As a step in this direction, this chapter centers on building scalable variational approximations for statistical physics models, and develops hierarchical probabilistic models to study their performance in large statistical physics systems.

3.1 Introduction

In statistical physics, building a model corresponding to a physical system is useful: comparing how the model's predictions differ from experiment can be used to understand how a system behaves. For computing properties corresponding to a model's behavior, the normalization constant of the model's Boltzmann distribution in Equation (2.1) is a central quantity. The partition function can be used to derive properties of physics models that can be measured in experimental realizations, such as specific heat or magnetization. Such properties of models can be compared to experimental values, which can inform how a model might be improved to better mirror reality. But the partition function is intractable for many probabilistic models of interest, as described in Chapter 2.

One workaround to the problem of an intractable partition function is to use an approximate inference algorithm such as Markov chain Monte Carlo (MCMC). MCMC relies on sampling likely configurations of a system and does not require calculating the partition function. In theory, these samples will be draws from the probability model of interest (Metropolis et al., 1953; Andrieu et al., 2003). For example, samples from the Boltzmann distribution can be used to approximate physical quantities derived from the probability model, such as specific heat.

However, with limited computation MCMC has limitations. This method requires practitioners to use convergence diagnostics (Brooks and Gelman, 1998) to assess whether samples from the algorithm are independent. Scalable MCMC requires careful consideration. While some scalable versions of MCMC have been developed (Neal et al., 2011; Welling and Teh, 2011), they are biased samplers that may not have guaranteed convergence to samples from the probability model of interest. This is similar to how in VI performance must often be assessed empirically. But in comparison to VI, unless a model-specific algorithm has been developed (Wolff, 1989), generic MCMC methods do not readily scale to large numbers of random variables.

In Chapter 2 we showed that the machine learning framework of VI is equivalent to the Gibbs-Bogoliubov-Feynman variational principle. This has allowed practitioners to study statistical physics models using many variational approximations, including variational autoregressive networks (VANs). As an example of a variational method enabled by VI, we study hierarchical variational models (HVMs) as approximations to the Boltzmann distribution of statistical physics models. We find that HVMs scale to larger systems sizes than VANs in Sherrington-Kirkpatrick and Ising models. Testing the feasibility of VI methods in statistical physics is a twofold opportunity. Statistical physics problems might serve as benchmarks for VI, and using VI for these problems can lead to improved computational methods in statistical physics.

Related Work. The GBF variational principle has been used to study Markov random fields (Zhang, 1996) and the connection between variational inference and statistical physics has been well-documented (Blei et al., 2017; Hoffman et al., 2013; MacKay, 2003). But this equivalence between VI and the GBF inequality might serve as an introduction to VI for physicists. Wu et al. (2019) implicitly use VI, by developing VANs and a reinforcement learning policy gradient algorithm (however, VI is not mentioned). Further, for a system of size L, autoregressive neural networks require $\mathcal{O}(L^2)$ forward passes to sample a system configuration, making VANs intractable in larger systems. The use of HVMs can be advantageous for statistical physics as these models can sample from a system in $\mathcal{O}(L)$ time and yield results for larger systems.

Variational Inference. VI is equivalent to the GBF variational principle and requires similar choices of a practitioner. The variational family $q(\mathbf{z}; \boldsymbol{\nu})$ to approximate a model must be chosen, in addition to a method to maximize the variational lower bound in Equation (2.17).

The VI literature provides several choices of variational family, such as a mean field, factorized variational distribution with independent latent variables. Another choice of variational family is the Bethe approximation, which constrains the variational distribution to the polytope of mean parameters that captures correlations between any two latent variables (Wainwright and Jordan, 2008). Some machine learning research focuses on developing variational approximations that capture correlations between latent variables (Hoffman and Blei, 2015; Kingma et al., 2016; Maaløe et al., 2016; Wu et al., 2019). An example of a variational family that can model correlations between latent variables is the VAN family (Wu et al., 2019), which uses autoregressive neural networks to parameterize the variational distribution $q(\mathbf{z}_i \mid \mathbf{z}_1, \ldots, \mathbf{z}_{i-1})$. We explore the HVM class of variational approximations (Ranganath, 2018).

The second choice required to employ VI is how to optimize the variational lower bound in Equation (2.17). The choice of variational family can limit the available optimization techniques. For a simple variational family like the mean field approximation, it may be possible to analytically evaluate the expectations in Equation (2.17). Then derivatives of the variational bound with respect to the variational parameters $\boldsymbol{\nu}$ and manual calculation can maximize the lower bound, as derived in Section 2.3.1. If more expressive variational families are used (e.g. VANs with thousands or millions of variational parameters), the analytic approach is infeasible. Stochastic optimization and automatic differentiation software have been used to develop several approaches to computing gradients of the variational lower bound, such as black box variational inference (Ranganath, 2018; Mohamed et al., 2019).

The choice of variational family $q(\mathbf{z}; \boldsymbol{\nu})$ and optimization method for maximizing the variational lower bound leads to a trade-off intrinsic to VI. Simple variational approximations such as the mean field family may be computationally feasible but inaccurate. The cost of increased accuracy, say by using a structured variational approximation, is increased computation. We illustrate the use of VI in statistical physics by comparing two choices of variational approximation, HVMs (Ranganath, 2018) and VANs (Wu et al., 2019). Many other variational approximations can be explored in future work.

3.2 Hierarchical Variational Models

For studying models with correlated random variables, such as frustrated spin systems (Zdeborová and Krzakala, 2016), unstructured variational families such

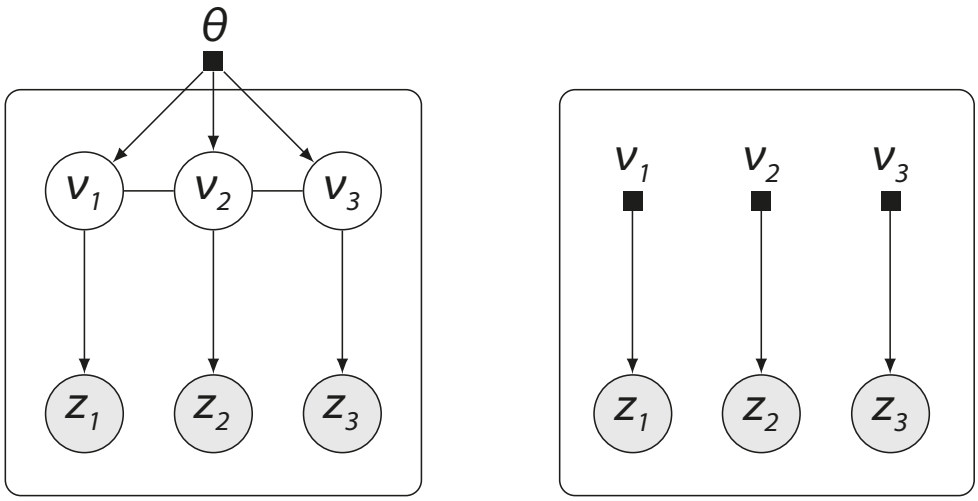


Figure 3.1: Hierarchical variational models (HVMs, left) capture dependencies between latent variables, compared to the mean field variational family with independent variables (right).

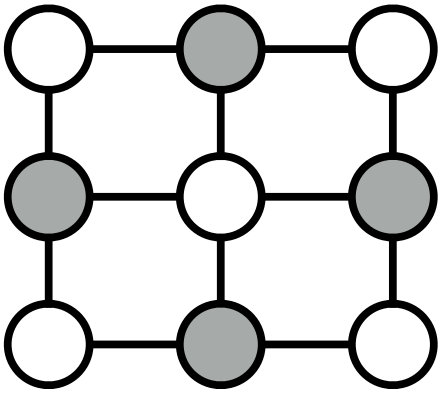


Figure 3.2: The Markov blanket of a node in an Ising model consists of the node's nearest neighbors (nodes in the Markov blanket of the central node are shaded). Conditioning on the Markov blanket of a node in a graphical model renders it conditionally independent of the rest of the variables. This enables building efficient variational approximations.

as the mean field are insufficient. Hierarchical variational models (HVMs) are one way to model correlated latent variables. An HVM is defined by placing a 'variational prior' on the variational parameters $\boldsymbol{\nu}$ of the mean field variational family, in analogy to hierarchical probabilistic models. By leveraging neural networks to parameterize the variational prior, HVMs can capture complex dependencies between random variables (Ranganath, 2018).

For studying a model $p(\mathbf{x}, \mathbf{z})$, the variational family defined by an HVM is defined as

$$q_{\text{HVM}}(\mathbf{z}; \theta) = \int q(\boldsymbol{\nu}; \theta) \prod_i q(\mathbf{z}_i \mid \boldsymbol{\nu}_i) d\boldsymbol{\nu}, \tag{3.1}$$

where $q_{\text{MF}}(\mathbf{z} \mid \boldsymbol{\nu}) = \prod_i q(\mathbf{z}_i \mid \boldsymbol{\nu}_i)$ is the mean field 'variational likelihood' with parameters $\boldsymbol{\nu}$, and $q(\boldsymbol{\nu}; \theta)$ is the variational prior with parameters θ. Figure 3.1 shows the graphical model for HVMs as compared to the mean field family graphical model.

To use an HVM in VI, the variational lower bound must be optimized. But the variational lower bound in Equation (2.17) requires calculating the entropy of the variational distribution, and such integration in high dimensions can be intractable. As detailed in Ranganath (2018), the entropy can be lower-bounded by introducing an auxiliary 'variational posterior' distribution $r(\boldsymbol{\nu} \mid \mathbf{z}; \boldsymbol{\phi})$ with parameters $\boldsymbol{\phi}$. This leads to the hierarchical evidence lower bound,

$$\widetilde{\mathcal{L}}(\theta, \boldsymbol{\phi}) = \mathbb{E}_{q(\mathbf{z}, \boldsymbol{\nu}; \theta)}[\log p(\mathbf{x}, \mathbf{z}) + \log r(\boldsymbol{\nu} \mid \mathbf{z}; \boldsymbol{\phi}) - \log q(\mathbf{z} \mid \boldsymbol{\nu}) - \log q(\boldsymbol{\nu}; \theta)], \tag{3.2}$$

and a stochastic optimization algorithm for this objective is developed in (Ranganath, 2018). VI with an HVM requires specifying the variational prior $q(\boldsymbol{\nu}; \theta)$ and the variational posterior $r(\boldsymbol{\nu} \mid \mathbf{z}; \boldsymbol{\phi})$, then optimizing the hierarchical ELBO in Equation (3.2).

Specifying an HVM with Normalizing Flows. We study several choices of variational prior and recursive variational posterior. One choice of variational prior $q(\boldsymbol{\nu}; \theta)$ is an inverse autoregressive flow (Kingma et al., 2016). If the variational posterior $r(\boldsymbol{\nu} \mid \mathbf{z}; \boldsymbol{\phi})$ is chosen to be a masked autoregressive flow (Papamakarios et al., 2017), the analytical forms of these flows are equivalent. These choices lead to a complexity of $\mathcal{O}(L)$ for sampling latent variables in a system of size L. (This is because the noise used to sample from the variational prior can be drawn in parallel.) HVMs should therefore be faster than VAN approximations in large systems: the autoregressive requirement in VANs leads to a complexity of $\mathcal{O}(L^2)$. A research question is whether the advantage in speed of HVMs leads to a drop in accuracy that is too large to answer a statistical physics question.

Scalable HVMs using Ising Model Structure. Although HVMs with autoregressive flows scale linearly, such variational approximations do not leverage the structure about the statistical physics under study. For example, it is difficult to index random variables so that nearest neighbors are grouped together when fed to an autoregressive model. However, consider the Markov blanket of a random variable in the Ising model—it contains all the information needed to render a variable conditionally independent of the rest of the model. The Markov blanket of a node in an Ising model consists of a node's nearest neighbors and is shown

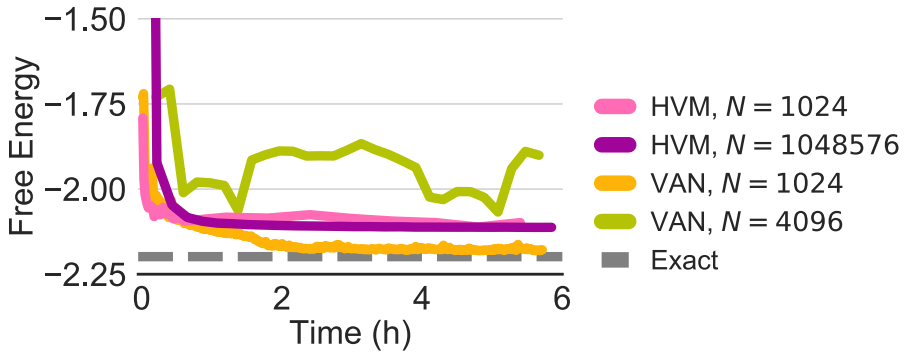


Figure 3.3: Hierarchical variational models (HVMs) scale to statistical physics models with millions of random variables, with over 100x parameter savings. The free energy is reported (the variational lower bound yields an upper bound on the free energy). The HVM variational approximations use 5400 parameters, while the VAN method uses over 700k.

in Figure 3.2. This means that an autoregressive model is overparameterized. For example, the last variable to be fed to the model depends on all the previous in an autoregressive model, whereas an efficient model might only consider nodes in a Markov blanket. A convenient way to build this structure into an HVM would ensure efficient use of information from nearest neighbors in the Ising model. One way to formalize this problem structure is with convolutional neural networks (LeCun et al., 2015). We parameterize the variational prior and recursive posterior with real non-volume preserving (REALNVP) transformations using a convolutional neural network architecture (Dinh et al., 2017). Specifically, we parameterize the convolutional kernels to mimic the Markov blanket shown in Figure 3.2: for every node, only its nearest neighbors are conditioned on.

3.3 Empirical Study

To study the utility of VI tools for statistical physics systems, we compare HVMs to VAN approximations (Wu et al., 2019).[1] We use the same benchmarks as in Wu et al. (2019): the models of Ising and Sherrington-Kirkpatrick. We evaluate variational inference methods by assessing whether they lead to lower estimates of the free energy of a model. (Lower is better, as the free energy in Equation (2.4) is proportional to the negative of the variational lower bound in Equations (2.17) and (3.2).)

Experimental Setup. To assess whether HVMs outperform VANs in large systems, the computational budget for the VI algorithm using both variational ap-

[1] Code is available at https://github.com/altosaar/hierarchical-variational-models-physics.

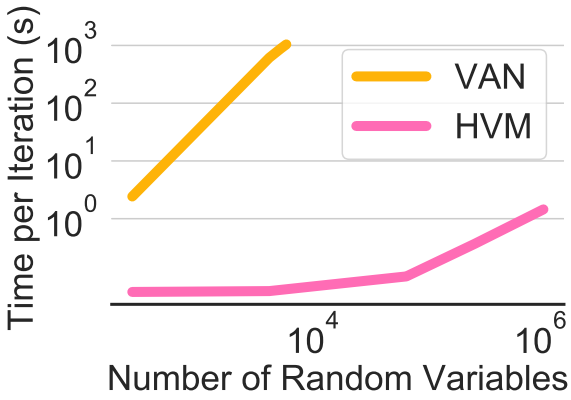


Figure 3.4: Hierarchical variational models (HVMs) are faster than Variational autoregressive networks (VANs) (Wu et al., 2019) and scale to larger systems. The scaling of both variational approximations is illustrated with the time taken per iteration in Ising models. The HVM variational approximations use 5400 parameters and the VAN method uses over 700k. The VAN approximation runs out of memory with 16384 random variables, while the HVM method scales to models with over 1M random variables.

proximations was set to 6 hours. All experiments were performed on NVIDIA Tesla P100 GPUs, and the reference implementation of VANs released in Wu et al. (2019) was used. VAN models were unable to complete sufficiently many iterations in the allocated compute time, so all experiments were run without annealing the temperature of the system. For calculating the free energy using HVMs, importance sampling (Owen, 2013) was used with an HVM as the proposal (for VANs, the increased cost of sampling prohibited drawing enough samples for low-variance importance sampling estimates, so Monte Carlo estimation was used). In HVMs, variational approximations that accounted for problem structure using REALNVP transformations outperformed autoregressive parameterizations, and we omit these results.

Ising Model. For small systems, HVMs were more accurate than VAN models at lower temperatures; at higher temperatures (such as the critical temperature), VAN models were slightly more accurate. This could be because annealing was not used to fit VAN models, and the randomness of the hierarchical latent variables in HVMs obviates the need for annealing. In large systems (e.g. $L = 128$), VAN models failed to complete a single iteration, while HVMs were able to complete many iterations (at the cost of some accuracy). The trade-off between model size and computational cost was significant between autoregressive choices of variational approximations in HVMs versus REALNVP convolutional approximations. Figure 3.3 shows that convolutional models were able to scale to models with over a million random variables, with only a slight decrease in accuracy relative to VAN

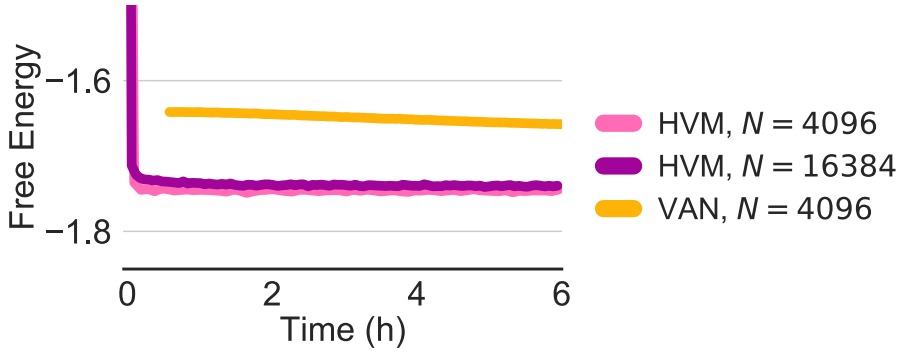


Figure 3.5: Hierarchical variational models (HVMs) scale to larger systems than variational autoregressive network (VAN) models (Wu et al., 2019) when fit to the Sherrington-Kirkpatrick model using variational inference. (Lower is better, as the variational lower bound yields an upper bound on the free energy.) For the system with $N = 4096$ variables the VAN method completed fewer than ten iterations, and with $N = 16384$ did not complete a single iteration.

methods, and with 100 times fewer parameters. This is also illustrated via the time per iteration for both a VAN and HVM variational approximation reported in Figure 3.4.

Sherrington-Kirkpatrick Model. The free energy estimates using VI with either HVM or VAN approximations are plotted in Figure 3.5 for the Sherrington-Kirkpatrick model. HVM approximations outperformed VAN approximations, and scaled to larger systems where the $\mathcal{O}(L^2)$ cost of sampling from a VAN prohibited even a single iteration.

3.4 Discussion

The GBF inequality holds for quantum systems (Feynman, 1972; Feynman, 2018), and applying VI and HVMs to quantum systems is a direction for future work. Physics tools (such as VI in its original incarnation) have been useful in machine learning (Bamler et al., 2017), and we hope the reverse holds—that tools from machine learning such as VI and HVMs continue to find use in statistical physics. Further scaling HVMs to statistical physics models where system size is a bottleneck is a goal of future work, especially in settings relevant to medicine, such as in protein folding or drug screening problems.

Chapter 4

RankFromSets: Scalable Set Recommendation with Optimal Recall

In the previous chapter, we built scalable and performant probabilistic models of likely configurations of interacting atoms in statistical physics systems. Modeling choices were guided by the structure of the underlying patterns of interaction between random variables. As another case study, we turn to recommender systems in this chapter, where the core problem is to model which items a user is likely to interact with. By building the structure of individual datapoints into a probabilistic model of user interaction, and considering the goals of the recommendation task, we develop a scalable, accurate framework for recommending items with attributes.

4.1 Introduction

Classical recommender system datasets contain a matrix where each row is a user and each column is an item. Each entry in the matrix indicates whether or not a user consumed an item. Modern applications often gather rich side information about items in the form of a set of attributes or tags. Item attributes provide valuable side information for recommender systems. With a large number of items or a sparse user-item matrix, attribute information is necessary for good performance.

We are motivated by a specific dataset with these properties: a dataset of 55k users logging 16M meals using the LoseIt! diet tracking app. Table 4.1 shows the kind of data logged by users, where each row is an item (meal), each left-hand column is an attribute (food), and each right-hand column is a user. The food attributes can clearly inform recommendations: User 1 does not log meat, User 4

	Attributes									Users				
Items	Pizza	Eggs	Taco	Salad	Avocado	Chicken	Sardines	Beer	Coffee	1	2	3	4	5
Morning Pizza	•	•							•		•		•	
Dinner Pizza	•						•	•		•			•	
Small Salad				•	•		•					•		•
Big Salad		•		•	•	•			•			•	•	
Taco			•		•	•			•					•
Fish Taco			•				•				•		•	

Table 4.1: An example of the data we focus on, where tagged items are recommended to users based on both item attributes and items users have consumed in the past. This example dataset of meals contains meals with different foods (left) and users log which meals they ate (right). The goal is to leverage the attributes to recommend items to users.

is omnivorous and undiscriminating, and User 3 mostly eats salads. In the LoseIt! data, there is a massive number of possible items to recommend: there are 12M unique meals, composed of subsets of 3M foods. Meals containing only a few foods, or those ordered at chain restaurants, may be logged by many users. But these represent a small proportion of the meals people actually eat, so a long tail of meals are logged by single users.

Modeling item attributes in these non-standard recommender systems is not straightforward. Popular ways to use item attributes like multiple matrix factorization (Wang and Blei, 2011; Gopalan et al., 2014) struggle when the attribute vocabulary or number of items is large. Conversely, simple models are computationally tractable but risk losing the ability to capture nonlinear patterns of user consumption. For instance, a user may enjoy meals tagged with foods `A` and `B`, `B` and `C`, or `A` and `C`, but not all three. Finding the right balance between scalability and flexibility is therefore a primary goal.

Even when a model can be scaled, it may not be clear how its training procedure connects to the recommender system evaluation metric. A matrix factorization method might minimize mean squared error when the recommender system is evaluated on recall. While it is plausible that minimizing mean squared error will improve recall, the connection between the two is implicitly assumed in many methods. Ideally, a recommender system should have an objective that matches its evaluation metric.

This paper proposes RANKFROMSETS (RFS), a class of principled, scalable models for recommending items with sets of attributes. RFS casts the recommendation problem as binary classification. Given a user and an item, RFS treats attributes as features and classifies whether or not the item is likely to be consumed by the user. RFS learns embeddings for each user and attribute; each item is represented as the mean of its attribute embeddings. To scale to large datasets, we develop

an RFS method that is trained using negative sampling of random items that are unlikely to be consumed.

RFS enjoys two benefits from framing the recommendation problem as classification. First, the RFS classification objective function is directly tied to recommender recall: we show that a classifier with zero worst-case error achieves maximum recall. Second, RFS is provably flexible enough to learn any class of recommendation model based on set-valued side information (including multiple matrix factorization). This generality makes RFS a natural drop-in replacement for many specialized models in the literature.

We study the performance of the negative sampling RFS model on a semi-synthetic benchmark dataset and the LoseIt! dataset. The semi-synthetic paper recommendation dataset consists of 65k users clicking on 636k papers posted to the arXiv; the attributes of each paper are the unique words in its abstract. We then apply the method to the LoseIt! dataset to make out-of-sample meal recommendations. In both cases, the RFS method outperforms the state of the art in terms of recall. In addition to good performance, the RFS model learns interpretable embeddings that intuitively capture the structure of the underlying data.

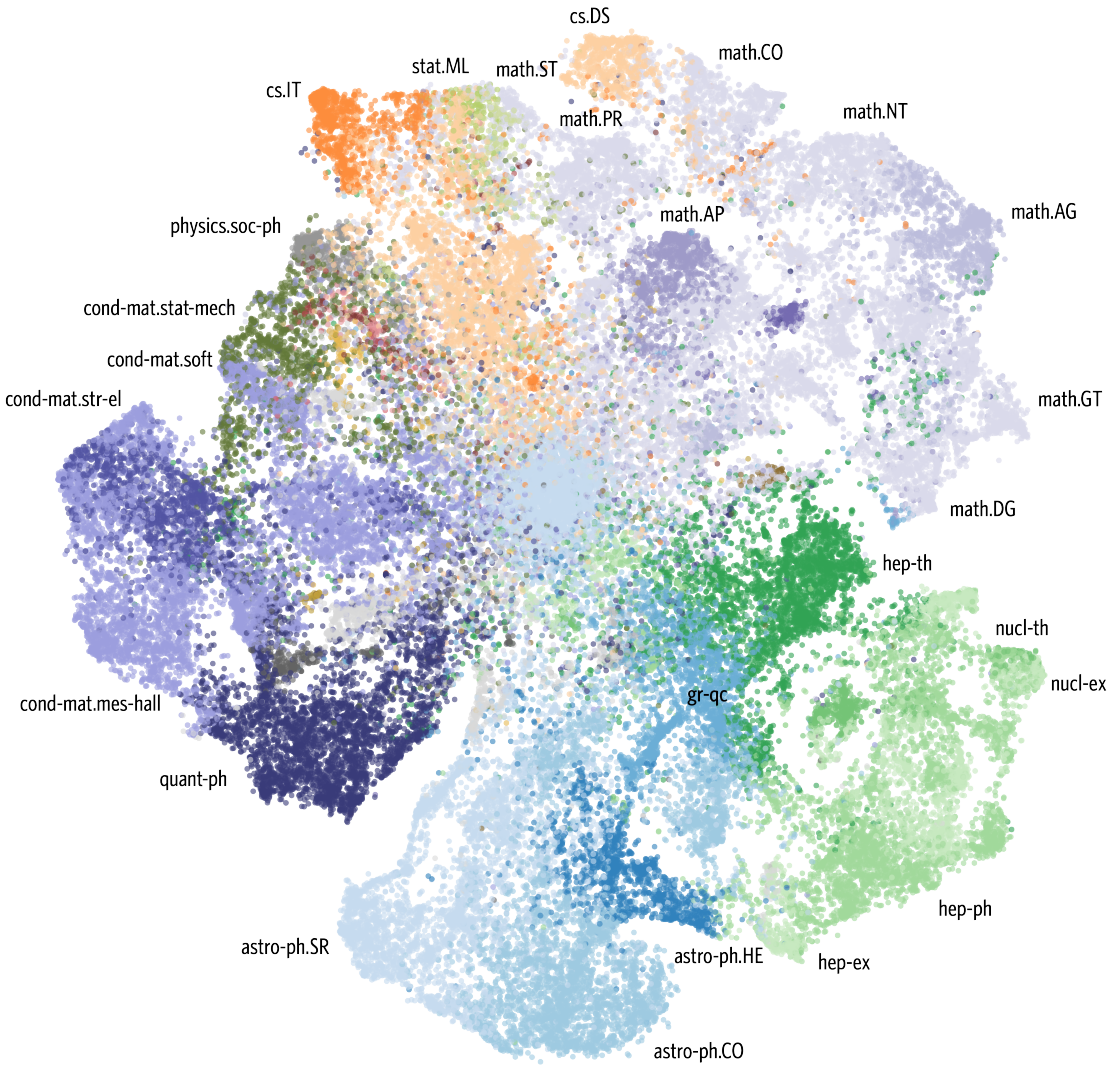


Figure 4.1: RANKFROMSETS trained on arXiv reading behavior clusters researchers by their most frequently-read arXiv category (best viewed on a screen). RFS is trained to recommend items using their attributes (words in the abstract). t-SNE (Maaten and Hinton, 2008) is used to visualize the user embeddings θ_u in the inner product regression function in Equation (4.2). Each marker represents a user embedding; its color represents a user's most-read arXiv category. Unique colors are determined using the most-read categories across the arXiv, and colors are assigned according to the arXiv ontology. RFS captures usage patterns, as fields of study are related by patterns of reading behavior across neighboring fields (e.g. `stat.ML` and `cs.IT`).

4.2 RankFromSets

RANKFROMSETS (RFS) is a class of recommendation models that recommend items with attributes to users. Let $u \in \{1, \ldots, N\}$ be a user, $m \in \{1, \ldots, M\}$ be an item, and $y_{um} \in \{0, 1\}$ be a binary indicator where 1 indicates user u consumed item m. For each item m, there is an associated set of attributes $x_m \in \{0, 1\}^{|V|}$ from a vocabulary of V attributes. The observed data is a collection of user-item interactions $\{(u, m)\}$ and the sets of attributes associated with items $\{x_m\}$.

We assume that a recommendation model is given a budget of K recommendations to be made for each user. In response, the recommender system produces a list of K distinct recommendations $\mathbf{r}_u = (r_{u1}, \ldots, r_{uK})$ for each user. The goal of the recommendation task in this paper is to maximize the expected Recall@K,

$$\text{Recall@}K = \mathbb{E}_u \left[\frac{\sum_{r \in \mathbf{r}_u} y_{ur}}{\sum_m y_{um}} \right], \tag{4.1}$$

with the expectation over users in the empirical distribution $\mathcal{D}$.

We combine three techniques to maximize Recall@K with RFS. First, we cast recommendation as a classification task. Second, we learn user- and attribute-level embeddings. Statistical strength is shared between items with similar attributes by representing items as the mean of their attribute embeddings. Third, we scale RFS to large datasets using a stochastic optimization-based negative sampling training procedure.

RFS casts the recommendation problem as a classification task. Given a user-item pair (u, m) and regression function f, RFS learns to predict the probability that item m will be consumed by user u:

$$p(y_{um} = 1 \mid u, m) = \sigma\left(f\left(u, x_m\right)\right),$$

where x_m is the set of attributes of item m and σ is the sigmoid function. Recommendations made by RFS are the maximum likelihood set formed by ranking a set of items for a user according to the model $f(u, x_m)$. We motivate treating recommendation as classification with the following observation.

Proposition 1. *Let $u \in \mathcal{U}$ be a user, $m \in \mathcal{M}$ be an item, and $y(u, m) \in \{0, 1\}$ be an indicator of whether user u logged item m. Let $\mathcal{E}$ be the worst-case error for binary classifier $\hat{y}(u, m)$ on any (u, m) pair drawn from the data $\mathcal{D}$,*

$$\mathcal{E} = \max_{(u,m) \in \mathcal{D}} \mathbb{1}\left[\hat{y}(u, m) \neq y(u, m)\right].$$

A binary classifier with zero worst-case error ($\mathcal{E} = 0$) maximizes recommendation recall.

Proof. A model with zero worst-case error is a perfect classifier, assigning greater probability to data with positive labels than to data with negative labels. In other words, it ranks positive examples above negative examples. Recall@K is measured by the fraction of items with positive labels in a ranking returned by the model. In a classifier that achieves zero worst-case error, positively-labeled datapoints must be ranked higher than other datapoints, maximizing recall. □

Proposition 1 is simple, but conceptually important. Under the assumption that a perfect classifier exists, a consistent method for learning a classifier will be a consistent method for learning a recommendation system that targets expected recall. Put another way, recall is inherently binary: a model does or does not recall an item; an item is or is not in the top K recommendations in the numerator of Equation (4.1). So the best one can hope to do if recall is used to assess recommendation performance is to train a binary classifier. In practice, as with any regression method, a perfect classifier is unachievable. Proposition 1 is a guiding principle rather than a finite-sample guarantee of maximal performance. As we show in Section 4.4, the classification approach of RFS performs well in practice.

For recommending items with attributes, Proposition 1 says that building a classifier such as RFS is optimal if we measure recommendation performance with recall. To parameterize the RFS classifier, a regression function $f(u, x_m)$ is needed. A straightforward parameterization is an inner product,

$$f(u, x_m) = \theta_u^\top \left(\frac{1}{|x_m|} \sum_{j \in x_m} \beta_j + g(x_m) \right) + h(x_m). \tag{4.2}$$

Each element in the inner product regression function in Equation (4.2) has an intuitive interpretation. The user embedding $\theta_u \in \mathbb{R}^d$ captures the latent preferences for user u. This captures the individual-level tastes of a user and is analogous to the user preference vector in classical collaborative filtering or the row embedding in matrix factorization. The attribute embedding $\beta_j \in \mathbb{R}^d$ is the latent quality conveyed through item m having attribute j. (The set x_m contains only attributes with $x_{mj} = 1$. Attributes that are not associated with item m are ignored.) The item embedding function $g(x_m) \in \mathbb{R}^d$ represents qualities not conveyed through the set of item attributes. This term in the regression function enables collaborative filtering by capturing unobserved patterns in item consumption such as popularity. We describe how to construct this function below. The scalar item intercept function $h(x_m) \in \mathbb{R}$ makes an item more or less likely due to availability.

To define scalable item embedding and item intercept functions, note that the parameterization of the item embedding function $g(x_m)$ depends on the size of the data. If the number of items is small, g can function as a lookup for unique in-

Query Item	Nearest Item by Cosine Similarity
Two scoops of Raisin Bran cereal, organic Moroccan green tea, almond milk, light honey, tap water, large banana, large strawberries	Vita Bee bread, salted butter, fresh medium tomatoes, large fried whole egg, small banana
Iceberg lettuce, cantaloupe cubes, diced honeydew melon, cherry tomatoes, olives, dry-cooked unsalted hulled sunflower seed kernels, chopped hard-boiled egg, cucumbers, dried cranberries, fat-free ranch dressing	Green leaf lettuce, chopped sweet red bell peppers, crumbled feta cheese, large hard-boiled egg, chopped cucumber, oil-roasted salted sunflower seeds, sliced radishes, sliced strawberries, pitted Calamata olives, fat-free balsamic vinegar
Boston roast pork, mackerel, artichoke hearts, spinach, pimiento-stuffed Manzanilla olives, carrots, mushrooms, peppercorn ranch dressing	Broiled top round steak, tomatoes, cucumber, baby yellow squash, zucchini, black olives, extra virgin olive oil
Meatloaf with tomato sauce, chopped sweet red bell peppers, extra virgin olive oil, cooked asparagus spears, sweet potatoes, orange, cantaloupe cubes	Chicken breast, breadcrumbs, fresh tomatoes, shredded green leaf lettuce, extra virgin olive oil, spinach, chopped yellow onion, sweet large yellow bell peppers, whole mushrooms, chili peppers, vinaigrette
Ciabatta bun, cooked skinless chicken breast, fresh baby spinach, shredded iceberg lettuce, shredded mozzarella cheese, ketchup, frozen yogurt bar	Small whole wheat submarine roll, broiled round roast beef, roasted light turkey meat without skin, fresh medium tomatoes, honey smoked ham, shredded iceberg lettuce, sliced mozzarella cheese

Table 4.2: RANKFROMSETS trained on food consumption data provides diverse meal recommendations. RFS with Equation (4.4) is fit to data from a diet tracking app; items are meals and attributes are the ingredients in the meal. Meals are represented the average of their attribute embeddings, and cosine similarity between meal representations is used to find the nearest neighbors of meals (user-level information cannot be shown as this is personal diet data). RFS reveals eating patterns: for example, the second-last query meal is a mix of meat, vegetables, and fruit, and the nearest neighbor meal is a different meat with a side of salad; the last query meal is a sandwich, and its nearest neighbor is also a sandwich with different ingredients.

tercepts for every item. However, if the number of items is so large that unique item intercepts lead to overfitting, a scalable parameterization of item embeddings g can be defined using additional information about every item. For example, if the data consists of foods in meals, we can define a meal intercept as the mean of food intercepts, yielding a scalable item intercept function. The item intercept function $h(x_m)$ that maps item attributes to scalars is constructed in the same way. We study both of these choices in Section 4.4.

The inner product regression function in Equation (4.2) has several benefits. It requires computing a sum over only the attributes with which each item is associated. This enables RFS to scale to large attribute vocabularies where traditional matrix factorization methods are intractable. Second, the embed-and-average approach to set modeling is provably flexible as we show later. We now describe deep variants of RFS and detail how RFS can approximate other recommendation models.

The RFS inner product regression function in Equation (4.2) is a log-bilinear model. But there are several other choices of regression function, and we draw on the deep learning toolkit for classification to build two other example architectures. With finite data and finite compute, one architecture may outperform another, or prove insufficient to capture patterns in user consumption. (Later, we show that all architectures are equivalent under fewer assumptions.) First, as an alternative to the log-bilinear model in Equation (4.2), we can use a deep neural network as a regression function:

$$f(u, x_m) = \phi\left(\theta_u, \frac{1}{|x_m|}\sum_{j \in x_m} \beta_j, g(x_m)\right) + h(x_m), \tag{4.3}$$

where the deep network ϕ has weights and biases and takes as inputs the user embedding, sum of attribute embeddings, and item intercept. Such a neural network can represent functions that may or may not include the inner product in Equation (4.2); *ex ante*, it is unclear whether a finite-depth, finite-width neural network can represent the inner product.

Another regression function for RFS is a combination of Equations (4.2) and (4.3), using an idea borrowed from deep residual networks for image classification (He et al., 2016). In this architecture, a neural network ϕ with the same inputs as in Equation (4.3) learns the residual of the inner product model:

$$f(u, x_m) = \theta_u^\top\left(\frac{1}{|x_m|}\sum_{j \in x_m} \beta_j + g(x_m)\right) + \phi + h(x_m). \tag{4.4}$$

The choice of regression function in RFS depends on the data. On finite data, with finite compute, one parameterization of RFS will outperform another. To demonstrate this, we simulated synthetic data from the same generative process RFS employs with a ground-truth regression function (a square kernel), and found that the residual and deep parameterizations outperformed the inner product architecture. These results are included in Section 4.7.4, and motivate exploring other architectures than the three examples here.

Stepping back from the setting of finite data and compute, a bigger picture emerges, which reveals the choice of regression function in RFS does not matter.

We show that any RFS architecture is sufficiently flexible to approximate recommendation models that operate on set-valued input. We define permutation-invariant models before deriving this result.

The regression function f in RFS operates on set-valued input: the unordered collection of item attributes x_m. A set is, by definition, permutation-invariant: it remains the same if we permute its elements. Functions that operate on set-valued inputs must also be permutation-invariant. RFS is permutation-invariant; the set of attributes associated with an item enter into Equations (4.2) to (4.4) via summation. Other examples of permutation-invariant recommendation models are multiple matrix factorization, models based on word embeddings, and permutation-marginalized recurrent neural networks. These models are shown to be permutation-invariant in Section 4.3 and evaluated in Section 4.4. We now show that RFS can approximate other permutation-invariant recommendation models such as matrix factorization.

Proposition 2. *Assume the vocabulary of attributes (set elements) is countable,* $|V| < |\mathbb{N}_0|$. *Then RFS can approximate any permutation-invariant recommendation model.*

The proof follows directly from Theorem 2 in Zaheer et al. (2017) and we will not restate it here. (The only change to the proof is the mapping from set elements to one-hot vectors, $c : V \to \{0,1\}^{|V|}$ to yield a unique representation of every object in the powerset.) Proposition 2 means that any of the parameterizations in Equations (4.2) to (4.4) is flexible enough to approximate other principled recommendation models that leverage item attributes, such as multiple matrix factorization (Gopalan et al., 2014; Wang and Blei, 2011).

The parameters for RFS are learned by stochastic optimization. Denote the full set of RFS model parameters by γ, and let $\mathcal{D}_u$ be the empirical data distribution for a user. Let λ_u be a reweighting parameter. The per-user maximum likelihood objective for RFS is

$$\mathcal{L}(\gamma, \lambda_u) = \mathbb{E}_u\left[\mathbb{E}_{m\sim\mathcal{D}_u|y_{um}=1}\left[\log p(y_{um}=1 \mid x_m;\gamma)\right] + \lambda_u \mathbb{E}_{k\sim\mathcal{D}_u|y_{uk}=0}\left[\log p(y_{uk}=0 \mid x_k;\gamma)\right]\right] \tag{4.5}$$

In traditional regression, altering the ratio of positive to negative examples by reweighting leads to inconsistent parameter estimation. The inconsistency stems from the randomness in the labels, given the features. However, Recall@K assumes that each user, item attribute set pair (u, x_m) uniquely determines whether the item was consumed or not (the label y_{um}). Here, all reweightings produce the same result. This means that for any negative example weight λ_u, the learned model will be the same. In practice we set λ_u to balance the positive and negative examples for each user. We use stochastic optimization to maximize Equation (4.5), and describe two negative sampling schemes that are dependent on the choice of evaluation metric.

Negative samples can be drawn uniformly over the entire corpus of items, which we define to be corpus sampling. If the item set is large, this can be an expensive procedure. This negative sampling scheme leads to objective functions used in other recommender systems (He et al., 2017; Song et al., 2018).

On large datasets, it is infeasible to calculate Recall@K for evaluation, as this requires ranking every item for every user (e.g. in Section 4.4 we study a dataset with over 10M items). We define a scalable evaluation metric based on recall, and describe how it leads to a natural choice of negative sampling distribution.

Sampled recall is defined as follows. Consider held-out datapoints with positive labels, $(x_m, y_{um} = 1)$. For every held-out datapoint, $K-1$ datapoints with negative labels $(x_k, y_{uk} = 0)$ are sampled from the rest of the held-out data, which together yield a set of K datapoints. A recommendation model is used to rank the K datapoints $r_{u1}, \ldots, r_{uK}$. SampledRecall@k is the fraction of the K held-out datapoints that the model ranks in the top k:

$$\text{SampledRecall@}k = \frac{1}{K}\mathbb{E}_{um}\left[\sum_{r \in \{\mathbf{r}_{u1}, \ldots \mathbf{r}_{uk}\}} y_{ur}\right]. \tag{4.6}$$

The expectation is over users and items in the held-out set of datapoints. This evaluation metric is scalable: instead of using a model to rank every item, SampledRecall@k requires ranking only K items. Sampled recall is 1 if $k = K$, as the held-out datapoint with $y_{um} = 1$ is in each list of K datapoints to be ranked. This metric is used in recommender systems when the number of items is large (Ebesu et al., 2018; Yang et al., 2018).

When sampled recall is used as an evaluation metric, batch sampling is a natural way to draw negative samples. Sampled recall is calculated on items drawn from other user's data. We define batch sampling as generating negative samples by permuting mini-batch items. Besides corresponding to the sampled recall metric, this technique is memory-efficient, as it requires that only the current mini-batch be in memory.

In addition to scalability, both negative sampling procedures above have the advantage of implicitly balancing the classifier. As shown in Veitch et al. (2019), using stochastic gradient descent with negative sampling is equivalent to a Monte Carlo approximation of the reweighted (balanced) classification loss.

4.3 Permutation-invariant Recommender Models

Proposition 2 shows that RFS can approximate permutation-invariant recommendation models. We describe several common recommendation models and show that they are permutation-invariant, before comparing their performance to RFS in Section 4.4.

Gopalan et al. (2014) develop a probabilistic matrix factorization model of user consumption data. Collaborative topic Poisson factorization (CTPF) models user preferences using a generative process,

1. **Document model:**
   (a) Draw topics $\beta_{vk} \sim \text{Gam}(a, b)$
   (b) Draw document topic intensities $\theta_{dk} \sim \text{Gam}(c, d)$
   (c) Draw word count $w_{dv} \sim \text{Poisson}(\theta_d^T \beta_v)$.
2. **Recommendation model:**
   (a) Draw user preferences $\eta_{uk} \sim \text{Gam}(e, f)$
   (b) Draw document topic offsets $\epsilon_{dk} \sim \text{Gam}(g, h)$
   (c) Draw $r_{ud} \sim \text{Poisson}(\eta_u^T(\theta_d + \epsilon_d))$.

To show that CTPF is permutation-invariant, consider the Poisson likelihood function over words w_{dv}. Conditional on the latent item representation θ_d and latent word representation β_v, every word in the document w_{dv} is independent; the joint probability of words in a document factorizes:

$$p(w_d \mid \theta_d, \beta_v) = \prod_{w_{dv} \in w_d} p(w_{dv} \mid \theta_d, \beta_v)\,. \tag{4.7}$$

CTPF makes predictions using expectations under the posterior. The posterior is proportional to the the log joint of the model, and the attributes of items (words in documents) enter into the model only via the above product. The product of the probability of words in a document is invariant to a reordering of the words in the document, and therefore CTPF is permutation-invariant.

Word embedding models (Mikolov et al., 2013) can be used as recommendation models if the embeddings are learned using a modified context window. For an item with attributes x_m, let the context window for attribute $j \in x_m$ be the set of other attributes of the same item $j' \in x_m : j' \neq j$. To recommend items using this model of attributes β_j for $j \in V$, item embeddings are computed as the average of their attribute embeddings. Users are represented as the average of the embed-

dings of the items they consume, and recommendation is performed using the cosine similarity of user and item embeddings. This is a permutation-invariant model, as the output of the model depends on the sum of attribute embeddings (summation is invariant to permutation).

StarSpace is also an embedding model and represents users as a sum over a user's consumed items' attribute embeddings (there is no explicit user embedding). In contrast to the word embedding model, StarSpace is trained on a classification objective with negative samples drawn from the the set of items (Wu et al., 2018). As model predictions depend on sums of attributes, StarSpace is a permutation-invariant recommendation model.

We next consider LightFM (Kula, 2015), a permutation-invariant recommendation model. We show that if the Bayesian Personalized Ranking (BPR) objective (Rendle et al., 2009) is used, LightFM is an instance of RFS. [1] Although the BPR objective is designed for ranking, models trained with it can be used to construct classifiers. The BPR objective is

$$\log \sigma \left(f(u, x_m; \boldsymbol{\gamma}) - f(u, x_k; \boldsymbol{\gamma}) \right),$$

where m corresponds to a positive label $y_{um} = 1$, k corresponds to a negative label $y_{uk} = 0$, and f is parameterized as in RFS (Kula, 2015). A ranking function f optimizes the BPR objective if $f \to \infty$ for the positive example and f is constant for the negative example; or, if f is constant for the positive example and $f \to -\infty$ for the negative example. In either case, a constant can be added to yield a perfect classifier from the ranking function f (positive examples are ranked higher than negative examples in the optimal ranking, so there exists such a constant). That we can construct a classifier from the BPR objective means that Proposition 1 applies: permutation-invariant models such as LightFM, trained with the BPR objective, are instances of the RFS class of recommendation models.

The regression function f in RFS can also be parameterized using a recurrent network, as in Bansal et al. (2016). Such a recommendation model can be made permutation-invariant if averaged over permutations of attributes fed to the network. Attributes are treated as a sequence and the marginalization is over these permutations,

$$p(y_{um} = 1 \mid x_m) = \frac{1}{|\pi(x_m)|} \sum_{\pi \in \pi(x_m)} \sigma\left(\phi(\theta_u, \{\beta_{\pi(1)}, \ldots, \beta_{\pi(J)}\})\right) . \tag{4.8}$$

[1] The LightFM paper (Kula, 2015) uses a logistic objective to which Proposition 1 applies. LightFM with the BPR objective is unpublished but implemented in code released by the author. For completeness, we studied LightFM with both objectives to ensure its performance is equivalent to RFS when the BPR objective is used.

Here β_j are attribute embeddings, $\pi(x_m)$ denotes the set of all permutations of the attributes x_m, and ϕ is the output of a recurrent neural network architecture (Bansal et al., 2016) projected to a scalar.

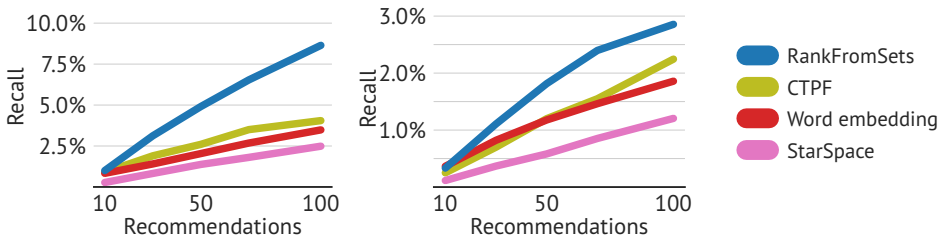
4.4 Empirical Study

We study RFS on two datasets and tasks. The first data consists of researcher reading behavior from the arXiv; the semi-synthetic task is to recommend documents to scientists. The second is crowdsourced food consumption data from a diet tracking app, and the task is meal recommendation. On both benchmarks, models in the RFS class outperform several baseline methods. The permutation-invariant models we compare to are described in Section 4.3, and the hyperparameters used are described in Section 4.7.1. To show the relative ease of implementation of RFS we give example code in Section 4.7.5.[2]

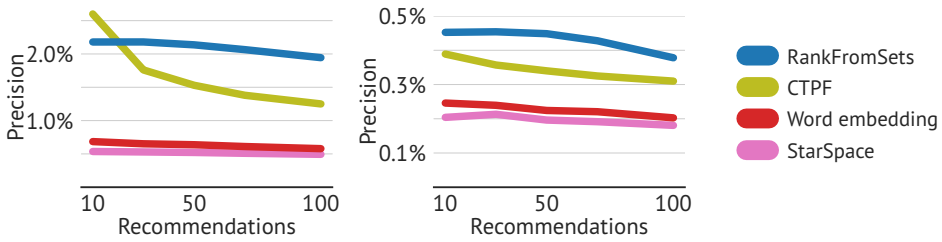
Recommending Research Papers. We benchmark RFS on data of scientists reading research papers on the arXiv, where the goal is to recommend papers to scientists. This is a semi-synthetic task: it uses real-world data, but the item side information (article abstracts) is not set-valued. Nevertheless, document recommendation is a standard benchmark to study whether RFS performs well in settings outside its target purview of meal recommendation. The arXiv data represents one year of usage (2012) and consists of 65k users, 636k preprints, and 7.6M clicks. For evaluation, we match (Gopalan et al., 2014), using the same test and validation splits and the same set of held-out 10k users. As in (Gopalan et al., 2014) we compute precision in addition to recall. The held-out validation and test splits each consist of 20% of the clicks and 1% of the documents. In-matrix documents refer to documents that have clicks in the training data, while out-matrix or cold-start documents have no previous clicks.

Figure 4.2 shows that models in the RFS class outperform others. RFS with the inner product parameterization or LightFM with the BPR objective have identical performance (as we showed, the BPR objective yields a classifier equivalent to RFS). These RFS models outperform CTPF in terms of in-matrix recall by over 90%. RFS models also improve over CTPF in terms of out-matrix recall, out-matrix precision, and in-matrix precision (for the latter, only when the number of recommendations is greater than 30). The word embedding model performs comparably to CTPF in terms of recall, and performs worse in terms of precision. Recurrent neural network recommendation models were implemented following Bansal et al. (2016) and given access to full sequence information, unlike RFS. (The permutation-marginalized version of these models in Equation (4.8) is eval-

[2]Full source code is available at https://github.com/altosaar/rankfromsets for reproducibility.



(a) Recall for in-matrix (left) and out-matrix (right) documents.



(b) Precision for in-matrix (left) and out-matrix (right) documents.

Figure 4.2: RANKFROMSETS outperforms collaborative topic Poisson factorization (CTPF) (Gopalan et al., 2014) and other models on recommending arXiv papers to scientists. The items are documents and the attributes are the unique words in the abstracts. Recommendation performance is evaluated using both precision and recall to match the evaluation in (Gopalan et al., 2014). The metrics are reported on training (in-matrix) documents and cold-start (out-matrix) documents with no clicks in the training set. All GRU and LSTM-based models in Bansal et al. (2016) performed an order of magnitude worse, and these results are omitted (training details are in Section 4.7.1).

uated on meal recommendation where the order of foods in a meal does not carry information.) The training details for the recurrent neural networks are in Section 4.7.1, but their performance was an order of magnitude worse than the other methods and these results are omitted. The RFS regression function used is in Equation (4.2); the other parameterizations did not fit in GPU memory.

Qualitatively, RFS reveals patterns in usage of the arXiv. Figure 4.1 is a dimensionality-reduced plot of the user embeddings that reveals connections between fields of study. Scientists who focus on high energy physics, `hep`, neighbor specialists in differential geometry, `math.DG`; these areas share techniques. Machine learning researchers (`stat.ML` readers) neighbor statisticians (`math.ST` readers), highlighting the close connection between these fields. Plots for document embeddings show similar patterns. This illustrates how RFS captures rich patterns of interaction between users and items, while benefitting from information in the item attributes.

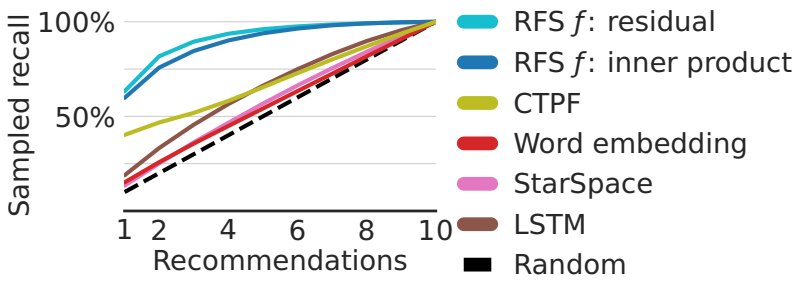


Figure 4.3: RANKFROMSETS models outperform competitors in meal recommendation in terms of sampled recall computed using Equation (4.6). Comparison models are described in Section 4.3 (see Section 4.7.1 for hyperparameters). The RFS regression functions f are defined in Equations (4.2) to (4.4) for the inner product, neural network, and residual models, respectively.

This experiment in recommending research papers also highlights a trade-off in computational budget and desired performance in recommender systems. As described in Section 4.7.2, the recurrent neural network recommendation models in Bansal et al. (2016) did not perform well with the computational budget allocated for all methods (one day of compute on Tesla P100 GPUs). In further experiments, the performance improved marginally with a larger computational budget of several days. Further research in this domain might compare to transformer models (Vaswani et al., 2017; Devlin et al., 2019). Transformers preserve sequence information, unlike RFS, although they require large computational budgets to make accurate predictions. This means transformer-based methods may present a different trade-off in recommendation performance than the recurrent neural networks we evaluated in this task.

Recommending Meals. We evaluate RFS on data collected from the LoseIt! diet tracking app. This app enables users to track their food intake to eat healthy. We use a year's worth of data from 55k active users. This corresponds to 16M meals, where each meal is comprised of a subset of 3M foods. To preprocess, we filter the vocabulary by keeping words that occur at least 20 times in the food names, resulting in 9963 words. A meal is represented as the union of the sets of words occurring in the food names. For evaluation, 1% of the items (meals) are held out for evaluating validation and test performance respectively. We evaluate models using SampledRecall@K with $K = 10$.

Figure 4.3 shows the sampled recall: models in the RFS class outperform others, such as permutation-marginalized recurrent neural networks and word embedding models. The residual RFS model outperforms the RFS inner product parameterization (and the equivalent LightFM model trained on the BPR objective).

The code released with Gopalan et al. (2014) or Wang and Blei (2011) did not scale to this size of data, despite sufficient computing resources. This experiment further verifies Proposition 1: RFS models can maximize recall.

Qualitatively, RFS learns an interpretable representation of items, as shown by nearest neighbors of meals in Table 4.2. In this table, we display breakfast, lunch, and dinner meals, alongside their nearest neighbors. We find that the nearest neighbors are also breakfast, lunch, and dinner meals respectively, showing that the attribute embeddings learned by the model can be used to explore qualitative patterns in the learned latent space.

Model	Attributes	Implicit	Scalable	Invariant	Evaluation
RANKFROMSETS	✔	✔	✔	✔	✔
CTPF, Gopalan et al. (2014)	✔	✔		✔	
StarSpace, Wu et al. (2018)	✔	✔	✔	✔	
LightFM, Kula (2015)	✔	✔	✔	✔	
BPR, Rendle et al. (2009)		✔			✔
Wang and Blei (2011)	✔	✔		✔	
Lian et al. (2018)			✔		
Dong et al. (2017)	✔			✔	
Chen et al. (2017)	✔		✔		
Bansal et al. (2016)	✔	✔			
Xu et al. (2017)	✔		✔	✔	
Shi et al. (2012b)		✔			✔
Chen and Rijke (2018)	✔	✔		✔	
Liu et al. (2014)	✔			✔	✔
Cao et al. (2017)	✔				✔
Okura et al. (2017)	✔		✔		✔

Table 4.3: RANKFROMSETS recommends items using attributes, and is trained to maximize the evaluation metric of recall. Most methods we highlight leverage item attributes (Attributes); some require data in addition to the implicit feedback data of user-item interactions (Implicit). Few methods are scalable, as most models that use item side information require learning parameters for every item. Some models are invariant to permutation of the attributes (Invariant), and some enjoy a loss function that is connected to a recommender performance metric (Evaluation).

4.5 Related Work

We survey food recommender systems and recommendation models, focusing on models that leverage content information and scale to large numbers of users, items, and attributes.

Existing food recommendation systems focus on healthy recommendation (Trattner and Elsweiler, 2019a; Freyne et al., 2011; Khan et al., 2019; Yang et al., 2017), while RFS focuses on the scalability challenge of meal recommendation. After training a recommendation model, it is possible to filter the recommendations by nutritional information to nudge users towards healthier eating habits (Elsweiler et al., 2017); such approaches can be used to include nutritional information into RFS recommendations. When data is used in food recommender systems, it is usually recipe data (Trattner and Elsweiler, 2018); RFS is designed to recommend meals using crowdsourced food consumption data which may accurately reflect user behavior (Trattner and Elsweiler, 2019b).

We highlight several themes in research on recommendation models. We describe recommendation models that incorporate side information, models that recommend through classification, and models that optimize proxies of ranking metrics. This related work is summarized in Table 4.3. We focus on deep learning-based and matrix factorization methods to include side information in recommendation models. Item side information can be modeled with deep representations or can be included in content-based matrix factorization models as an additional matrix. Some deep learning approaches scale to large datasets, but may not have objective functions tied to evaluation metrics, or may require data beyond user-item interactions (Okura et al., 2017). Content-based matrix factorization methods require learning parameters for every item, and do not scale to data with large numbers of items (Wang and Blei, 2011; Gopalan et al., 2014), whereas RFS scales and is tied to evaluation.

Deep Representations of Side Information. Deep learning-based recommendation models incorporate side information in multiple ways (Zhang et al., 2019). For example, items that have words as attributes can be represented using neural networks (Bansal et al., 2016; Chen and Rijke, 2018) or embeddings (Wu et al., 2018). RFS uses both embeddings and deep learning techniques such as residual networks (He et al., 2016) to include side information. Lian et al. (2018) use an attention mechanism to weight recommendations according to available item and user side information, and Dong et al. (2017) use denoising autoencoders to model side information in a deep recommendation model, but these methods require fitting parameters for every item and hence cannot scale. An example of a more efficient approach is the method in Chen et al. (2017), where embeddings are jointly learned for users, items, and item text for recommendation, but this method focuses on unsupervised pre-training of text representations. RFS is

complementary to such approaches, as the user, attribute, and item embeddings can be initialized using pre-training. Deep structured semantic models are designed for document retrieval given query words (Huang et al., 2013; Palangi et al., 2016); it is unclear how to use this setup for recommending items with set-valued side information to users. There are several examples of 'tag-aware' or 'tag-based' deep recommendation models (Liang et al., 2018b; Zuo et al., 2016), such as Xu et al. (2017), which focuses on data where users and items have different attributes and uses autoencoders to learn user, item, and attribute representations. Xu et al. (2017) uses a cosine similarity-based objective function which is not tied to a metric used to evaluate recommendation performance, whereas RFS is tied to recall as shown in Proposition 1.

Recommendation via Classification. The framing of recommendation as classification has been around for a long time (Basu et al., 1998), and several works build deep learning-based classifiers for recommendation (Covington et al., 2016; Cheng et al., 2016; Guo et al., 2017; He et al., 2017). Covington et al. (2016) focus on scalable inclusion of user and item attributes for video recommendation, Cheng et al. (2016) jointly train generalized linear models and deep neural networks for recommendation, while Guo et al. (2017) use factorization machines to learn high- and low-order interactions of features. Our work is complementary to these approaches: RFS focuses on scalable inclusion of set-valued side information, and provides theoretical undergirding to these recommendation models. We connect such models that rely on classification to optimal recall in Proposition 1. And if a specific architecture developed in these works is a permutation-invariant recommendation model, we proved that RFS is a universal function approximator (Proposition 2). So if performance is measured by recall, an RFS model can converge to an optimal recommender.

Matrix Factorization with Side Information. While matrix factorization methods perform well in recommending items that have consumption data in the training set (Hu et al., 2008; Liang et al., 2016), they cannot recommend items that have not been consumed in the training data. Including side information in matrix factorization enables recommendation of these items with no consumption data. Shi et al. (2014) survey several matrix factorization methods that leverage side information. Gopalan et al. (2014) develop a Bayesian matrix factorization model for recommending items based on side information in the form of words in documents, and we compare RFS to this method in Section 4.4. Wang and Blei (2011) develop a regression model that uses a topic model to incorporate side information into recommendations. There are also several 'tag-based' or 'tag-aware' content-based matrix factorization models (Zhen et al., 2009; Loepp et al., 2019; Bogers, 2018). Such content-based matrix factorization methods maximize the conditional log-likelihood of the data (or a bound on the log-likelihood); optimizing these objective functions may not optimize an

evaluation metric. These methods are not scalable to large numbers of items as they require learning unique parameters for every item. Specifically, such content-based matrix factorization methods require learning a matrix that has a row for every item. For items with attributes, it is often infeasible to store this matrix in memory or exploit efficient coordinate ascent optimization schemes that require processing this entire matrix. RFS, however, is designed to scale to tens of millions of items, as we demonstrate empirically in Section 4.4.

Learning to Rank. The learning to rank literature includes several recommendation models trained on objectives that approximate ranking-based evaluation metrics (Yu et al., 2018; Liang et al., 2018a; Rendle et al., 2009; Song et al., 2018), and some of these models include side information (Shi et al., 2012a; Shi et al., 2012b; Yuan et al., 2016; Ying et al., 2016; Cao et al., 2017; Okura et al., 2017). Such approaches can require data in addition to the user-item matrix, such as per-item parameters, or might use models whose output depends on the ordering of item attributes (making them infeasible for set-valued side information). In Section 4.4, we show that the ranking-based BPR objective function (Rendle et al., 2009; Kula, 2015) is in the RFS class, so Proposition 1 can help frame this related work. Li et al. (2016) use an objective that is in the same class as BPR, and other work bounds the BPR objective (Zhang et al., 2018); these are also examples of RFS models if a permutation-invariant architecture is specified and we study one such choice (Kula, 2015) in Section 4.4.

4.6 Discussion

The task of recommending items with attributes is difficult for several reasons. It is unclear how to incorporate set-valued side information into models that scale to large numbers of items and attributes. In addition, existing recommendation models that leverage item attributes (for example, content-based matrix factorization) are not directly tied to evaluation metrics. We developed RFS, a class of scalable recommendation models for items with attributes. Theoretically, we showed that optimizing the RFS objective optimizes recall, and that RFS can approximate permutation-invariant recommendation models including content-based matrix factorization. Empirically, models in the RFS class outperform competing models and scale to large datasets, such as our motivating problem of meal recommendation for 55k users who consume 16M meals.

How well does binary classification perform for other ranking-based recommendation metrics, such as non-discounted cumulative gain? Analyzing this question is more difficult, and we leave this to future work. For generalization theory, we conjecture that a different loss function should allow a similar proof to Proposition 1. With sufficient data, RFS can learn arbitrary distributions of users con-

suming items with attributes. But performance on finite data can vary, and developing generalization theory for RFS remains an open question.

4.7 Appendix

4.7.1 Empirical Study Hyperparameters

Experiments for RFS, LightFM, and recurrent neural network models are run on a cluster with Tesla P100 GPUs using PyTorch; all other experiments are performed on a 20-core computer.

Hyperparameters for Word Embedding Model and StarSpace. For the word embedding model and StarSpace, we use the software packages released alongside the respective papers (Bojanowski et al., 2017; Wu et al., 2018) with recommended hyperparameters, and grid search over embedding sizes of $\{128, 256, 512, 1024\}$ for both datasets.

Hyperparameters for LightFM. On both datasets, LightFM with the logistic objective reported in Kula (2015) performs poorly and we omit these results. Kula (2015) does not use the BPR objective in the paper; nevertheless, we study LightFM with the BPR objective (this variant is unpublished, yet implemented in the code released in (Kula, 2015)) and use the same hyperparameters as RFS for comparison.

4.7.2 Recommending Research Papers

Hyperparameters for RFS. We test the stochastic gradient descent algorithm with and without momentum (Sutskever et al., 2013). We use a linear learning rate decay that decays to zero in the maximum number of iterations, 200k. We perform a grid search over learning rates of $\{1, 5, 10, 15, 25\}$ and momenta of $\{0.5, 0.9, 0.95, 0.99\}$. The minibatch size is set to 2^{16}. We use a single negative sample per datapoint, sampled uniformly over the entire dataset; such corpus sampling is defined in Section 4.2. As the number of items is small relative to the larger diet tracking data, the item intercept function is simply a scalar for every item, and the item embedding function learns item embeddings. To match the hyperparameters in Gopalan et al. (2014), we set the dimensionality of user and item embeddings to 100. Evaluation is performed every 20k iterations.

Hyperparameters for Recurrent Neural Network Models. We implement the model in Bansal et al. (2016) using PyTorch and match the hyperparameters where possible. We test gated recurrent unit (GRU) cells and long short-term memory (LSTM) cells with the objective function in Equation (4.5). The model

has access to full sequence information, unlike RFS, as abstracts of research papers have meaningful sequence information (marginalizing using Equation (4.8) would destroy information and decrease performance). The attribute embedding size is fixed to 100 to match the other models and attribute embeddings are initialized to word embeddings pretrained on all 636k document abstracts as in Bansal et al. (2016), using the word embedding implementation in Bojanowski et al. (2017). The first layer of the recurrent neural network is bidirectional and of hidden size 400, the second layer is unidirectional and of hidden size 200, and dropout is used with the same settings as in Bansal et al. (2016). Evaluation is performed every 20k iterations. We grid search over learning rates of $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ with the Adam optimizer (Kingma and Ba, 2015) and batch sizes of $\{64, 128, 256, 512, 1024, 4096, 8192\}$. As evaluation is much more expensive for sequence models, we randomly select a subset of 100 users from the held-out set of 10k users. If validation performance does not improve, we reload the best parameters and optimizer states, and divide the learning rate by half. In both experiments, LSTM cells outperformed GRU cells.

4.7.3 Recommending Meals

Hyperparameters for RFS. The embedding size is set to 128. For the neural network and residual models in Equations (4.3) and (4.4) the number of hidden layers is two, and the number of hidden units is set to 256 with rectifier nonlinearities. The item embeddings $g(x_m)$, and item intercepts $h(x_m)$, are computed as the mean of learned food embeddings and intercepts, respectively. We use the RMSProp optimizer in Graves (2013) and grid search over the learning rates $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. We use a batch size of 64 and a single negative sample for every datapoint in a minibatch (batch sampling is defined in Section 4.2). Evaluation is performed every 50k iterations.

Hyperparameters for Permutation-marginalized Recurrent Neural Networks. We use the same settings as described in Section 4.7.2, but the data in this case has no sequence information so we use Equation (4.8) to average predictions of the model in Bansal et al. (2016) over permutations. Evaluation for sequence models is already prohibitive, so for every item in a minibatch we sample a single permutation of attributes to approximate the sum over permutations in Equation (4.8). We use a single negative sample per datapoint (minibatch sampling), and set the embedding and hidden state sizes to 128. We use the Adam optimizer (Kingma and Ba, 2015) and grid search over the same learning rates, learning rate decay, and batch sizes as in Section 4.7.2, with evaluation every 1k iterations.

4.7.4 Generalization Simulation Study

Proposition 2 is a universal function approximation theorem in the regime of infinite data. With finite data and a finite number of parameters, the optimal parameterization of RFS is dependent on the data-generating distribution. From Figure 4.3, the inner product RFS parameterization outperforms the neural network parameterization. We demonstrate a simulated dataset where this order is reversed, to motivate the exploration of novel architectures. Recall that observations of user-item interactions are generated by a Bernoulli distribution with logit function f. We describe a choice of logit function f that leads to the residual and deep architectures in Equations (4.3) and (4.4) outperforming the inner product architecture in Equation (4.2) in terms of predictive performance. We will release all code required to replicate this experiment.

1. **For every user** u**:** Draw user embedding $\theta_u \sim \text{Normal}(0, \mathbf{I})$.
2. **For every attribute** j**:** Draw attribute embedding $\beta_j \sim \text{Normal}(0, \mathbf{I})$.
3. **For every item** m**:**
   (a) Draw item topics $\theta_m \sim \text{Dirichlet}(\alpha)$
   (b) Draw number of item attributes $M \sim \text{Poisson}(\lambda)$
   (c) Draw nonzero item attributes $x_m \sim \text{Multinomial}(M, \theta_m)$.
4. **For every user, item:** $y_{um} \sim \text{Bernoulli}\left(y_{um}; \sigma(f(\theta_u, x_m)\right)$.

The logit function f is the square kernel:

$$f(\theta_u, x_m) = \left(\theta_u^\top \frac{1}{|x_m|} \sum_{j \in x_m} \beta_j\right)^2.$$

The output of f is standardized across users and centered at 7 to achieve sparse user-item observations.

For this simulation study, we set the Dirichlet parameter to be $\alpha = 0.01$ and the Poisson rate to be $\lambda = 20$. We generate data for 1k users, 5k item attributes, 30k items, and hold out 100 users for each of the validation and test sets. The embedding sizes are fixed to 100, and for parameterizations with neural networks two hidden layers are used with rectifier nonlinearities. The hidden size of models with neural networks is chosen so the total number of parameters matches the number of parameters in the inner product model. We fix the momentum to 0.9 (Sutskever et al., 2013) and grid search over stochastic gradient descent learning rates of $10, 1, 0.1, 0.01$ and over two learning rate decay schedules. The first linear learning rate decay goes to zero over 100k iterations, while the second divides the learning rate by 10 if the validation in-matrix recall does not improve (evalu-

	Inner product	Deep	Residual
Recall	0.29 ± 0.15	0.32 ± 0.14	0.33 ± 0.18

Table 4.4: A simulation study demonstrating that the choice of parameterization of RANKFROMSETS is data-dependent. We report the in-matrix recall averaged over 100 users, over 30 replications of the simulation. The residual model in Equation (4.4) outperforms the deep model in Equation (4.3) and the inner product model in Equation (4.2).

ation is performed every 500 iterations). We run the grid search on one instance of data generated from this model. We regenerate data 30 times and average results over these synthetic datasets, using the best performing hyperparameters for each model trained on the first instance.

The results in Table 4.4 demonstrate that the residual model outperforms both the deep and inner product architectures for data generated by the above generative process. This shows that the choice of architecture in RFS is data-dependent and leads to considering when a RFS recommendation model supports generalization. To ensure that the model does not overfit as new users or items are included in the training data, we need to compare the number of parameters to the number of datapoints. A model with parameters the size of the training data can overfit by memorizing the training data. For generalization to be possible, overfitting can be avoided if the number of parameters grows slower than the size of the data. The technical backing for this comes from asymptotic statistics and the concept of sieved likelihoods. Specifically, the maximum likelihood estimation procedure with the objective function in Equation (4.5) can be replaced by maximization of a sieved likelihood function. The 'sieve' refers to filtering information as the number of parameters (in this case, the number of parameters in user and item representations) grows with the number of observations. The sieved likelihood function enables the analysis of asymptotic behavior as the number of users grows $U \to \infty$ and the number of items grows $I \to \infty$. An example of a technique to grow the number of parameters in a way that supports generalization is given in Chapter 25 of Vaart (1998).

4.7.5 Code

We give an example implementation of RANKFROMSETS with the inner product regression function in Equation (4.2) in python with the PyTorch package. This implementation is easy to port to new applications and achieves state-of-the-art results in Section 4.4.

```
import torch
import data
class InnerProduct(torch.nn.Module):
  def __init__(self, n_users, n_items, n_attr, emb_size):
    super().__init__()
    self.user_embeddings = torch.nn.Embedding(n_users, emb_size)
    self.attribute_emb = torch.nn.EmbeddingBag(n_attr, emb_size)
    self.item_embeddings = torch.nn.Embedding(n_items, emb_size)
    self.intercepts = torch.nn.Embedding(n_items, 1)
  def forward(self, users, items, item_attributes, offsets):
    user_emb = self.user_embeddings(users)
    attr_emb = self.attribute_emb(item_attributes, offsets)
    item_emb = self.item_embeddings(items)
    logits = (user_emb * (attr_emb + item_emb)).sum(-1)
    return logits + self.intercepts(items).squeeze()
train = data.load(batch_size=2 ** 16) # negative labels in last half of every
    batch
model = InnerProduct(train.n_users, train.n_items, train.n_attr, 100)
optim = torch.optim.SGD(model.parameters(), learning_rate=15.0)
loss = torch.nn.BCEWithLogitsLoss()
labels = (torch.arange(2 ** 16) < (2 ** 16 / 2)).float()
for batch in train:
  model.zero_grad()
  logits = model(*batch)
  L = loss(logits, labels)
  L.backward()
  optim.step()
```

Chapter 5

Proximity Variational Inference

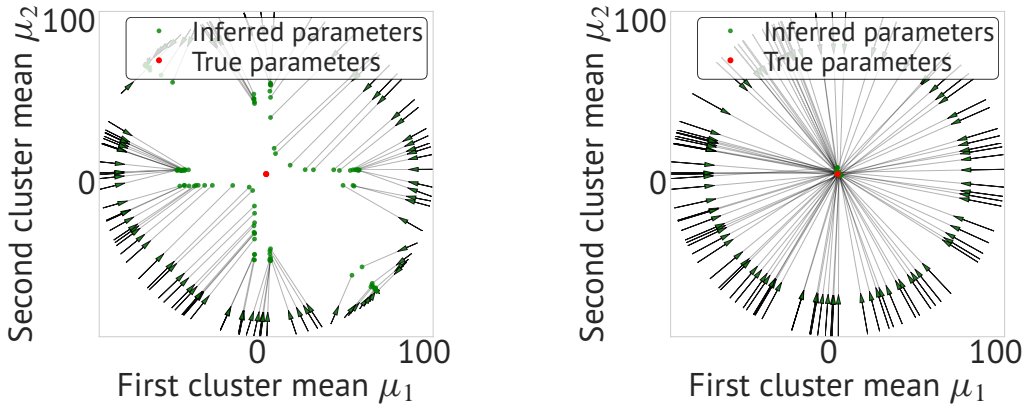
Previously, Chapters 3 and 4 highlighted how consideration of the problem structure enabled efficient probabilistic modeling solutions to applied questions in physics and recommender systems. Can problem structure be of use at a higher level, in an inference algorithm that can be reused across probability models? We develop proximity variational inference in this chapter, which enables variational inference to leverage information about variational approximations during optimization to improve the accuracy of inference.

5.1 Introduction

Variational inference (VI) is a powerful method for probabilistic modeling. VI uses optimization to approximate difficult-to-compute conditional distributions (Jordan et al., 1999). In its modern incarnation, it has scaled Bayesian computation to large data sets (Hoffman et al., 2013), generalized to large classes of models (Kingma and Welling, 2014; Ranganath et al., 2014; Rezende and Mohamed, 2015), and has been deployed as a computational engine in probabilistic programming systems (Mansinghka et al., 2014; Kucukelbir et al., 2015; Tran et al., 2016).

Despite these significant advances, however, VI has drawbacks. For one, it tries to iteratively solve a difficult nonconvex optimization problem and its objective contains many local optima. Consequently, VI is sensitive to initialization and easily gets stuck in a poor solution. We develop a new optimization method for VI and show that it finds better optima.

Consider a probability model $p(\mathbf{z}, \mathbf{x})$ and the goal of calculating the posterior $p(\mathbf{z} \mid \mathbf{x})$. The idea behind VI is to posit a family of distributions over the hidden variables $q(\mathbf{z}; \boldsymbol{\lambda})$ and then fit the variational parameters $\boldsymbol{\lambda}$ to minimize the Kullback-



(a) Variational Inference **(b)** Proximity VI, Algorithm 2

Figure 5.1: Proximity variational inference is robust to bad initialization. We study a Bernoulli factor model. Model parameters are randomly initialized on a ring around the known true parameters (in red) used to generate the data. The arrows start at these parameter initializations and end at the final parameter estimates (shown as green dots). **(a)** Variational inference with gradient ascent suffers from multiple local optima and cannot reliably recover the truth. **(b)** PVI with an entropy proximity statistic reliably infers the true parameters using Algorithm 2.

Leibler (KL) divergence between the approximating family and the exact posterior, $\mathrm{KL}(q(\mathbf{z};\boldsymbol{\lambda})||p(\mathbf{z} \mid \mathbf{x}))$. The KL is not tractable so VI optimizes a proxy. That proxy is the evidence lower bound (ELBO),

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}[\log p(\mathbf{z}, \mathbf{x})] - \mathbb{E}[\log q(\mathbf{z};\boldsymbol{\lambda})], \tag{5.1}$$

where expectations are taken with respect to $q(\mathbf{z};\boldsymbol{\lambda})$. Maximizing the ELBO with respect to $\boldsymbol{\lambda}$ is equivalent to minimizing the KL divergence. The issues around VI stem from the ELBO and the iterative algorithms used to optimize it. When the algorithm zeroes (or nearly zeroes) some of the support of $q(\mathbf{z};\boldsymbol{\lambda})$, it becomes hard to later "escape," i.e., to add support for the configurations of the latent variables that have been assigned zero probability (MacKay, 2003; Burda et al., 2015). This leads to poor local optima and to sensitivity to the starting point, where a misguided initialization will lead to such optima. These problems happen in both gradient-based and coordinate ascent methods. We address these issues with proximity variational inference (PVI), a variational inference algorithm that is specifically designed to avoid poor local optima and to be robust to different initializations.

PVI builds on the proximity perspective of gradient ascent. The proximity perspective views each step of gradient ascent as a constrained minimization of a Taylor expansion of the objective around the previous step's parameter (Spall,

2003; Boyd and Vandenberghe, 2004). The constraint, a *proximity constraint*, enforces that the next point should be inside a Euclidean ball of the previous. The step size relates to the size of that ball. We construct PVI by questioning whether such a Euclidan distance-based constraint is appropriate, and whether other notions of proximity may be useful in constraining gradient ascent steps.

In VI, a constraint on the Euclidean distance means that all dimensions of the variational parameters are equally constrained. We posit that this leads to problems; some dimensions need more regularization than others. For example, consider a variational distribution that is Gaussian. A good optimization will change the variance parameter more slowly than the mean parameter to prevent rapid changes to the support. The Euclidean constraint cannot enforce this. Furthermore, the constraints enforced by gradient descent are transient; the constraints are relative to the previous iterate—one poor move during the optimization can lead to permanent optimization problems.

To this end, PVI uses proximity constraints that are more meaningful to variational inference and to optimization of probability parameters. A constraint is defined using a proximity statistic and distance function. As one example, we consider a constraint based on the entropy proximity statistic. This limits the change in entropy of the variational approximation from one step to the next. Consider again a Gaussian approximation. The entropy is a function of the variance alone and thus the entropy constraint counters the pathologies induced by the Euclidean proximity constraint. We also study constraints built from other proximity statistics, such as those that penalize the rapid changes in the mean and variance of the approximate posterior.

Figure 5.1 provides an illustration of the advantages of PVI. Our goal is to estimate the parameters of a factor analysis model with variational inference, i.e., using the posterior expectation under a fitted variational distribution. We run variational inference 100 times, each time initializing the estimates (the model parameters) to a different position on a ring around the truth.

In the figure, red points indicate the true value. The start locations of the green arrows indicate the initialized estimates. Green points indicate the final estimates, after optimizing from the initial points. Panel (a) shows that optimizing the standard ELBO with gradients leads to poor local optima and misplaced estimates. Panel (b) illustrates that regardless of the initialization, PVI with an entropy proximity statistic finds estimates that are close to the true value.

The rest of this chapter is organized as follows. Section 5.2 reviews variational inference and the proximity perspective of gradient optimization. Section 5.3 derives PVI; we develop four proximity constraints and two algorithms for optimizing the ELBO. We study four models in Section 5.4: a Bernoulli factor model, a sigmoid belief network (Mnih and Rezende, 2016), a variational autoencoder (Kingma and Welling, 2014; Rezende et al., 2014), and a deep exponential

family model of text (Ranganath et al., 2015). PVI outperforms classical methods for variational inference.

Related Work. Recent work has proposed several related algorithms. Khan et al. (2015) and Theis and Hoffman (2015) develop a method to optimize the ELBO that imposes a soft limit on the change in KL of consecutive variational approximations. This is equivalent to PVI with identity proximity statistics and a KL distance function. Khan et al. (2016) extend both prior works to other divergence functions. Their general approach is equivalent to PVI identity proximity statistics and distance functions given by strongly-convex divergences. Compared to prior work, PVI generalizes to a broader class of proximity statistics. We develop proximity statistics based on entropy, KL, orthogonal weight matrices, and the mean and variance of the variational approximation.

The problem of model pruning in variational inference has also been studied and analytically solved in a matrix factorization model in Nakajima et al. (2013)—this method is model-specific, whereas PVI applies to a much broader class of latent variable models. Finally, deterministic annealing (Katahira et al., 2008) consists of adding a temperature parameter to the entropy term in the ELBO that initialized to a large value then annealed to unity during inference. This is similar to PVI with the entropy proximity statistic which keeps the entropy stable across iterations. Deterministic annealing enforces global penalization of low-entropy configurations of latent variables rather than the smooth constraint used in PVI, and cannot accommodate the range of proximity statistics we design in this work.

5.2 Variational Inference

Consider a model $p(\mathbf{x}, \mathbf{z})$, where $\mathbf{x}$ is the observed data and $\mathbf{z}$ are the latent variables. As described in Section 5.1, VI posits an approximating family $q(\mathbf{z}; \boldsymbol{\lambda})$ and maximizes the ELBO in Equation (5.1). Solving this optimization is equivalent to finding the variational approximation that minimizes KL divergence to the exact posterior (Jordan et al., 1999; Wainwright and Jordan, 2008).

5.2.1 Gradient Ascent has Euclidean Proximity

Gradient ascent maximizes the ELBO by repeatedly following its gradient. One view of this algorithm is that it repeatedly maximizes the linearized ELBO subject to a proximity constraint on the current variational parameter (Spall, 2003). The name 'proximity' comes from constraining subsequent parameters to remain close in the proximity statistic. In gradient ascent, the proximity statistic for the variational parameters is the identity function $f(\boldsymbol{\lambda}) = \boldsymbol{\lambda}$, and the distance function is the square difference.

Let λ_t be the variational parameters at iteration t and ρ be a constant. To obtain the next iterate λ_{t+1}, gradient ascent maximizes the linearized ELBO,

$$U(\lambda_{t+1}) = \mathcal{L}(\lambda_t) + \nabla\mathcal{L}(\lambda_t)^\top(\lambda_{t+1} - \lambda_t) - \frac{1}{2\rho}(\lambda_{t+1} - \lambda_t)^\top(\lambda_{t+1} - \lambda_t). \tag{5.2}$$

Specifically, this is the linearized ELBO around λ_t, subject to λ_{t+1} being close to λ_t in squared Euclidean distance.

Finding the λ_{t+1} which maximizes Equation (5.2) yields

$$\lambda_{t+1} = \lambda_t + \rho\nabla\mathcal{L}(\lambda_t). \tag{5.3}$$

This is the familiar gradient ascent update with a step size of ρ. The step size ρ controls the radius of the Euclidean ball which demarcates valid next steps for the parameters. Note that the Euclidean constraint between subsequent iterates is implicit in all gradient ascent algorithms.

5.2.2 An Example where Variational Inference Fails

We study a setting where variational inference suffers from poor local optima. Consider a factor model, with Bernoulli latent variables and Gaussian likelihood:

$$z_{ik} \sim \text{Bernoulli}(\pi) \tag{5.4}$$
$$x_i \sim \text{Gaussian}\left(\mu = \textstyle\sum_k z_{ik}\mu_k, \sigma^2 = 1\right). \tag{5.5}$$

This is a "feature" model of real-valued data x; when one of the features is on (i.e., $z_{ik} = 1$), the ith mean shifts according the that feature's mean parameter (i.e., μ_k). Thus the binary latent variables z_{ik} control which cluster means μ_k contribute to the distribution of x_i.

The Bernoulli prior is parametrized by π; we choose a Bernoulli approximate posterior $q(z_k; \lambda_k) = \text{Bernoulli}(\lambda_k)$. A common approach to VI is coordinate ascent (Bishop, 2006), where we iteratively optimize each variational parameter. The optimal variational parameter for z_{ik} is

$$\lambda_{ik} \propto \exp\left\{\mathbb{E}_{-z_{ik}}\left[-\frac{1}{2\sigma^2}(x_i - \sum_j z_{ij}\mu_j)^2\right]\right\}. \tag{5.6}$$

Algorithm 1: Proximity Variational Inference

Input: Initial parameters λ_0, proximity statistic $f(\lambda)$, distance function d
Output: Parameters λ of variational $q(\lambda)$ that maximize the ELBO objective
while $\mathcal{L}$ *not converged* **do**
  $\lambda_{t+1} \leftarrow \lambda_t + \text{Noise}$
  while U *not converged* **do**
    Update $\lambda_{t+1} \leftarrow \lambda_{t+1} + \rho \nabla_\lambda U(\lambda_{t+1})$
  end
  $\lambda_t \leftarrow \lambda_{t+1}$
end
return λ

We can use this update in a variational expectation-maximization setting. The corresponding gradient for μ_k is

$$\frac{\partial \mathcal{L}}{\partial \mu_k} = -\frac{1}{\sigma^2} \sum_i \left(-x_i \lambda_{ik} + \lambda_{ik} \mu_k + \lambda_{ik} \sum_{j \neq k} \lambda_{ij} \mu_j \right). \tag{5.7}$$

Meditating on these two equations reveals a deficiency in mean field variational inference. First, if the mean parameters μ are initialized far from the data then $q^*(z_{ik} = 1)$ will be very small. The reason is in Equation (5.6), where the squared difference between the data x_i and the expected cluster mean will be large and negative. Second, when the probability of cluster assignment is close to zero, λ_{ik} is small. This means that the norm of the gradient in Equation (5.7) will be small. Consequently, learning will be slow. We see this phenomenon in Figure 5.1 (a). Variational inference arrives at poor local optima and does not recover the correct cluster means.

5.3 Proximity Variational Inference

We now develop PVI, a variational inference method that is robust to initialization and can consistently reach good local optima (Section 5.3.1). PVI alters the notion of proximity. We further restrict the iterates of the variational parameters by deforming the Euclidean ball implicit in classical gradient ascent. This is done by choosing proximity statistics that are not the identity function, and distance functions that are different than the square difference. These design choices help guide the variational parameters away from poor local optima (Section 5.3.2). One drawback of the proximity perspective is that it requires an inner optimization at each step of the outer optimization. We use a Taylor expansion to avoid this computational burden (Section 5.3.3).

Algorithm 2: Fast Proximity Variational Inference

Input: Initial parameters λ_0, adaptive learning rate optimizer, proximity statistic $f(\lambda)$, distance d

Output: Parameters λ of the variational distribution $q(\lambda)$ that maximize the ELBO objective

while $\mathcal{L}_{proximity}$ *not converged* **do**

$\lambda_{t+1} = \lambda_t + \rho(\nabla\mathcal{L}(\lambda_t) - k \cdot (\nabla d(f(\tilde{\lambda}), f(\lambda_t))\nabla f(\lambda_t))$.

$\tilde{\lambda} = \alpha\tilde{\lambda} + (1-\alpha)\lambda_{t+1}$

end

return λ

5.3.1 Proximity Constraints for Variational Inference

PVI enriches the proximity constraint in gradient ascent of the ELBO. We want to develop constraints on the iterates λ_t to counter the pathologies of standard variational inference.

Let $f(\cdot)$ be a *proximity statistic*, and let d be a differentiable distance function that measures distance between proximity statistic iterates. A *proximity constraint* is the combination of a distance function d applied to a proximity statistic f. (Recall that in classical gradient ascent, the Euclidean proximity constraint uses the identity as the proximity statistic and the square difference as the distance.) Let k be the scalar magnitude of the proximity constraint. We define the proximity update equation for the variational parameters λ_{t+1} to be

$$U(\lambda_{t+1}) = \mathcal{L}(\lambda_t) + \nabla\mathcal{L}(\lambda_t)^\top(\lambda_{t+1} - \lambda_t) - \frac{1}{2\rho}(\lambda_{t+1} - \lambda_t)^\top(\lambda_{t+1} - \lambda_t) - k \cdot d(f(\tilde{\lambda}), f(\lambda_{t+1})), \tag{5.8}$$

where $\tilde{\lambda}$ is the variational parameter to which we are measuring closeness. In gradient ascent, this is the previous parameter $\tilde{\lambda} = \lambda_t$, but our construction can enforce proximity to more than just the previous parameters. For example, we can set $\tilde{\lambda}$ to be an exponential moving average[1] — this adds robustness to one-update optimization missteps.

The next parameters are found by maximizing Equation (5.8). This enforces that the variational parameters between updates will remain close in the proximity statistic $f(\lambda)$. For example, $f(\lambda)$ might be the entropy of the variational approximation; this can avoid zeroing out some of its support. This procedure is detailed in Algorithm 1. The magnitude k of the constraint is a hyperparameter. The inner optimization loop optimizes the update equation U at each step.

[1] The exponential moving average of a variable λ is denoted $\tilde{\lambda}$ and is updated according to $\tilde{\lambda} \leftarrow \alpha\tilde{\lambda} + (1-\alpha)\lambda$, where α is a decay close to one.

5.3.2 Proximity Statistics for Variational Inference

We describe four proximity statistics $f(\lambda)$ appropriate for variational inference. Together with a distance function, these proximity statistics yield proximity constraints. (We study them in Section 5.4.)

Entropy Proximity Statistic. Consider a constraint built from the entropy proximity statistic, $f(\lambda) = \mathrm{H}(q(\mathbf{z};\lambda))$. Informally, the entropy measures the amount of randomness present in a distribution. High entropy distributions look more uniform across their support; low entropy distributions are peaky.

Using the entropy in Equation (5.8) constrains all updates to have entropy close to their previous update. When the variational distributions are initialized with large entropy, this statistic balances the "zero-forcing" issue that is intrinsic to variational inference (MacKay, 2003). Figure 5.1 demonstrates how PVI with an entropy constraint can correct this pathology.

KL Proximity Statistic. We can rewrite the ELBO to include the KL between the approximate posterior and the prior (Kingma and Welling, 2014),

$$\mathcal{L}(\lambda) = \mathbb{E}[\log p(\mathbf{x} \mid \mathbf{z})] - \mathrm{KL}(q(\mathbf{z} \mid \mathbf{x};\lambda)||p(\mathbf{z})).$$

Flexible models tend to minimize the KL divergence too quickly and get stuck in poor optima (Bowman et al., 2016). The choice of KL as a proximity statistic prevents the KL from being optimized too quickly relative to the likelihood.

Mean/Variance Proximity Statistic. A common theme in the problems with variational inference is that the bulk of the probability mass can quickly move to a point where that dimension will no longer be explored (Burda et al., 2015). One way to address this is to restrict the mean and variance of the variational approximation to change slowly during optimization. This constraint only allows higher order moments of the variational approximation to change rapidly. The mean $\mu = \mathbb{E}_{q(\mathbf{z};\lambda)}[\mathbf{z}]$ and variance $\mathrm{Var}(\mathbf{z}) = \mathbb{E}_{q(\mathbf{z};\lambda)}[(\mathbf{z}-\mu)^2]$ are the statistics $f(\lambda)$ we constrain.

Orthogonal Proximity Statistic. In Bayesian deep learning models such as the variational autoencoder (Kingma and Welling, 2014; Rezende et al., 2014) it is common to parametrize the variational distribution with a neural network. Orthogonal weight matrices make optimization easier in neural networks by allowing gradients to propagate further (Saxe et al., 2014). We can exploit this fact to design an orthogonal proximity statistic for the weight matrices W of neural networks: $f(W) = WW^\top$. With an orthogonal initialization for the weights, this statistic enables efficient optimization.

We gave four examples of proximity statistics that, together with a distance function, yield proximity constraints. We emphasize that any function of the varia-

tional parameters $f(\lambda)$ can be designed to ameliorate issues with variational inference. We discuss how to select a proximity statistic in Section 5.5.

5.3.3 Taylor-expanding the Proximity Constraint for Speed

PVI in Algorithm 1 requires optimizing the update equation, Equation (5.8), at each iteration. This rarely has a closed-form solution and requires a separate optimization procedure that is computationally expensive.

An alternative is to use a first-order Taylor expansion of the proximity constraint. Let ∇d be the gradient with respect to the second argument of the distance function, and $f(\tilde{\lambda})$ be the first argument to the distance. We compute the expansion around λ_t (the variational parameters at step t),

$$\begin{aligned} U(\lambda_{t+1}) =& \mathcal{L}(\lambda_t) + \nabla\mathcal{L}(\lambda_t)^\top(\lambda_{t+1} - \lambda_t) \\ &- \frac{1}{2\rho}(\lambda_{t+1} - \lambda_t)^\top(\lambda_{t+1} - \lambda_t) \\ &- k \cdot (d(f(\tilde{\lambda}), f(\lambda_t)) \\ &+ \nabla d(f(\tilde{\lambda}), f(\lambda_t))\nabla f(\lambda_t)^\top(\lambda_{t+1} - \lambda_t)). \end{aligned}$$

This Taylor expansion enjoys a closed-form solution for the variational parameters λ_{t+1},

$$\lambda_{t+1} = \lambda_t + \rho(\nabla\mathcal{L}(\lambda_t) - k \cdot (\nabla d(f(\tilde{\lambda}), f(\lambda_t))\nabla f(\lambda_t)). \tag{5.9}$$

Note that setting $\tilde{\lambda}$ to the current parameter λ_t removes the proximity constraint. Distance functions are minimized at zero so their derivative is zero at that point.

Fast PVI is detailed in Algorithm 2. Unlike PVI in Algorithm 1, the update in Equation (5.9) does not require an inner optimization. Fast PVI is tested in Section 5.4. The complexity of fast PVI is similar to standard VI because fast PVI optimizes the ELBO subject to the distance constraint in f. (The added complexity comes from computing the derivative of f; no inner optimization loop is required.)

Finally, note that fast PVI implies a global objective which varies over time. It is

$$\mathcal{L}_{\text{proximity}}(\lambda_{t+1}) = \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z})] - \mathbb{E}_q[\log q(\lambda_{t+1})] - k \cdot d(f(\tilde{\lambda}), f(\lambda_{t+1})).$$

Because d is a distance, this remains a lower bound on the evidence, but where new variational approximations remain close in f to previous iterations' distributions.

Inference Method	ELBO	Likelihood
Variational Inference	-121.4	-113.7
Deterministic Annealing	-116.8	-108.8
PVI, Entropy Constraint	**-113.3**	**-106.7**
PVI, Mean/Variance Constraint	-114.9	-107.4

Table 5.1: Proximity variational inference improves on deterministic annealing (Katahira et al., 2008) and VI in a one-layer sigmoid belief network. We report the test set evidence lower bound (ELBO) and marginal likelihood on the binary MNIST dataset (Larochelle and Murray, 2011). The model has one stochastic layer of 200 latent variables. PVI outperforms deterministic annealing (Katahira et al., 2008) and the classical variational inference algorithm.

5.4 Empirical Study

We developed proximity variational inference (PVI). We now empirically study PVI, variational inference, and deterministic annealing (Katahira et al., 2008).[2]

We first study sigmoid belief networks and find that PVI improves over deterministic annealing and VI in terms of held-out values of the ELBO and marginal likelihood. We then study a variational autoencoder model of images. Using an orthogonal proximity statistic, we show that PVI improves over classical VI by reducing overpruning. Finally, we study a deep generative model fit to a large corpus of text, where PVI yields better predictive performance with little hyperparameter tuning.[3]

Hyperparameters. For PVI, we use the inverse Huber distance for d.[4] The inverse Huber distance penalizes smaller values than the square difference. For PVI Algorithm 2, we set the exponential moving average decay constant for $\tilde{\lambda}$ to $\alpha = 0.9999$. We set the constraint scale k (or temperature parameter in deterministic annealing) to the initial absolute value of the ELBO unless otherwise specified. We explore two annealing schedules for PVI and deterministic annealing: a linear decay and an exponential decay. For the exponential decay, the value of the magnitude at iteration t of T total iterations is set to $k \cdot \gamma^{\frac{t}{T}}$ where γ is the

[2] Source code for reproducibility is available at https://github.com/altosaar/proximity_vi.

[3] We also compared PVI to Khan et al. (2015). Specifically, we tested PVI on the Bayesian logistic regression model from that paper and with the same data. Because Bayesian logistic regression has a single mode, all methods performed equally well. We note that we could not apply their algorithm to the sigmoid belief network because it would require approximating difficult iterated expectations.

[4] We define the inverse Huber distance $d(x, y)$ to be $|x - y|$ if $|x - y| < 1$ and $0.5(x - y)^2 + 0.5$ otherwise. The constants ensure the function and its derivative are continuous at $|x - y| = 1$.

Inference Method	ELBO	Likelihood
Variational Inference	-116.2	-104.9
Deterministic Annealing	-102.0	-94.2
PVI, Entropy Constraint	**-99.7**	**-93.2**
PVI, Mean/Variance Constraint	-100.7	-93.3

Table 5.2: Proximity variational inference improves over deterministic annealing and VI in a three-layer sigmoid belief network. The model has three layers of 200 latent variables. We report the evidence lower bound (ELBO) and marginal likelihood on the MNIST test set (Larochelle and Murray, 2011).

decay rate. We use the Adam optimizer (Kingma and Ba, 2015) unless otherwise specified.

5.4.1 Sigmoid Belief Network

The sigmoid belief network is a discrete latent variable model with layers of Bernoulli latent variables (Neal, 1992; Ranganath et al., 2015). It is used to benchmark variational inference algorithms (Mnih and Rezende, 2016). The approximate posterior is a collection of Bernoullis, parameterized by an inference network with weights and biases. We fit these variational parameters with VI, deterministic annealing (Katahira et al., 2008), or PVI, and learn the model parameters (weights and biases) using variational expectation-maximization.

We learn the weights and biases of the model with gradient ascent. We use a step size of $\rho = 10^{-3}$ and train for 4×10^6 iterations with a batch size of 20. For PVI Algorithm 2 and deterministic annealing, we grid search over exponential decays with rates $\gamma \in \{10^{-5}, 10^{-6}, ..., 10^{-10}, 10^{-20}, 10^{-30}\}$ and report the best results for each algorithm. (We also explored linear decays but they did not perform as well.) To reduce the variance of the gradients, we use the leave-one-out control variate of Mnih and Rezende (2016) with 5 samples. (This is an extension to the black box variational inference algorithm in Ranganath et al. (2014).)

Results on MNIST. We train a sigmoid belief network model on the binary MNIST dataset of handwritten digits (Larochelle and Murray, 2011). For evaluation, we compute the ELBO and held-out marginal likelihood with importance sampling on the validation set of 10^4 digits using 5000 samples, as in Rezende et al. (2014). In Table 1 we show the results for a model with one layer of 200 latent variables. Table 5.2 displays similar results for a three-layer model with 200 latent variables per layer. In both one and three-layer models the KL proximity statistic performs worse than the mean/variance and entropy statistics; it requires different decay schedules. Overall, PVI with the entropy and mean/variance proximity

Inference Method	ELBO	Likelihood
Variational Inference	-101.0	-94.2
PVI, Orthogonal Constraint	**-100.4**	**-93.9**

Table 5.3: Proximity variational inference with an orthogonal proximity statistic makes optimization easier in a variational autoencoder model (Kingma and Welling, 2014; Rezende et al., 2014). We report the held-out evidence lower bound (ELBO) and estimates of the marginal likelihood on the binarized MNIST (Larochelle and Murray, 2011) test set.

statistics yields improvements in the held-out marginal likelihood in comparison to deterministic annealing and VI.

5.4.2 Variational Autoencoder

To demonstrate the value of designing proximity statistics tailored to specific models, we study the variational autoencoder (Kingma and Welling, 2014; Rezende et al., 2014). This model is difficult to optimize, and current optimization techniques yield solutions that do not use the full model capacity (Burda et al., 2015). In Section 5.3.2 we designed an orthogonal proximity statistic to make backpropagation in neural networks easier. We show that this statistic enables us to find a better approximate posterior in the variational autoencoder by reducing overpruning.

We fit the variational autoencoder to binary MNIST data (Larochelle and Murray, 2011) with variational expectation-maximization. The model has one layer of 100 Gaussian latent variables. The inference network and generative network are chosen to have two hidden layers of size 200 with rectified linear units. We use an orthogonal initialization for the inference network weights. The learning rate is set to 10^{-3} and we run VI and PVI for 5×10^4 iterations. The orthogonal proximity statistic changes rapidly during optimization, so we use constraint magnitudes $k \in \{1, 10^{-1}, 10^{-2}, ..., 10^{-5}\}$, with no decay, and report the best result.

We compute the ELBO and importance-sampled marginal likelihood estimates on the validation set. Table 5.3 shows that PVI with the orthogonal proximity statistic on the weights of the inference network enables easier optimization and improves over VI.

Why does PVI improve upon VI in the variational autoencoder? The choice of rectified linear units in the inference network allows us to study overpruning of the latent code (MacKay, 2001; Burda et al., 2015). We study the fraction of 'dead units'— the fraction of rectified linear units in each layer of the inference neural network whose input is below zero. With PVI Algorithm 2 and the orthogonal

Inference Method	Perplexity
Variational Inference	2329
PVI, Mean/Variance Constraint	**2294**

Table 5.4: Proximity variational inference with a mean/variance proximity statistic improves predictive performance in a deep exponential family model with Poisson latent variables. We report the held-out perplexity on the *Science* corpus of journal articles.

proximity constraint, the inference network has 1.6% fewer dead units in the hidden layer and shows a 3.2% reduction in the output layer than in the same model learned using classic variational inference.

Once the input to a rectified linear unit drops below zero, the unit stops receiving gradient updates. The output layer parametrizes the latent variable distribution, so this means PVI reduced the pruning of the approximate posterior and led to the utilization of 3 additional latent variables. This is the reason it outperformed a variational autoencoder fit with VI.

5.4.3 Deep Generative Model of Text

Deep exponential family models, Bayesian analogues to neural networks, represent a flexible class of models (Ranganath et al., 2015). However, black box variational inference is commonly used to fit these models, which requires variance reduction (Ranganath et al., 2014). Deep exponential family models with Poisson latent variables present a challenging approximate inference problem because they are discrete and high-variance. We demonstrate that PVI with the mean/-variance proximity constraint improves predictive performance in such an unsupervised model of text.

The generative process for a single-layer deep exponential family model of text, with Poisson latent variables and Poisson likelihood, is

$$\begin{aligned}\mathbf{z} &\sim \text{Poisson}(\lambda)\\ \mathbf{x} &\sim \text{Poisson}(\mathbf{z}^\top g(W)),\end{aligned}$$

where W are real-valued model parameters and g is an elementwise function that maps to the positive reals (we use the softplus function). The dimension of $\mathbf{z}$ is K, so the model parameters must have shape (K, V) where V is the cardinality of the count-valued observations $\mathbf{x}$. We use this as a model of documents, so $\mathbf{x}$ is the bag-of-words representation of word counts, W represents the common factors in documents, and the per-document latent variable $\mathbf{z}$ captures factors prevalent in documents' language.

university	fig	disease
new	dna	virus
department	protein	hiv
york	cells	aids
research	cell	human
science	gene	patients
state	binding	diseases
laboratory	two	cases
national	sequence	infection
california	proteins	infected

Table 5.5: The top ten words for three factors of a deep exponential family model with Poisson latent variables fit to the *Science* corpus of scientific articles. We show topics from a model fit with proximity variational inference; the topics for the same model fit with variational inference are similar.

We study the performance of our method on a corpus of articles from the academic journal *Science*. The corpus contains 138k documents in the training set, 1k documents in the test set, and 5.9k terms. We set the latent dimension to 100, and fit the variational Poisson parameters using black box variational inference (Ranganath et al., 2014) using minibatches of size 64 and 32 samples of the latent variables to estimate the gradients.

Poisson variables have high variance, so we use the optimal control variate scaling developed in Ranganath et al. (2014) and estimate this scaling in a round-robin fashion as in Mnih and Rezende (2016) for efficiency. We use the RMSProp adaptive gradient optimizer (Tieleman and Hinton, 2012) with a step size of 0.01. For PVI Algorithm 2 with the mean/variance proximity statistic, we use an exponential decay for the constraint and test decay rates γ of 10^{-5} and 10^{-10}. We train for 10^6 iterations on the *Science* corpus, using variational expectation-maximization to learn the model parameters.

For evaluation, we keep the model parameters fixed and hold out 90% of the words in each document in the test set. Using the 10% of observed words in each document, we learn the variational parameters using PVI or variational inference with 300 iterations per document. We compute perplexity on the held-out documents, which is given by

$$\exp\left(\frac{-\sum_{d\in\text{docs}}\sum_{w\in d}\log p(w \mid \#\text{ held-out in d})}{N_{\text{held-out words}}}\right).$$

Conditional on the number of held-out words in a document, the distribution over held-out words is multinomial. The mean of the conditional multinomial is the normalized Poisson rate of the document matrix-multiplied with the softplus of the weights. This is the same evaluation metric as in Ranganath et al. (2015).

The results of fitting the model to the corpus of *Science* documents are reported in Table 5.4 and Table 5.5. While the topics found by models fit with both PVI and VI are similar, PVI gives better predictive performance in terms of held-out perplexity.

5.5 Discussion

We presented proximity variational inference, a flexible method designed to avoid bad local optima. We showed that classic variational inference gets trapped in these local optima and cannot recover. The choice of proximity statistic f and distance d enables the design of a variety of constraints that improve optimization. As examples of proximity statistics, we gave the entropy, KL divergence, orthogonal proximity statistic, and the mean and variance of the approximate posterior. We evaluated our method in four models to demonstrate that it is easy to implement, readily extensible, and leads to beneficial statistical properties of variational inference algorithms.

The empirical results also yield guidelines for choosing proximity statistics. The entropy is useful for models with discrete latent variables which are prone to quickly getting stuck in local optima or flat regions of the objective. We also saw that the KL statistic gives poor performance empirically, and that the orthogonal proximity statistic reduces pruning in deep generative models such as the variational autoencoder. In models like the deep exponential family model of text, the entropy is not tractable so the mean/variance proximity statistic is a natural choice.

Future Work. Simplifying optimization is necessary for truly black-box variational inference. An adaptive magnitude decay based on the value of the constraint should further improve the technique (this could be done per-parameter). New proximity constraints are also easy to design and test. For example, the variance of the gradients of the variational parameters is a valid proximity statistic—which can be used to avoid variational approximations that have high-variance gradients. Another set of interesting proximity statistics are empirical statistics of the variational distribution, such as the mean, for when analytic forms are unavailable. We also leave the design and study of constraints that admit coordinate updates to future work.

Chapter 6

Discussion

PROBABILISTIC modeling is useful across scientific domains. However, probabilistic modeling methods that do not take into account the structure of a problem, the form of individual datapoints, or information about probability distributions during optimization leave performance gains on the table.

As a motivating example, we built the structure of a statistical physics model into a probabilistic modeling method with hierarchical variational models. Efficient use of the connectivity patterns in physics models enabled scaling variational approximations to models with millions of random variables.

There is also utility in constructing probabilistic models with knowledge about individual datapoints. RANKFROMSETS outperforms competitive recommendation models that either fail to take into account the goals of recommendation or the structure of items with sets of attributes.

We also improved variational inference, by making use of information about probability distributions within the proximity variational inference algorithm. This enabled accurate inferences about probability distributions.

To further unify the thesis of problem structure as utile in probabilistic modeling, we test proximity variational inference to measure whether the benefits of leveraging knowledge about a probability distributions are additive to performance gains from developing applied methods.

Consider an Ising model studied in Chapter 3, where the goal is accurate inference of the free energy. Table 6.1 shows a comparison between VI and PVI in an HVM. This is a result of testing the best-performing settings from Chapter 5 with the entropy constraint on both the variational prior and recursive variational approximation in an HVM. The additional information PVI makes available to the

Inference Method	Free Energy
Variational Inference	-2.144
PVI, Entropy Constraint	**-2.158**

Table 6.1: Proximity variational inference with the entropy proximity statistic improves the accuracy of an HVM applied to an Ising model with 256 random variables. Building on the tools developed in Chapter 3 and Chapter 5, we report an importance sampling estimate of the free energy (lower is better; the exact value is $F \approx -2.198$ at inverse temperature $\beta = 0.4$).

Model	Recall @ 10 (%)	Recall @ 100 (%)
RFS	0.32	2.54
RFS, Entropy Constraint	**0.44**	2.54

Table 6.2: Proximity variational inference with the entropy proximity statistic improves top-10 out-matrix recall of RFS fit to arXiv user behavior data. We report the recall for items with no clicks in the training data (described in Section 4.4) for the best-performing settings of both PVI and RFS. Recall at 100 recommendations is comparable between the methods.

variational approximation during optimization leads to more accurate inference of the free energy.

Further, PVI can be applied to probability models fit with maximum likelihood estimation. Table 6.2 reports the performance of a RFS model from Section 4.4 fit to arXiv user behavior data. Fitting the recommendation model using the PVI entropy proximity constraint improves top-10 recommendation recall. Metrics other than out-matrix recall (e.g. in-matrix recall) were comparable between these methods. With the PVI entropy constraint, the recommendation performance of RFS also improved in the meal recommendation task. Table 6.3 reports these results. The best-performing settings from Chapter 5 generalize to maximum likelihood estimation in recommender systems, here giving a 6.9% boost in top-1 recommender recall.

That PVI yielded improvements when applied to both HVMs applied to statistical physics problems and the RFS recommendation model highlights several directions for further research. First, might PVI yield further gains in accuracy when applied to statistical physics models with millions of random variables? Practitioners are willing to trade off diminished accuracy for scale in some cases, and PVI is straightforward to test in new probability models and might help reduce the need for such trade-offs.

Model	Sampled Recall (%)
RFS	58
RFS, Entropy Constraint	**62**

Table 6.3: Proximity variational inference with the entropy proximity statistic improves top-1 recall on a meal recommendation task.

Studying where PVI yields marginal gains is also worth considering. For example, the entropy proximity constraint yielded less-significant improvements when applied to RFS fit to the meal recommendation data in Section 4.4. This may be because the large size of data helped prevent overfitting, leading to reduced benefits of constraining parameter updates. In contrast, HVMs fit to statistical physics models in Section 3.3 converged to a solution very quickly, so monitoring convergence rates may be an additional source of information for proximity statistics.

While Chapter 3 studied classical statistical physics models, future work in computational materials science and computational drug discovery will need to incorporate or approximate quantum effects. Density functional theory calculations based on quantum mechanics are expensive (Schmidt et al., 2019) and limit the length of time that a material or drug binding to a protein can be simulated. Future work in this area should include study of the trade-off between the size of a system and the accuracy needed to study the behavior of a system to achieve a materials design or drug design goal. For example, suppose the behavior of a drug binding to a protein over the course of several seconds is of clinical interest. Then a practitioner might tolerate more inaccuracy in an HVM approximation than they would if the short-run behavior could be accurately captured in a density functional theory calculation. One way of improving the trade-off may be to reduce the cost of fitting HVMs by derive objective functions with better gradient signal-to-noise ratio (Tucker et al., 2019; Rainforth et al., 2018). A similar trade-off occurs for system size, and it is unclear where HVMs may provide the only way to model a large-scale physical system.

Chapter 4 developed RFS, and there remain several directions for future work on recommendation models for items with sets of attributes. Probabilistic generative models for use in recommendation may enable better recommendations under uncertainty, or easier incorporation of prior knowledge. However, probability distributions of sets of attributes are difficult to parameterize. One example of a distribution defined on sets is the Wallenius distribution (Wallenius, 1963; Junqu et al., 2000). It is interesting to consider how a distribution on sets might be parameterized using a permutation-invariant model such as RFS (Bloem-Reddy and Teh, 2019; Lee et al., 2018). Further, generalization bounds are necessary follow-up work to universal approximation properties. A

model may be able to represent a distribution, but for practical purposes a key desideratum is finding functions, nonlinearities, and architectures that make optimization easy and generalization feasible (Dziugaite and Roy, 2017).

Another line of work is in developing robust negative sampling-based objective functions. The numeric value of the negative log-likelihood objective function used in RFS or other models that use negative samples and embeddings cannot reliably assess convergence. This is due to embeddings that are used in both positive and negative examples, leading stochastic gradient updates to increase and decrease Monte Carlo estimates of the objective during optimization. Reliable methods to estimate the value of objective functions may help reduce the need for expensive recommender systems evaluation metrics where a model may need to be evaluated on every item in an evaluation set. While RFS was designed for the recall evaluation metric, connecting binary classification objective functions with negative examples to ranking-based metrics such as normalized discounted cumulative gain would make these models useful broadly.

In Chapter 4, we found that RFS outperforms LSTM recurrent neural networks in the task of recommending arXiv documents to users. This is counterintuitive, as the order of item attributes (words in abstracts) should carry significant information. However, the computational budget was fixed for both models, and it is unclear which recommendation model to use with a large computational budget. Models such as transformers (Vaswani et al., 2017; Devlin et al., 2019; Lee et al., 2018) might lead to improved recommendation performance, but at a greater computational cost than models such as RFS with inner product parameterizations. Analyzing these trade-offs will help make informed choices of computational budget given performance requirements in practice. Under computational constraints due to monetary budget or privacy regulation, such as in clinical settings (Huang et al., 2020), models such as RFS that make fast, accurate, predictions may be preferable to more accurate, slower models.

Through careful consideration of how to build problem structure into probabilistic models, we were able to scale variational methods to statistical physics models with millions of random variables, fit recommender systems to tens of millions of datapoints, and improve the accuracy of variational inference. This highlights the need to ensure that progress in probabilistic modeling continues to be translated into progress in applied domains such as statistical physics and recommender systems.

Bibliography

Altosaar, Jaan, Rajesh Ranganath, and David M. Blei (2018). "Proximity Variational Inference". *Artificial Intelligence and Statistics*.

Altosaar, Jaan, Rajesh Ranganath, and Kyle Cranmer (2019). "Hierarchical Variational Models for Statistical Physics". *Machine Learning and the Physical Sciences Workshop, Neural Information Processing Systems*.

Altosaar, Jaan, Wesley Tansey, and Rajesh Ranganath (2020). "RankFromSets: Scalable Set Recommendation with Optimal Recall". *American Statistical Association Symposium on Data Science & Statistics*.

Andrieu, C., N. de Freitas, A. Doucet, and M. Jordan (2003). "An introduction to MCMC for machine learning". *Machine Learning*.

Bamler, Robert, Cheng Zhang, Manfred Opper, and Stephan Mandt (2017). "Perturbative Black Box Variational Inference". *Neural Information Processing Systems*.

Bansal, Trapit, David Belanger, and Andrew McCallum (2016). "Ask the GRU: Multi-task Learning for Deep Text Recommendations". *ACM Recommender Systems*.

Basu, Chumki, Haym Hirsh, and William Cohen (1998). "Recommendation As Classification: Using Social and Content-based Information in Recommendation". *AAAI Conference on Artificial Intelligence*.

Bishop, Christopher M (2006). *Pattern Recognition and Machine Learning*.

Blei, David M (2014). "Build, compute, critique, repeat: Data analysis with latent variable models". *Annual Review of Statistics and Its Application*.

Blei, David M., Alp Kucukelbir, and Jon D. McAuliffe (2017). "Variational Inference: A Review for Statisticians". *Journal of the American Statistical Association*.

Bloem-Reddy, Benjamin and Yee Whye Teh (2019). "Probabilistic symmetry and invariant neural networks". *arXiv:1901.06082*.

Bogers, Toine (2018). "Social Information Access". *Social Information Access: Systems and Technologies*. Chap. Tag-Based Recommendation.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov (2017). "Enriching Word Vectors with Subword Information". *Association for Computational Linguistics*.

Bowman, Samuel R., Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio (2016). "Generating Sentences from a Continuous Space". *Conference on Computational Natural Language Learning*.

Boyd, Stephen and Lieven Vandenberghe (2004). *Convex Optimization*.

Brooks, Stephen P and Andrew Gelman (1998). "General Methods for Monitoring Convergence of Iterative Simulations". *Journal of Computational and Graphical Statistics*.

Burda, Yuri, Roger Grosse, and Ruslan Salakhutdinov (2015). "Importance weighted autoencoders". *International Conference on Learning Representations*.

Cao, Da, Liqiang Nie, Xiangnan He, Xiaochi Wei, Shunzhi Zhu, and Tat-Seng Chua (2017). "Embedding Factorization Models for Jointly Recommending Items and User Generated Lists". *ACM Special Interest Group on Information Retrieval*.

Chandler, D. and D. Wu (1987). *Introduction to Modern Statistical Mechanics*.

Chen, Ting, Liangjie Hong, Yue Shi, and Yizhou Sun (2017). "Joint Text Embedding for Personalized Content-based Recommendation". *arXiv:1706.01084*.

Chen, Yifan and Maarten de Rijke (2018). "A Collective Variational Autoencoder for Top-N Recommendation with Side Information". *ACM Workshop on Deep Learning for Recommender Systems*.

Cheng, Heng-Tze, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah (2016). "Wide & Deep Learning for Recommender Systems". *ACM Workshop on Deep Learning for Recommender Systems*.

Cover, Thomas M and Joy A Thomas (2012). *Elements of information theory*.

Covington, Paul, Jay Adams, and Emre Sargin (2016). "Deep Neural Networks for YouTube Recommendations". *ACM Conference on Recommender Systems*.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". *Association for Computational Linguistics*.

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2017). "Density estimation using Real NVP". *International Conference on Learning Representations*.

Dong, Xin, Lei Yu, Zhonghuo Wu, Yuxia Sun, Lingfeng Yuan, and Fangxi Zhang (2017). "A Hybrid Collaborative Filtering Model with Deep Structure for Recommender Systems". *AAAI Conference on Artificial Intelligence*.

Dziugaite, Gintare Karolina and Daniel M. Roy (2017). "Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data". *Uncertainty in Artificial Intelligence*.

Ebesu, Travis, Bin Shen, and Yi Fang (2018). "Collaborative Memory Network for Recommendation Systems". *ACM Special Interest Group on Information Retrieval*.

Elsweiler, David, Christoph Trattner, and Morgan Harvey (2017). "Exploiting Food Choice Biases for Healthier Recipe Recommendation". *ACM SIGIR Conference on Research and Development in Information Retrieval*.

Feynman, R.P. (1972). *Statistical Mechanics: A Set Of Lectures*.

– (2018). *Statistical Mechanics: A Set Of Lectures*.

Freyne, Jill, Shlomo Berkovsky, and Gregory Smith (2011). "Recipe Recommendation: Accuracy and Reasoning". *User Modeling, Adaption and Personalization*.

Gopalan, Prem, Laurent Charlin, and David M. Blei (2014). "Content-based Recommendations with Poisson Factorization". *Neural Information Processing Systems*.

Graves, Alex (2013). "Generating Sequences With Recurrent Neural Networks". *arXiv:1308.0850*.

Guo, Huifeng, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He (2017). "DeepFM: A Factorization-Machine based Neural Network for CTR Prediction". *International Joint Conference on Artificial Intelligence*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition". *IEEE Conference on Computer Vision and Pattern Recognition*.

He, Xiangnan, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua (2017). "Neural collaborative filtering". *International World Wide Web Conference*.

Henelius, P., T. Lin, M. Enjalran, Z. Hao, J. G. Rau, J. Altosaar, F. Flicker, T. Yavors'kii, and M. J. P. Gingras (2016). "Refrustration and competing orders in the prototypical $Dy_2Ti_2O_7$ spin ice material". *Physical Review B*.

Hoffman, M., D. Blei, C. Wang, and J. Paisley (2013). "Stochastic Variational Inference". *Journal of Machine Learning Research*.

Hoffman, Matthew and David Blei (2015). "Stochastic Structured Variational Inference". *Artifical Intelligence and Statistics*.

Hu, Yifan, Florham Park, Yehuda Koren, Chris Volinsky, and Florham Park (2008). "Collaborative Filtering for Implicit Feedback Datasets". *IEEE International Conference on Data Mining*.

Huang, Kexin, Jaan Altosaar, and Rajesh Ranganath (2020). "ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission". *ACM Conference on Health, Inference, and Learning*.

Huang, Po-Sen, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck (2013). "Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data". *ACM International Conference on Information & Knowledge Management*.

Jordan, M. (2004). "Graphical models". *Statistical Science*.

Jordan, M., Z. Ghahramani, T. Jaakkola, and L. Saul (1999). "Introduction to Variational Methods for Graphical Models". *Machine Learning*.

Junqu, Enric, David Martens, and Foster Provost (2000). "Wallenius Naive Bayes".

Katahira, K, K Watanabe, and M Okada (2008). "Deterministic annealing variant of variational Bayes method". *Journal of Physics: Conference Series*.

Khan, Mansura A., Ellen Rushe, Barry Smyth, and David Coyle (2019). "Personalized, Health-Aware Recipe Recommendation: An Ensemble Topic Modeling Based Approach". *ACM Conference on Recommender Systems, Workshop on Health Recommender Systems*.

Khan, Mohammad E., Reza Babanezhad, Wu Lin, Mark Schmidt, and Masashi Sugiyama (2016). "Faster Stochastic Variational Inference Using Proximal-Gradient Methods with General Divergence Functions". *Uncertainty in Artificial Intelligence*.

Khan, Mohammad E., Pierre Baqué, François Fleuret, and Pascal Fua (2015). "Kullback-Leibler Proximal Variational Inference". *Neural Information Processing Systems*.

Kingma, Diederik P and Max Welling (2014). "Auto-Encoding Variational Bayes". *International Conference on Learning Representations*.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". *International Conference on Learning Representations*.

Kingma, Durk P, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling (2016). "Improved Variational Inference with Inverse Autoregressive Flow". *Neural Information Processing Systems*.

Koren, Y., R. Bell, and C. Volinsky (2009). "Matrix Factorization Techniques for Recommender Systems". *Computer*.

Kucukelbir, Alp, Rajesh Ranganath, Andrew Gelman, and David M Blei (2015). "Automatic Variational Inference in Stan". *Neural Information Processing Systems*.

Kula, Maciej (2015). "Metadata Embeddings for User and Item Cold-start Recommendations". *arXiv:1507.08439*.

Larochelle, Hugo and Iain Murray (2011). "The Neural Autoregressive Distribution Estimator". *Artificial Intelligence and Statistics*.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". *Nature*.

Lee, Juho, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh (2018). "Set Transformer". *arXiv:1810.00825*.

Li, Huayu, Richang Hong, Defu Lian, Zhiang Wu, Meng Wang, and Yong Ge (2016). "A Relaxed Ranking-based Factor Model for Recommender System from Implicit Feedback". *International Joint Conference on Artificial Intelligence*.

Lian, Jianxun, Fuzheng Zhang, Xing Xie, and Guangzhong Sun (2018). "Towards Better Representation Learning for Personalized News Recommendation: a Multi-Channel Deep Fusion Approach". *International Joint Conference on Artificial Intelligence*.

Liang, Dawen, Jaan Altosaar, Laurent Charlin, and David M. Blei (2016). "Factorization Meets the Item Embedding: Regularizing Matrix Factorization with Item Co-occurrence". *ACM Conference on Recommender Systems*.

Liang, Junjie, Jinlong Hu, Shoubin Dong, and Vasant G. Honavar (2018a). "Top-N-Rank: A Scalable List-wise Ranking Method for Recommender Systems". *arXiv:1812.04109*.

Liang, Nan, Hai-Tao Zheng, Jin-Yuan Chen, Arun Sangaiah, and Cong-Zhi Zhao (2018b). "TRSDL: Tag-Aware Recommender System Based on Deep Learning–Intelligent Computing Systems". *Applied Sciences*.

Liu, Yidan, Min Xie, and Laks V.S. Lakshmanan (2014). "Recommending User Generated Item Lists". *ACM Recommender Systems*.

Loepp, Benedikt, Tim Donkers, Timm Kleemann, and Jürgen Ziegler (2019). "Interactive recommending with Tag-Enhanced Matrix Factorization (TagMF)". *International Journal of Human-Computer Studies*.

Maaløe, Lars, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther (2016). "Auxiliary Deep Generative Models". *International Conference on Machine Learning*.

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing Data using t-SNE". *Journal of Machine Learning Research*.

MacKay, David (2003). *Information Theory, Inference, and Learning Algorithms*.

MacKay, David J.C. (2001). "Local minima, symmetry-breaking, and model pruning in variational free energy minimization." *Accessed online at* `inference.org.uk`.

Mansinghka, Vikash, Daniel Selsam, and Yura N. Perov (2014). "Venture: a higher-order probabilistic programming platform with programmable inference". *arXiv:1404.0099*.

Metropolis, Nicholas, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller (1953). "Equation of State Calculations by Fast Computing Machines". *The Journal of Chemical Physics*.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Distributed Representations of Words and Phrases and their Compositionality". *Neural Information Processing Systems*.

Mnih, Andriy and Danilo J. Rezende (2016). "Variational Inference for Monte Carlo Objectives". *International Conference on Machine Learning*.

Mohamed, Shakir, Mihaela Rosca, Michael Figurnov, and Andriy Mnih (2019). "Monte Carlo Gradient Estimation in Machine Learning". *arXiv:1906.10652*.

Nakajima, Shinichi, Masashi Sugiyama, S. Derin Babacan, and Ryota Tomioka (2013). "Global Analytic Solution of Fully-observed Variational Bayesian Matrix Factorization". *Journal of Machine Learning Research*.

Neal, Radford M (1992). "Connectionist Learning of Belief Networks". *Artificial Intelligence*.

Neal, Radford M et al. (2011). "MCMC using Hamiltonian dynamics". *Handbook of Markov Chain Monte Carlo*.

Okura, Shumpei, Yukihiro Tagami, Shingo Ono, and Akira Tajima (2017). "Embedding-based News Recommendation for Millions of Users". *ACM Knowledge Discovery and Data Mining*.

Owen, Art B. (2013). *Monte Carlo Theory, Methods and Examples*.
Palangi, Hamid, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward (2016). "Deep Sentence Embedding Using Long Short-term Memory Networks: Analysis and Application to Information Retrieval". *IEEE/ACM Transactions on Audio, Speech and Language Processing*.
Papamakarios, George, Theo Pavlakou, and Iain Murray (2017). "Masked Autoregressive Flow for Density Estimation". *Neural Information Processing Systems*.
Rainforth, Tom, Adam R. Kosiorek, Tuan Anh Le, Chris J. Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh (2018). "Tighter Variational Bounds are Not Necessarily Better". *International Conference on Machine Learning*.
Ranganath, Rajesh (2018). "Black Box Variational Inference: Scalable, Generic Bayesian Computation and its Applications". PhD thesis. Princeton University.
Ranganath, Rajesh, Sean Gerrish, and David Blei (2014). "Black Box Variational Inference". *Artificial Intelligence and Statistics*.
Ranganath, Rajesh, Linpeng Tang, Laurent Charlin, and David M Blei (2015). "Deep Exponential Families". *Artificial Intelligence and Statistics*.
Regier, Jeffrey, Keno Fischer, Kiran Pamnany, Andreas Noack, Jarrett Revels, Maximilian Lam, Steve Howard, Ryan Giordano, David Schlegel, Jon McAuliffe, Rollin Thomas, and Prabhat (2019). "Cataloging the visible universe through Bayesian inference in Julia at petascale". *Journal of Parallel and Distributed Computing*.
Rendle, Steffen, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme (2009). "BPR: Bayesian Personalized Ranking from Implicit Feedback". *Uncertainty in Artificial Intelligence*.
Rezende, Danilo J. and Shakir Mohamed (2015). "Variational Inference with Normalizing Flows". *International Conference on Machine Learning*.
Rezende, Danilo J., Shakir Mohamed, and Daan Wierstra (2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". *International Conference on Machine Learning*.
Robbins, H. and S. Monro (1951). "A Stochastic Approximation Method". *The Annals of Mathematical Statistics*.
Saxe, Andrew M, James L McClelland, and Surya Ganguli (2014). "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks". *International Conference on Learning Representations*.
Schmidt, Jonathan, Mário R. G. Marques, Silvana Botti, and Miguel A. L. Marques (2019). "Recent advances and applications of machine learning in solid-state materials science". *npj Computational Materials*.
Shamay, Yosi, Janki Shah, Mehtap Işık, Aviram Mizrachi, Josef Leibold, Darjus F. Tschaharganeh, Daniel Roxbury, Januka Budhathoki-Uprety, Karla Nawaly, James L. Sugarman, Emily Baut, Michelle R. Neiman, Megan Dacek, Kripa S. Ganesh, Darren C. Johnson, Ramya Sridharan, Karen L. Chu, Vinagolu K.

Rajasekhar, Scott W. Lowe, John D. Chodera, and Daniel A. Heller (2018). "Quantitative self-assembly prediction yields targeted nanomedicines". *Nature Materials*.

Shi, Yue, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Alan Hanjalic, and Nuria Oliver (2012a). "TFMAP: Optimizing MAP for Top-N Context-Aware Recommendation". *ACM Special Interest Group on Information Retrieval*.

Shi, Yue, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic (2012b). "CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-more Filtering". *ACM Recommender Systems*.

Shi, Yue, Martha Larson, and Alan Hanjalic (2014). "Collaborative Filtering Beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges". *ACM Computing Surveys*.

Song, Bo, Xin Yang, Yi Cao, and Congfu Xu (2018). "Neural Collaborative Ranking". *ACM Conference on Information and Knowledge Management*.

Spall, James (2003). *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*.

Stokes, Jonathan M., Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M. Donghia, Craig R. MacNair, Shawn French, Lindsey A. Carfrae, Zohar Bloom-Ackerman, Victoria M. Tran, Anush Chiappino-Pepe, Ahmed H. Badran, Ian W. Andrews, Emma J. Chory, George M. Church, Eric D. Brown, Tommi S. Jaakkola, Regina Barzilay, and James J. Collins (2020). "A Deep Learning Approach to Antibiotic Discovery". *Cell*.

Stone, Anthony J (2013). *The theory of intermolecular forces; 2nd ed.*

Sutskever, Ilya, James Martens, George Dahl, and Geoffrey Hinton (2013). "On the Importance of Initialization and Momentum in Deep Learning". *International Conference on Machine Learning*.

Swendsen, Robert H. and Jian-Sheng Wang (1987). "Nonuniversal critical dynamics in Monte Carlo simulations". *Physical Review Letters*.

Theis, Lucas and Matthew D Hoffman (2015). "A trust-region method for stochastic variational inference with applications to streaming data". *Journal of Machine Learning Research*.

Tieleman, Tijmen and Geoffrey Hinton (2012). "Lecture 6.5 - rmsprop". *COURSERA: Neural Networks for Machine Learning*.

Tran, Dustin, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei (2016). "Edward: A library for probabilistic modeling, inference, and criticism". *arXiv:1610.09787*.

Trattner, Christoph and David Elsweiler (2018). "Food Recommender Systems: Important Contributions, Challenges and Future Research Directions". *Collaborative Recommendations: Algorithms, Practical Challenges and Applications*.

– (2019a). "An Evaluation of Recommendation Algorithms for Online Recipe Portals". *ACM Conference on Recommender Systems, Workshop on Health Recommender Systems*.

Trattner, Christoph and David Elsweiler (2019b). "What online data say about eating habits". *Nature Sustainability*.
Tucker, George, Dieterich Lawson, Shixiang Gu, and Chris J. Maddison (2019). "Doubly Reparameterized Gradient Estimators for Monte Carlo Objectives". *International Conference on Learning Representations*.
Vaart, Adrianus Willem van der (1998). *Asymptotic Statistics*.
Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". *Neural Information Processing Systems*.
Veitch, Victor, Morgane Austern, Wenda Zhou, David M. Blei, and Peter Orbanz (2019). "Empirical Risk Minimization and Stochastic Gradient Descent for Relational Data". *International Conference on Artificial Intelligence and Statistics*.
Wainwright, M. and M. Jordan (2008). "Graphical models, exponential families, and variational inference". *Foundations and Trends in Machine Learning*.
Wallenius, K. T. (1963). "Biased Sampling: The Non-central Hypergeometric Probability Distribution". PhD thesis. Stanford University.
Wang, Chong and David M. Blei (2011). "Collaborative Topic Modeling for Recommending Scientific Articles". *ACM Knowledge Discovery and Data Mining*.
Wang, Yixin and David M. Blei (2019). "Frequentist Consistency of Variational Bayes". *Journal of the American Statistical Association*.
Welling, M and Y W Teh (2011). "Bayesian Learning via Stochastic Gradient Langevin Dynamics". *International Conference on Machine Learning*.
Wolff, Ulli (1989). "Comparison between cluster Monte Carlo algorithms in the Ising model". *Physics Letters B*.
Wu, Dian, Lei Wang, and Pan Zhang (2019). "Solving Statistical Mechanics Using Variational Autoregressive Networks". *Physical Review Letters*.
Wu, L., A. Fisch, S. Chopra, K. Adams, A. Bordes, and J. Weston (2018). "StarSpace: Embed All The Things!" *AAAI Conference on Artificial Intelligence*.
Xu, Zhenghua, Thomas Lukasiewicz, Cheng Chen, Yishu Miao, and Xiangwu Meng (2017). "Tag-Aware Personalized Recommendation Using a Hybrid Deep Model". *International Joint Conference on Artificial Intelligence*.
Yang, Longqi, Eugene Bagdasaryan, Joshua Gruenstein, Cheng-Kang Hsieh, and Deborah Estrin (2018). "OpenRec: A Modular Framework for Extensible and Adaptable Recommendation Algorithms". *ACM International Conference on Web Search and Data Mining*.
Yang, Longqi, Cheng-Kang Hsieh, Hongjian Yang, John P. Pollak, Nicola Dell, Serge Belongie, Curtis Cole, and Deborah Estrin (2017). "Yum-Me: A Personalized Nutrient-Based Meal Recommender System". *ACM Transactions on Information Systems*.
Ying, Haochao, Liang Chen, Yuwen Xiong, and Jian Wu (2016). "Collaborative Deep Ranking: A Hybrid Pair-Wise Recommendation Algorithm with

Implicit Feedback". *Pacific-Asia Conference on Knowledge Discovery and Data Mining*.
Yu, Lu, Chuxu Zhang, Shichao Pei, Guolei Sun, and Xiangliang Zhang (2018). "WalkRanker: A Unified Pairwise Ranking Model With Multiple Relations for Item Recommendation". *AAAI Conference on Artificial Intelligence*.
Yuan, Fajie, Guibing Guo, Joemon M. Jose, Long Chen, Haitao Yu, and Weinan Zhang (2016). "Optimizing Factorization Machines for Top-N Context-Aware Recommendations". *Web Information Systems Engineering*.
Zaheer, Manzil, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola (2017). "Deep Sets". *Neural Information Processing Systems*.
Zdeborová, Lenka and Florent Krzakala (2016). "Statistical physics of inference: thresholds and algorithms". *Advances in Physics*.
Zhang, Jun (1996). "The application of the Gibbs-Bogoliubov-Feynman inequality in mean field calculations for Markov random fields". *IEEE Transactions on Image Processing*.
Zhang, Shuai, Lina Yao, Aixin Sun, and Yi Tay (2019). "Deep Learning Based Recommender System: A Survey and New Perspectives". *ACM Computing Surveys*.
Zhang, Yan, Haoyu Wang, Defu Lian, Ivor W. Tsang, Hongzhi Yin, and Guowu Yang (2018). "Discrete Ranking-based Matrix Factorization with Self-Paced Learning". *ACM Knowledge Discovery and Data Mining*.
Zhen, Yi, Wu-Jun Li, and Dit-Yan Yeung (2009). "TagiCoFi: Tag Informed Collaborative Filtering". *ACM Recommender Systems*.
Zuo, Yi, Jiulin Zeng, Maoguo Gong, and Licheng Jiao (2016). "Tag-aware recommender systems based on deep neural networks". *Neurocomputing*.