Model Selection in Compositional Spaces

by

Roger Baker Grosse

B.S., Stanford University (2007)
M.S., Stanford University (2008)

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

Doctor of Philosophy in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2014

© Massachusetts Institute of Technology 2014. All rights reserved.

Author .....
Department of Electrical Engineering and Computer Science
January 24, 2014

Certified by .....
William T. Freeman
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chair of the Committee on Graduate Students

Model Selection in Compositional Spaces

by

Roger Baker Grosse

Submitted to the Department of Electrical Engineering and Computer Science
on January 24, 2014, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science

Abstract

We often build complex probabilistic models by composing simpler models—using one model to generate parameters or latent variables for another model. This allows us to express complex distributions over the observed data and to share statistical structure between different parts of a model. In this thesis, we present a space of matrix decomposition models defined by the composition of a small number of motifs of probabilistic modeling, including clustering, low rank factorizations, and binary latent factor models. This compositional structure can be represented by a context-free grammar whose production rules correspond to these motifs. By exploiting the structure of this grammar, we can generically and efficiently infer latent components and estimate predictive likelihood for nearly 2500 model structures using a small toolbox of reusable algorithms. Using a greedy search over this grammar, we automatically choose the decomposition structure from raw data by evaluating only a small fraction of all models. The proposed method typically finds the correct structure for synthetic data and backs off gracefully to simpler models under heavy noise. It learns sensible structures for datasets as diverse as image patches, motion capture, 20 Questions, and U.S. Senate votes, all using exactly the same code.

We then consider several improvements to compositional structure search. We present compositional importance sampling (CIS), a novel procedure for marginal likelihood estimation which requires only posterior inference and marginal likelihood estimation algorithms corresponding to the production rules of the grammar. We analyze the performance of CIS in the case of identifying additional structure within a low-rank decomposition. This analysis yields insights into how one should design a space of models to be recursively searchable. We next consider the problem of marginal likelihood estimation for the production rules. We present a novel method for obtaining ground truth marginal likelihood values on synthetic data, which enables the rigorous quantitative comparison of marginal likelihood estimators. Using

this method, we compare a wide variety of marginal likelihood estimators for the production rules of our grammar. Finally, we present a framework for analyzing the sequences of distributions used in annealed importance sampling, a state-of-the-art marginal likelihood estimator, and present a novel sequence of intermediate distributions based on averaging moments of the initial and target distributions.

Thesis Supervisor: William T. Freeman
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

First, I would like to thank my mentors for their constant encouragement and insight: my advisor, Bill Freeman, as well as Josh Tenenbaum and Ryan Adams. I thank Josh for suggesting the initial idea of a matrix decomposition grammar, which eventually formed the core of this thesis. It's always hard to predict where a research program will lead. In the first two years of my Ph.D., I attempted a range of projects in the convex hull of machine learning, computer vision, and cognitive science, and the one which finally panned out was right at the machine learning vertex. Bill and Josh were very supportive throughout, even as my research drifted away from their overall research agendas. They both encouraged me to work on ambitious problems, and to stick with them even while the work was still struggling to gain traction in the machine learning community.

Later on, Ryan Adams arrived at Harvard, and somehow found the time to mentor me on machine learning. (I suspect he has one of those time turners from Harry Potter.) He is full of insights, both practical and philosophical, and has greatly influenced the later parts of this thesis, especially the work on evaluating marginal likelihood estimators.

I was fortunate enough to visit Cambridge University and the University of Toronto, both of which led to fantastic collaborations which formed integral parts of this thesis. At Cambridge, Zoubin Gharamani taught me the tricks of Bayesian inference and got me thinking about the "automated Bayesian statistician," and his group had an enjoyable and informative assortment of weekly seminars, some of which we imported back at MIT. My visit to Toronto led me to the surprisingly rich topic of choosing annealing paths, and Geoff Hinton, Ruslan Salakhutdinov, and Radford Neal all gave a lot of insightful advice which contributed to this thesis. I would like to thank Zoubin and Ruslan for generously hosting me for these trips.

I would like to thank my other collaborators over the course of my Ph.D.: Ted Adelson, David Duvenaud, Kimo Johnson, Matt Johnson, Joseph Lim, James Lloyd, Chris Maddison, and Colorado Reed. I would also like to thank Martin Szummer for mentoring me in an internship at Microsoft Research Cambridge. I have learned a ton from working with all of these people, and from my interactions with the members of the MIT Vision Group, CoCoSci, and Harvard Intelligent Probabilistic Systems.

I would like to thank my parents, Eric and Brenda, and my sister Emily, for being supportive every step of the way. My parents have always encouraged my academic interests, and didn't flinch when I told them I'd chosen an undergraduate major called "symbolic systems." I will always be grateful for the advice and support they have given me over the years.

Contents

Chapter 1

Introduction

In supervised learning, there are a handful of generic algorithms which perform very well out of the box, including support vector machines, deep neural networks, and random forests. Unsupervised learning, however, still requires a great deal of human effort. The difference is that the supervised learning toolbox consists of a set of algorithms, while the unsupervised learning toolbox consists of modeling *motifs*, such as clustering and dimensionality reduction, which can be composed together in myriad ways to construct models.

Compositionality is a blessing because richly structured models can be tailored towards domains as diverse as vision, speech, and biology. It is also a curse, because modeling data requires specifying a model, designing, implementing, and debugging efficient inference algorithms, evaluating the fit of the model, and possibly refining the model if it is not a good enough match. Because of this challenge, most unsupervised learning software packages implement a handful of overly simple models such as principal component analysis or k-means. At the other extreme, probabilistic programming (Goodman et al., 2008; Borgstrom et al., 2011) aims to perform inference generically across a wide range of models. Such systems typically use general-purpose inference algorithms described at the level of individual random variables, such as Metropolis-Hastings or variational message passing. Unforunately, when designing algorithms at such a low level, it is difficult to take advantage of the specialized inference techniques researchers have developed for particular models.

In this thesis, we adopt a middle ground between the extremes of probabilistic programming and model-specific inference algorithms. Specifically, many probabilistic models are built compositionally out of simpler models for which effective approximate inference algorithms have been developed, such as dimensionality reduction, linear dynamics, and sparse coding. We propose a set of inference techniques geared

towards such high-level motifs rather than individual random variables, allowing for practical inference algorithms in a wide range of model structures.

Because the correct modeling assumptions may not be obvious in advance, a general purpose toolbox for unsupervised learning must also confront the problem of model selection. In principle, a software package could exhaustively evaluate all plausible models on a given dataset using a criterion such as held-out likelihood or marginal likelihood, and then choose whichever one performs the best. However, evaluating models is generally at least as hard as fitting them, so this would raise the same algorithmic challenges as inference. Furthermore, due to the combinatorial number of possible models, fitting all of them would likely be infeasible, or at least wasteful. For these reasons, model selection techniques are more commonly used to make smaller decisions within a pre-specified model class, such as which variables directly influence one another or how many clusters are needed to explain the data.

We propose a model selection strategy which explicitly takes advantage of the compositional structure of the space. In paticular, one can fit a simple model to the data, look for patterns in the model parameters or latent variables which weren't part of the model specification, and refine the model to capture the additional structure. We show that the recursive application of a handful of simple probabilistic modeling motifs allows for the discovery of rich, high-level structure.

In this thesis, we propose compositional frameworks for representing two types of models: matrix decompositions and Gaussian processes. Each of these frameworks can compactly represent a variety of existing models from the literature. We present techniques for posterior inference, model scoring, and structure search which directly exploit the compositional structure of a mdoel space. These techniques are shown to perform well empirically at discovering high-level representations of a wide variety of datasets using exactly the same code. Finally, we present some novel algorithms which should serve as building blocks in the compositional estimation of marginal likelihood. We believe this is a step towards a general purpose system for unsupervised learning and structure discovery.

1.1 Model composition

There are many operations by which simple models can be combined. In supervised learning, one often *averages* the predictions of an ensemble of models, a strategy that was used effectively in the Netflix challenge (Bell and Koren, 2007). In unsupervised learning, it is more common to use what we will term *composition*: using one generative model as a prior for certain random variables in another generative model. Specifically, consider a latent variable model where the observations $\mathbf{y}$ are

modeled in terms of latent variables $\mathbf{z}$ specific to each observation and parameters $\boldsymbol{\theta}$ which are shared between all observations. If we start with a model which assumes a generic prior over $\mathbf{z}$ (such as independent Bernoulli random variables), we can often fit more complex distributions by replacing the generic prior with a more structured one. Such models are commonly termed "hierarchical" or "deep," but we will use the term *compositional* because it reflects the way in which they are constructed. Some examples of this sort of composition include:

- **Hierarchical models of images.** In a landmark paper, Olshausen and Field (1996) proposed a sparse coding model of natural images, whereby image patches are modeled as linear combinations of a small number of basis functions drawn from a larger dictionary. Sparse coding can be represented as a Gaussian scale mixture model, where each coefficient is the product of a scale variable and a unit Gaussian variable, and all of the scale variables and Gaussian variables are independent. Various hierarchical models have been proposed which relax the independence assumption on the scale variables, modeling the joint distribution as a tree graphical model (Wainwright et al., 2001) or a low-rank Gaussian (Karklin and Lewicki, 2008). Such models constitute the state-of-the-art for low-level image processing tasks (Portilla et al., 2003) and capture textural properties of images (Karklin and Lewicki, 2008).

- **Deep sigmoid belief networks.** Deep sigmoid belief networks (Neal, 1992) are an early example of a model used to learn multi-level feaure representations. A single-layer sigmoid belief net models an observation vector (such as an image of handwritten digit) as a superposition of multiple independent events such as strokes. Thus, the observations are encoded as a vector of binary hidden variables determining which events are present. In a single layer sigmoid belief net, the events are modeled as independent Bernoulli random variables; a two-layer belief net models the events themselves using a single-layer sigmoid belief net. This process can be repeated to construct deep sigmoid belief nets.

It is also possible to add structure to the parameters $\boldsymbol{\theta}$. This can lead to more efficient use of limited data by sharing structure between different parts of a model. Some examples of this strategy include:

- **Multitask learning.** For many classification tasks such as object detection, the set of categories follows a heavy-tailed distribution: a handful of categories appear many times in a given dataset, while a large number of categories appear only a few times (Spain and Perona, 2008). One may wish to share statistical structure between different categories so that the rare ones can be learned

from fewer examples; this general area is known as multitask learning. For instance, similar parameters may work well for detecting trucks, school buses, and sports cars. This assumption can be captured by placing a structured prior, such as a mixture of Gaussians, on the parameter vectors for detecting different categories (Salakhutdinov et al., 2011). Such a model can be regarded as the composition of a clustering model with a supervised learning model.

- **Hierarchical Dirichlet processes.** Dirichlet processes are a Bayesian nonparametric model used to fit mixture models with an unbounded number of components. A Dirichlet process is defined in terms of a *base distribution*, the distribution that new components are drawn from. A hierarchical Dirichlet process (HDP) (Blei et al., 2003) uses a Dirichlet process as the base measure for other Dirichlet processes. For instance, the distribution over words in a document can be modeled as a Dirichlet process whose base measure is a global Dirichlet process corresponding to a set of topics shared between all documents. This can be viewed as the composition of two Dirichlet processes.

While these two forms of compositionality have distinct motivations, there is no fundamental difference between them. In Chapter 3, we present a space of matrix decomposition models defined in terms of a context-free grammar whose production rules correspond to simple probabilistic models. Let $\mathbf{Y}$ denote an observation matrix, where each row corresponds to a single observation vector. The simplest matrix decomposition models assume a factorization $\mathbf{Y} \approx \mathbf{Z}\mathbf{\Theta}$, where $\mathbf{Z}$ is a matrix representing the latent variables and $\mathbf{\Theta}$ is a matrix representing the parameters. Either $\mathbf{Z}$ or $\mathbf{\Theta}$ can be recursively decomposed, thereby capturing both types of compositional model described above. In fact, our matrix decomposition grammar is symmetric with respect to rows and columns: the system would behave the same if the input dimensions were treated as data points, and vice versa. This grammar gives a compact way of describing compositional models, since composition maps directly to applying productions of the grammar.

Composition is not the only operator for combining models. For instance, Chapter 5 presents a space of Gaussian process kernel structures defined in terms of addition and multiplication of simple base kernels. However, model composition is extremely versatile, and a surprising variety of models can be constructed purely through the composition of a few simple motifs.

1.2 Bayesian model comparison

In order to perform model selection, we need to specify a criterion for comparing models. One such criterion is the marginal likelihood of the model, or $p(\mathcal{D}\,|\,\mathcal{M}_i)$, where $\mathcal{D}$ denotes the observed data and $\mathcal{M}_i$ denotes the model (Kass and Raftery, 1995). This is a principled criterion from a Bayesian perspective because, combined with a prior over models (such as a uniform prior), the marginal likelihood can be plugged into Bayes' Rule to compute the posterior over models:

$$p(\mathcal{M}_i\,|\,\mathcal{D}) = \frac{p(\mathcal{M}_i)\,p(\mathcal{D}\,|\,\mathcal{M}_i)}{\sum_j p(\mathcal{M}_j)\,p(\mathcal{D}|\mathcal{M}_j)}.$$

Comparing models with marginal likelihood is known as Bayesian model comparison, and the marginal likelihood is sometimes referred to as the Bayes factor.

Marginal likelihood is an appealing model selection criterion for several reasons. First, it manages the tradeoff between model complexity and the goodness of fit to the data. Integrating out the model parameters results in a sophisticated form of Occam's Razor which penalizes the complexity of the model itself, rather than the specific parameterization (Rasmussen and Ghahramani, 2001). Second, it is closely related to description length (Barron et al., 1998), a compression-based criterion for model selection. Finally, since the marginal likelihood can be decomposed into a product of predictive likelihoods, it implicitly measures a model's ability to make predictions about novel examples. For these reasons, marginal likelihood is often the model selection criterion of choice when it is available. It is widely used to compare Gaussian process models (Rasmussen and Williams, 2006) and Bayesian network structures (Teyssier and Koller, 2005), where either closed-form solutions or accurate, tractable approximations are available.

Marginal likelihood has been criticized as being overly sensitive to the choice of model hyperparameters (Kass and Raftery, 1995). However, a number of modifications have been proposed to deal with this issue, including intrinsic Bayes factors (Berger and Pericchi, 1996), fractional Bayes factors (O'Hagan, 1995), the evidence approximation (MacKay, 1999), and careful choices of prior which avoid favoring one model over another (Heckerman et al., 1995; Neal, 2001b). All of these techniques are best seen as extensions of marginal likelihood, and they involve similar computations.

The main difficulty in applying marginal likelihood is that it is intractable to compute for most models of interest. Implicitly, $p(\mathcal{D}\,|\,\mathcal{M}_i)$ requires marginalizing out all of the parameters and latent variables of the model, an extremely high-dimensional summation or integration problem. It is equivalent to computing a partition function,

a problem which is #P-hard for graphical models in general. Much of the technical novelty of this thesis involves algorithms for estimating marginal likelihood which exploit the compositional structure of the model space.

1.2.1 Learning abstract structure

In cognitive science, Bayesian model comparison has also served as a model of how humans learn abstract concepts. Cognitive scientists have long debated the question of nature vs. nurture: how much of our knowledge is innate, and how much must be inferred from the environment (Chomsky, 1980; Pinker, 2003)? The work of Kemp and Tenenbaum (2008) showed that, at least in principle, it is possible to choose from a fairly large space of structures given a realistic amount of data. They defined a space of 20 structural forms of graphs which could potentially be used to organize the entities of a given domain, such as grids, trees, and cylinders. The structures were fit to human similarity judgments for a variety of domains and evaluated using approximate marginal likelihood. The highest-scoring structures for a variety of real-world datasets corresponded closely to human intuition: colors were arranged in a ring, Supreme Court justices were arranged on a one-dimensional spectrum, and animals were placed in a tree structure. Bayesian model comparison techniques were similarly used to model the learning of other forms of abstract structure, such as causal structure (Goodman et al., 2009) and language (Perfors et al., 2011). Piantadosi et al. (2012) showed that the structure of the natural numbers could be inferred from small amounts of observational data using Bayesian model comparison with a generic prior over programs.

Interestingly, in those studies, it was often possible to identify the correct structure with far fewer observations than were needed to fill in the details of the structure. *I.e.*, even when only a subset of the attributes were observed, animals were grouped into a tree structure, but many of the animals were placed in odd locations within the tree. Goodman et al. (2009) dubbed this phenomenon the *blessing of abstraction*.

In principle, therefore, one way to approach machine learning would be to define a prior over a large number of probabilistic models, and simply condition on the observed data. If the blessing of abstraction idea is correct, such an approach would find a near-optimal predictor after a small number of observations, and then proceed to make nearly the same predictions as the optimal one. This is the idea behind Solomonoff induction (Solomonoff, 1964; Li and Vitanyi, 1997), an idealized model of inductive inference. The idea is to use a *universal* prior, or one which assigns positive probability mass to all computable generative models. (For instance, generating and executing random Turing machines is one example of a universal prior.)

In principle, Solomonoff induction is able to learn any computable generative model. Unfortunately, the posterior is itself intractable. Even the more constrained structure discovery systems outlined above each required a significant amount of implementational work, since they required developing probabilistic inference algorithms which were both generic and efficient. Furthermore, the approach of exhaustively evaluating every model in a given set is unlikely to scale to a large number of models, since evaluating a model is at least comparable in difficulty to fitting the model.

1.3 Recursive structure discovery

If it is infeasible to exhaustively evaluate a large set of models, how is it that humans are able to make scientific progress? While we undoubtedly use a wide array of heuristics, one strategy we often adopt is to look for meaningful representations underlying the data. These may correspond to hidden causes, or may simply be succinct ways of summarizing the observations. Then, by looking for patterns in these representations, additional structure often becomes apparent. This section outlines several examples where this recursive structure discovery strategy has successfully been applied computationally.

1.3.1 Equation discovery

One form of structure discovery is equation discovery, where one desires a simple set of equations to describe a (usually scientific) phenomenon. An early example was BACON, a system developed by Langley et al. (1984) as a computational model of the formulation of scientific theories. The system was equipped with operators for detecting constant functions, linear relationships, and inverse relationships. By applying these operators *recursively*, it was able to uncover some complex relationships. In particular, after fitting a law to a subset of the variables, it would postulate *theoretical quantities* corresponding to the parameters of that law. It would then attempt to uncover patterns relating the theoretical quantities to the existing variables and to each other.

Consider the example of discovering the ideal gas law, which can be stated as $PV = 8.32N(T - 273)$, where P is the pressure on the gas, V is the volume, T is the temperature in degrees Celsius, and N is the quantity of gas in moles. Assume the system is able to run experiments where it chooses the values of enough variables to determine the equation, and then observes the value of the remaining variable. It begins by fixing N and T to arbitrary values and measuring V as a function of P. For each setting of N and T, it finds an inverse relationship between P and V, which

can be written as:

$$V = \alpha(N, T)P^{-1}.$$

It then treats α as a theoretical quantity, and it attempts to find relationships between α and the other variables the same way that it looks for relationships between observables. *I.e.*, holding N fixed, it varies T and measures α. It finds the linear relationship

$$\alpha(N, T) = b(N)T + c(N).$$

Repeating this process again, it finds $b(N) = 8.32N$ and $c(N) = -2271.4N$, so $\alpha(N, T) = 8.32NT - 2271.4N = 8.32N(T - 273)$. Rearranging terms recovers the ideal gas law. (In principle, the recursion goes deeper: while BACON treated quantities like pressure and temperature as observables, they began as theoretical quantities postulated to explain other scientific observations.)

This heuristic was taken a step further by the FAHRENHEIT system (Koehn and Zytkow, 1987; Langley and Zytkow, 1989), which attempted to determine the range in which a given law holds. The endpoints of the interval were treated as theoretical quantities. The algorithm recursively looked for laws which held between these quantities, and determined the range of values for which those laws held. What is remarkable about both systems is how much could be accomplished through recursive application of a handful of simple and seemingly innocuous rules.

1.3.2 Graphical model structure learning

In modern machine learning, the phrase "structure learning" is most closely associated with structure learning in graphical models. In the most basic formulation, one is given a set of variables and some samples from a distribution, and the task is to determine the pattern of edges in the graphical model which best matches the distribution. Equivalently, one wishes to determine the conditional independence structure of the model. When all of the variables are fully observed, it is possible to define a score function, such as the marginal likelihood of the graph structure, and optimize the score function. For tree-structured graphical models, the Chow-Liu algorithm (Chow and Liu, 1968) returns the optimal tree structure. For more general structures, a search procedure is required. For instance, in fully observed directed models, one can sample from the posterior over graph structures by sampling orderings of the variables (Teyssier and Koller, 2005). In the undirected case, a sparse pattern of edges can be recovered by optimizing the model likelihood with an ℓ_1 regularization penalty on the edge weights (Lee et al., 2006). When some of the variables are missing some of the time, they can be imputed using the structural

EM algorithm (Friedman, 1997).

The situation is more difficult when some variables are *latent*, or *never* observed. Structural EM is not applicable, since if a latent variable starts out disconnected from the rest of the network, the E step will leave it uncorrelated with the rest of the variables. Most of the focus has been on heuristics for postulating latent causes; these heuristics can also be used to initialize the structural EM algorithm (Elidan et al., 2000). One particularly relevant instance is the work of Harmeling and Williams (2011), who used a recursive procedure to incrementally grow forest-structured directed models. In each step, they would take the current set of root variables, and add a common parent to the pair of variables with the highest mutual correlation. This greedy approach to structure learning performed as well as a previous system which had a full set of search operators. Thus, as with BACON and FAHRENHEIT, the learned latent variables were treated on equal footing with the original observables.

1.3.3 Deep learning

Training deep neural networks is difficult because the responses of the network are highly nonlinear in the network parameters. If one applies gradient descent naively using backpropagation, the gradients can explode or die off as the error signals are backpropagated (Bengio et al., 1994). A major advance in applying neural networks was the idea of generative pre-training (Hinton and Salakhutdinov, 2006): first learning a neural network which models the data distribution, and then using that network as an initialization for the discriminative model.

Generative pre-training has been shown to have two advantages: it serves as a regularizer which attenuates overfitting, and it helps with the optimization (Erhan et al., 2009). We focus on the latter effect. It was observed by Hinton et al. (2006) that a particular kind of generative neural net, called *deep belief nets* (DBN), could be trained in a layerwise manner. First, one fits a single-layer generative neural network called a *restricted Boltzmann machine* (RBM) to the observed data. The hidden units of the RBM are then treated as input data, and a higher-level RBM is fit to those. By fitting RBM models recursively, one obtains a deep belief network, where the higher layers capture increasingly abstract properties of the data. The same strategy has been used to train other deep generative architectures, such as deep Boltzmann machines (DBMs) (Salakhutdinov and Hinton, 2009) and hierarchical-deep models (Salakhutdinov et al., 2013). For training the latter model, one first trains a DBM, and then fits a hierarchical Dirichlet process using the top layer representations as the inputs. It is a composition of two compositional models!

1.3.4 Discussion

The preceding examples are cases where one can find surprisingly complex structures purely through the recursive application of simpler pattern discovery methods. Taken separately, they likely appear to be one-off tricks for exploring particular search spaces. But in combination, the similarity is striking: if one is able to find meaningful representations of the data, one can often find additional patterns in those representations. We would argue that much of modern machine learning research follows the same pattern. One researcher may fit a simple model, with the aim of uncovering meaningful parameters and latent variables (the machine learning analogues of theoretical quantities). Then somebody else may notice additional structure in the parameters or latent variables and add that to the model, possibly uncovering additional meaningful representations in the process. The recursive nature of probabilistic modeling research is less obvious than in the preceding examples, because complex generative models are often built incrementally by multiple research teams over the course of many conference publications. However, Chapter 4 will show that an analogous recursive procedure can successfully identify many of the models currently used in the field of machine learning.

1.4 A toy example

We now discuss a toy example where the "obvious" set of reasoning steps can be viewed as recursive structure discovery in a compositional model space using marginal likelihood as the criterion. This is not a computationally challenging example, and we don't claim that a purely recursive approach would solve all problems of this form. Still, the intuitions should extend to those cases where the individual steps are difficult enough to require a computer. (Tenenbaum (1999) discussed the modeling of sequences using Bayesian model comparison in are more systematic way.)

Suppose we are given the number sequence

$$4,\ 13,\ 38,\ 87,\ 208,\ \ldots,$$

and we wish to predict the next term in the sequence. In the absence of an obvious pattern, one common trick is to take successive differences. If we include the first term, this gives us

$$4,\ 9,\ 25,\ 49,\ 121$$

All of these terms are squares! When we take square roots, we get:

$$2,\ 3,\ 5,\ 7,\ 11.$$

These are the first five prime numbers. So there we have it. The nth term in the sequence is simply the sum of squares of the first n primes. The next term is 377.

We can describe this search process as exploring a sequence of expressions of increasing specificity. If we use MATLAB-like notation and let R denote "I don't know where this came from," the models can be written as:

```
R
cumsum(R)
cumsum(square(R))
cumsum(square(nthprime(R)))
cumsum(square(nthprime(1:5)))
```

If we were to define a contex-free grammar for this language, it may include productions such as the following:

$$\begin{aligned}
\mathsf{R} &\longrightarrow \texttt{cumsum}(\mathsf{R}) \\
\mathsf{R} &\longrightarrow \texttt{square}(\mathsf{R}) \\
\mathsf{R} &\longrightarrow \texttt{nthprime}(\mathsf{R}) \\
\mathsf{R} &\longrightarrow 1{:}N
\end{aligned}$$

(The implementation of a programming language would not include particular function names in the CFG itself. Conceptually, however, we can think of these as rules in the grammar.) Each step of reasoning above is licensed by one of these productions. Each production corresponds to an *explanation* of where a particular sequence came from. Applying the production may yield a new sequence, analogous to latent variables or theoretical quantities, and one can then look for patterns in that sequence.

How do we know if we are "getting warmer?" Maybe experience plays a role, but our intuitive judgments can be explained from basic principles. If we attach a probabilistic semantics to the expressions, each step can be viewed as refining a generative model to one with higher marginal likelihood. Since the functions in our mini-language are all deterministic, we simply need to describe a generative process corresponding to the ignorance term R. We will let R denote the following procedure:

1. Sample two integers a and b between 1 and 1000 independently from the distribution $\Pr(a = n) = \Pr(b = n) \propto 1/n$. (This is commonly used as an uninformative prior over integers.)

2. Sample N numbers uniformly between a and b, inclusive (where N is the length of the sequence).

This distribution captures the intuition that explanations involving small numbers are better than ones involving large numbers, and explanations involving a small range of numbers are better than those involving a wide range. The prior over expressions is a probabilistic context-free grammar (PCFG) where in each step, the choice is uniform over all four productions, plus stopping.

We can score each expression $\mathcal{E}$ with $p(\mathcal{E}, \mathcal{S}) = p(\mathcal{E})\, p(\mathcal{S} \,|\, \mathcal{E})$, where $\mathcal{S}$ is the observed sequence. The term $p(\mathcal{S} \,|\, \mathcal{E})$ is the marginal likelihood of the expression, and the random choices to be marginalized out are the bounds a and b.

The prior probability of the trivial expression R is $1/5$, and the marginal likelihood $p(\mathcal{S} \,|\, \mathcal{E})$ is 3.94×10^{-14}, for a total score of 7.87×10^{-15}. In the second step, 3 of the 4 possible successor expressions cannot generate the data, so their scores are all zero. The expression $\texttt{cumsum}(\mathsf{R})$ generates the data if the R process generates the values $4, 9, 25, 49, 121$. The probability of this happening is 6.05×10^{-13}, and when combined with the prior probability of $1/5^2$, the score for this expression is 2.42×10^{-14}. This is only a moderate improvement in the score, which corresponds with the intuition that taking differences wasn't obviously the right thing to do.

In the next step, there are two successor expressions which generate $\mathcal{S}$ with nonzero probability. The first one, $\texttt{cumsum}(\texttt{cumsum}(\mathsf{R}))$, corresponds to taking differences again. Using the same calculations as before, the score for this expression is 6.73×10^{-14}, roughly a factor of 3 improvement. Now consider the alternative expression, $\texttt{cumsum}(\texttt{square}(\mathsf{R}))$, which corresponds to noticing that the integers were all perfect squares. In this case, the "uniformly generated" sequence is $2, 3, 5, 7, 11$, which is much more probable than the previous one, *i.e.* 9.72×10^{-8}, as compared with 6.05×10^{-13}. (Recall that our generative process for R favors numbers that are small and/or clumped together.) The total score of this expression is $1/5^3\,(9.72 \times 10^{-8}) = 7.77 \times 10^{-10}$, roughly a 3200-fold improvement. This corresponds to how we considered reaching a sequence of perfect squares to be a very promising find.

This process continues until we arrive at the final answer, $\texttt{cumsum}(\texttt{square}(\texttt{nthprime}(1{:}5)))$, in two more steps. For convenience, The following table reflects the expressions considered, as well as their score and the improvement over the previous step:

Expression	Score	Improvement
R	7.88×10^{-15}	—
cumsum(R)	2.42×10^{-14}	3.07
cumsum(square(R))	7.77×10^{-10}	3213.04
cumsum(square(nthprime(R)))	5.84×10^{-9}	7.51
cumsum(square(nthprime(1:5)))	0.0016	274186.18

This example illustrates how recursively looking for patterns can be interpreted as applying productions in a grammar in order to increase the marginal likelihood of the model. While this was a contrived example, the rest of the thesis will focus on algorithms for carrying out this sort of reasoning in a more realistic setting.

1.5 Outline

We now outline the overall structure of this thesis. **Chapter 2** provides some general background on several topics which are needed throughout the thesis. It introduces a variety of latent variable models which can be viewed as matrix decompositions and well as some MCMC algorithms which can be used to perform inference in these models. Finally, the chapter gives an overview of the problem of estimating partition functions and summarizes a variety of algorithms which have been applied to the problem.

In this introduction, we've spoken in general terms about the compositional structure of probabilistic models. **Chapter 3** makes this notion more concrete by introducing a space of matrix decomposition models, where an observation matrix is represented in terms of sums and products of matrices drawn from a few simple priors. All of the models in this space can be written as algebraic expressions generated by a context-free grammar whose production rules correspond to simple motifs of probabilistic modeling. This grammar yields a concise notation for describing a wide variety of probabilistic models. We discuss a number of models from the literature which can be represented in this framework.

The main idea of the thesis, compositional structure search, is described in **Chapter 4**. We present a method for performing inference across a wide variety of matrix decomposition models using a handful of specialized inference algorithms corresponding to the productions of the grammar. We introduce a recursive structure search procedure where, in each step, the current best models are expanded by applying all production rules. Empirically, this procedure was typically able to determine the correct structure on synthetic data in low noise settings, and fell back to simpler models under heavy noise. It also recovered known or plausible structures on a variety of

real-world datasets, all using exactly the same code with no hand-tuned parameters.

While most of this thesis focuses on matrix decompositions, **Chapter 5** discusses another case where compositional structure search can recover meaningful structures: Gaussian process (GP) covariance kernels. (This chapter is taken from a paper on which I was the third author, but I include it with permission of the primary authors because it fits nicely with the overall theme.) We define a grammar over GP kernel structures where kernels are constructed as sums and products of a few simple base kernels. A structure search procedure analogous to the one from Chapter 4 was able to recover the kernel structures for synthetic data. The kernels learned from a variety of time series datasets yielded interpretable additive decompositions into structure occurring at different scales, and often allowed for more accurate extrapolation compared to simpler kernels.

The structure search procedure of Chapter 4 includes compositional approaches to inference, but predictive likelihood estimation requires an ad-hoc procedure. **Chapter 6** introduces compositional importance sampling (CIS), a purely compositional approach for estimating the marginal likelihood of a model. CIS requires only black box algorithms for posterior inference and marginal likelihood estimation in models corresponding to the production rules. We give a theoretical analysis of the performance of the estimator in the case of finding additional structure within a low-rank factorization. Under certain conditions, CIS yields accurate marginal likelihood estimates and posterior samples. This analysis should hopefully lead to criteria for determining which spaces of models are efficiently searchable using recursive inference and model selection procedures.

Implementing CIS requires algorithms for estimating marginal likelihoods of models corresponding to the production rules. This by itself is a difficult task, since it requires integrating out both the parameters and the latent variables. Compounding this difficulty, it is difficult even to measure the performance of the algorithms, since they return estimates of a scalar quantity whose true value is unknown. **Chapter 7** introduces a novel method for obtaining ground truth marginal likelihood values for simulated data from a generative model. We use this method to evaluate a wide variety of marginal likelihood estimators for three production rules of the grammar. These experiments should inform the choice of algorithms in the context of compositional structure search and model selection more generally.

Two of the three algorithms which were found to be competitive at marginal likelihood estimation—annealed importance sampling (AIS) and sequential Monte Carlo (SMC)—are homotopy methods which pass through a sequence of distributions bridging from a tractable initial distribution to an intractable target distribution. Drawing upon ideas from information geometry, **Chapter 8** gives a framework

for choosing alternative sequences of distributions. We propose a novel sequence of intermediate distributions for exponential family models, defined by averaging the moments of the initial and target distributions. This method performs well at estimating the partition functions of restricted Boltzmann machines, the building block of deep belief networks and deep Boltzmann machines.

This thesis is a first step towards a general model selection procedure which exploits the compositional nature of probabilistic models, but it opens up more questions and avenues of research than it closes off. **Chapter 9** ties together the different threads of this thesis and outlines a variety of ways to build upon the work presented here.

1.6 Notation

For reference, we list some notational conventions which are adopted throughout the thesis. Matrices are denoted by boldface capital letters (e.g. $\mathbf{A}$), vectors by boldface lowercase letters (e.g. $\mathbf{a}$), and scalars by plain letters (e.g. a). Symbols in the grammar are denoted with a sans-serif font (e.g. G).

Unless otherwise specified, we adopt the following conventions for matrix decomposition models. $\mathbf{Y}$ represents the observation matrix, and $\mathbf{X}$ denotes a matrix of latent Gaussian variables to which some operation is applied to produce $\mathbf{Y}$. (For instance, $\mathbf{Y}$ may be a binary matrix obtained by thresholding $\mathbf{X}$ at zero.) The letters N, D, and K denote the numbers of data points, input dimensions, and latent dimensions, respectively. For instance, a matrix decomposition $\mathbf{X} \approx \mathbf{UV}$, $\mathbf{U}$ is an $N \times K$ matrix and $\mathbf{V}$ is an $K \times D$ matrix. The variables i, j, and k index the data points, input dimensions, and latent components, respectively.

In models based on importance sampling, p denotes the normalized target distribution and q denotes the normalized proposal distribution. The respective unnormalized distributions are denoted f and g.

Chapter 2

Background

In this chapter, we introduce several models and algorithms which are used throughout the thesis. This chapter is not intended to be read linearly; references are given throughout the text to particular sections as they are needed.

2.1 Motifs of probabilistic modeling

In this section, I outline some examples of motifs which are commonly used to construct probabilistic models. All of the models described in this section are sometimes used as stand-alone models, but their power lies in the fact that they can be combined to construct even richer models.

What all of these models have in common is that the aim is to learn a *reprsentation* of the data, *i.e.* a mapping from observations to a space on which various mathematical operations can be performed. For instance, one might like to encode data points as integers, real vectors, or binary vectors. There are a vast number of choices beyond what we cover here; for instance, in hierarchical clustering, the goal is to represent the data in terms of a tree structure.

2.1.1 Linear dimensionality reduction

Often, one has a collection of high-dimensional data points and wants to find a lower dimensional representation of the data. This may be motivated by computational or statistical considerations: high-dimensional models may be computationally expensive to fit, or there may not be enough data to fit them accurately. Alternatively, one may believe that there are a small number of factors explaining most of the variation in the data, and that it is inherently interesting to figure out what those

factors are. In other situations, such as recommendation systems, one might have a large, sparsely observed matrix and desire to make predictions about the missing entries. These situations motivate the use of a dimensionality reduction algorithm, where the data matrix is approximated with a low rank one.

Perhaps the simplest such algorithm is principal component analysis (PCA). In PCA, we try to find a low-dimensional subspace such that the projection of the data onto the subspace captures as much of the variance as possible. A basis for the optimal K-dimensional subspace is given in terms of the top K eigenvectors of the empirical covariance matrix $\mathbf{\Sigma}$. Because $\mathbf{\Sigma}$ is symmetric, the eigendecomposition is equivalent to the singular value decomposition SVD of $\tilde{\mathbf{Y}}$, which denotes the observation matrix $\mathbf{Y}$ with the mean of each column subtracted out. Recall that the SVD is a decomposition

$$\tilde{\mathbf{Y}} \approx \mathbf{U}\mathbf{D}\mathbf{V}^T,$$

where $\mathbf{U}$ and $\mathbf{V}$ have orthogonal columns and $\mathbf{D}$ is diagonal. The optimal subspace is obtained by keeping only the rows or columns corresponding to the K largest singular values. In this way, the SVD yields a low-rank approximation of the data matrix. Since the rows of $\mathbf{U}$ correspond to the data points, each row of $\mathbf{U}$ can be interpreted as a representation of that data point as a real-valued K-dimensional vector.

PCA has also been formulated as a probabilistic model where the maximum likelihood parameters can be computed in terms of an SVD (Tipping and Bishop, 1999; Roweis, 1998). Another closely related model is factor analysis (Basilevsky, 1994), which is similar to PCA, except that the Gaussian noise is assumed to be diagonal rather than spherical. This is advantageous, because the noise variance may differ for different input dimensions. Furthermore, if different inputs are expressed in different (arbitrary) units, it is meaningless to assume the variance is shared between different dimensions. Still another variant is probabilistic matrix factorization (PMF; Salakhutdinov and Mnih, 2008), a low rank approximation of a sparsely observed input matrix, which was a strong contender in the Netflix competition.

It should be noted that the algorithms listed above are *linear* dimensionality reduction algorithms. There has also been much work on nonlinear dimensionality reduction algorithms, which fit low-dimensional embeddings of the data points which are not constrained to be linear. Notable examples include multidimensional scaling (Shepard, 1980), the Gaussian process latent variable model (Lawrence, 2005), and deep autoencoders (Hinton and Salakhutdinov, 2006). In this thesis, dimensionality reduction will always refer to linear dimensionality reduction.

2.1.2 Clustering

One of the most common tasks in machine learning is clustering: dividing a set of data points into groups, where the data points within a group are somehow more similar than data points in different groups. In other words, each data point is assigned a discrete label $z_i \in \{1, \ldots, K\}$, where K is the number of clusters, and the ordering of the cluster indices is arbitrary.

Within probabilistic modeling, the most common approach to clustering is mixture modeling. Each cluster k is assumed to have an associated distribution $p(\mathbf{y}_i|\boldsymbol{\theta}, z_i = k) = p_{\boldsymbol{\theta}_k}(\mathbf{y}_i)$, where $p_{\boldsymbol{\theta}}$ is a parametric family of distributions parameterized by a vector $\boldsymbol{\theta}$. If the data are real-valued, $p_{\boldsymbol{\theta}}$ is often taken to be a Gaussian distribution, while if the data are binary valued, it is often taken to be the parameters of independent Bernoulli distributions, one for each input dimension. Fitting a clustering model involves fitting both the cluster parameters $\boldsymbol{\theta}_k$ and the mixture probabilities $\pi_k = p(z_i = k)$. An approximate maximum likelihood solution can be found using the Expectation-Maximization (EM) algorithm (Dempster et al., 1977).

In the Bayesian modeling approach, we place priors over both $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)^T$ and $\boldsymbol{\theta}$. Most commonly, the priors are chosen to be conjugate: the posterior distribution given the data and the latent assignments should have the same form as the prior. The prior over mixture probabilities $p(\boldsymbol{\pi})$ is usually taken to be a Dirichlet distribution because it is conjugate to the multinomial distribution. In the case of a Gaussian mixture model, the Gaussian distributions are parameterized by a mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, and the corresponding conjugate prior is a normal-inverse-Wishart distribution. For simplicity, in this thesis, the covariance matrices are assumed to be diagonal, and the mean and variance parameters have Gaussian and inverse gamma priors, respectively. The full generative model, then, is given by:

$$
\begin{aligned}
\boldsymbol{\pi} &\sim \text{Dirichlet}(\alpha) \\
s^2 &\sim \text{InverseGamma}(a, b) \\
\boldsymbol{\mu}_k &\sim \mathcal{N}(\mathbf{0}, s^2\mathbf{I}) \\
\sigma_{kj}^2 &\sim \text{InverseGamma}(a, b) \\
z_i &\sim \text{Multinomial}(\boldsymbol{\pi}) \\
\mathbf{y}_i &\sim \mathcal{N}(\boldsymbol{\mu}_{z_i}, \text{diag}(\sigma_{z_i 1}^2, \ldots, \sigma_{z_i D}^2))
\end{aligned}
$$

The hyperparameters α, a, and b need to be either specified or fit.

The above discussion assumes the number of clusters K is specified in advance. Bayesian nonparametrics (Ghahramani, 2012) is a branch of Bayesian statistics which places distributions on structures of unbounded complexity, so that the complexity of

the model can be adjusted automatically to match the data. The Bayesian nonparametric analogue of the Dirichlet-multinomial distribution is the Chinese restaurant process (CRP; Aldous, 1985), a prior over partitions of a set. (A partition is like a labeling, except that all permutations of the labels are treated as equivalent). The distribution is given in terms of the following analogy: consider a Chinese restaurant with an infinite number of tables. The customers enter the restaurant one at a time. If there are K tables with custoemers seated at them, the next customer chooses a given table k with probability $N_k/(N+\alpha)$, and sits at an empty table with probability $\alpha/(N+\alpha)$, where N is the total number of customers, N_k is the number seated at table k, and α is a concentration parameter. Larger values of α favor splitting the people between a larger number of tables. While this is not obvious from the culinary metaphor, the distribution is exchangeable: it does not matter in what order the customers enter the restaurant.

2.1.3 Binary attributes

Clustering is a fairly restrictive model, because each data point is assumed to belong to only one cluster, and the clusters are assumed to be unrelated to each other. In many domains, the natural categories may overlap; for instance, in modeling a social network, we may assume that individuals have multiple interests, such as tennis and programming. In this case, we would want to model the data in terms of overlapping subsets. Equivalently, each data point would be represented with a K-dimensional binary vector, where each of the entries denotes membership in one of the categories. A classic algorithm for learning binary vector representations is cooperative vector quantization (Zemel and Hinton, 1994). Most commonly, the binary attributes are assumed to be independent, so the vector describing each data point is a vector of independent Bernoulli random variables with parameters $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$.

Since data points which share a binary attribute may be similar in some ways but not others, defining the full model requires specifying how the distribution over observations depends on the binary attributes. If the observations $\mathbf{y}_i$ are real-valued vectors, the $\mathbf{y}_i$ are most commonly assumed to depend linearly on $\mathbf{z}_i$. In other words,

$$\mathbf{y}_i \sim \mathcal{N}\left(\mathbf{b} + \sum_{k=1}^{K} z_{ik}\mathbf{a}_k,\ \sigma_n^2\mathbf{I}\right),$$

where $\mathbf{b}$ is a bias vector, $\mathbf{a}_k$ is a real-valued feature vector associated with component k, and σ_n^2 is a noise parameter. While linearity is a fairly strong assumption, it is a convenient assumption algorithmically because it ensures conjugacy.

To build a Bayesian model, we place priors on the model parameters. The conjugate prior for the Bernoulli parameter π_k is a beta distribution. It is common to assume that the features $\mathbf{a}_k$ are all Gaussian distributed with spherical covariance. The Bayesian model, then, is given by:

$$\begin{aligned}
\pi_k &\sim \text{Beta}(\alpha, \beta) \\
\sigma_f^2, \sigma_n^2, \sigma_b^2 &\sim \text{InverseGamma}(a, b) \\
\mathbf{b} &\sim \mathcal{N}(\mathbf{0}, \sigma_b^2 \mathbf{I}) \\
\mathbf{a}_k &\sim \mathcal{N}(\mathbf{0}, \sigma_f^2 \mathbf{I}) \\
z_{ik} &\sim \text{Bernoulli}(\pi_k) \\
\mathbf{y}_i &\sim \mathcal{N}\left(\mathbf{b} + \sum_{k=1}^{K} z_{ik}\mathbf{a}_k,\ \sigma_n^2 \mathbf{I}\right)
\end{aligned}$$

If we regard the binary attribute vectors $\mathbf{z}_i$ as rows of a matrix $\mathbf{Z}$ and the real-valued features $\mathbf{a}_k$ as rows of a matrix $\mathbf{A}$, the observation model can be written in terms of a matrix factorization:

$$\mathbf{Y} \approx \mathbf{ZA}.$$

As with clustering, when the true number of components K is not known in advance, it is possible to use a Bayesian nonparametric analogue of the above model. The Indian Buffet Process (IBP; Griffiths and Ghahramani, 2005) is a prior over infinite sparse binary matrices. This model can be obtained as the limit of a sequence of distributions over binary matrices by letting $K \to \infty$ and simultaneously varying the parameters of the beta prior over π_k so that the expected number of nonzero entries in each row is held fixed. In order for the prior to be meaningful, the columns need to be reordered into a canonical order.

As with the CRP, the IBP can be described using a restaurant analogy. There is a buffet with an infinite number of dishes. When each customer enters the buffet, for each of the dishes sampled by previous customers, he samples it with probability $N_+/(N_+ + N_- + 1)$, where N_+ is the number of previous customers who chose the dish and N_- is the number of previous customers who did not. He then samples Poisson(α) new dishes. The customers correspond to rows of $\mathbf{Z}$ and the dishes correspond to columns, and the entry $z_{ik} = 1$ if customer i samples the kth dish. As in the CRP, the model is exchangeable, in that it does not matter in what order the customers entered the buffet.

2.1.4 Linear dynamics

All of the models discussed so far have been exchangeable: the distribution over datasets is invariant to permuting the order of the data points. When analyzing time series data, however, the system often evolves continuously through time, and one desires to model the dynamics. One basic model which can be used is a linear-Gaussian state space model, or linear dynamical system. Each data point is associated with a real-valued state $\mathbf{z}_t$ which evolves through time:

$$\mathbf{z}_t = \mathbf{A}_t \mathbf{z}_{t-1} + \boldsymbol{\epsilon}_t,$$

where $\boldsymbol{\epsilon}_t$ is Gaussian distributed system noise. Similarly to the models discussed previously, the observations at each time step are taken to be a Gaussian distribution whose mean is a linear function of the state variables:

$$\mathbf{y}_t = \mathbf{C}_t \mathbf{z}_t + \boldsymbol{\delta}_t,$$

where $\boldsymbol{\delta}_t$ is Gaussian distributed observation noise. Even though the linearity assumption is not always realistic, linear dynamical systems are commonly used because if the model parameters are known, inference can be performed efficiently and exactly using Kalman filtering and smoothing. If the parameters are not known, they can be fit using an EM procedure.

2.1.5 Sparsity

One drawback of linear dimensionality reduction algorithms such as PCA (Section 2.1.1) is that the individual coordinates of the real-valued vector representation may not correspond to anything meaningful. The reason is that PCA searches for a *subspace* which approximates the data, and there are many possible bases for the subspace. In particular, any rotation of the coordinate system is equally good at explaining the data. (In PCA, to break symmetry, it is common to use the top eigenvectors as the basis, which leads to meaningful coordinates in the case where one factor dominates the others. Only the first few coordinates are likely to be meaningful, though.) One way to find meaningful coordinates is to encourage *sparsity*: to encourage most of the components in the representation to be exactly zero, or close to zero. Unlike PCA, such a representation would not be rotationally invariant, since rotation destroys sparsity.

A heuristic motivation for sparse models is that in many domains, the causal structure is sparse: only a small subset of the causes are active in any particular

instance. For instance, in explaining a visual scene, only a small fraction of all possible edges are likely to be present. We may hope, therefore, that by learning a sparse representation of the inputs, we can find a representation which reveals something about the underlying causes.

The sparse coding model of Olshausen and Field (1996) was an early instantiation of this idea. They argued that an natural image patch $\mathbf{x}$ could be represented as a linear combination

$$\mathbf{x} = \sum_{k=1}^{K} s_k \mathbf{a}_k + \boldsymbol{\epsilon}, \tag{2.1}$$

where the $\mathbf{a}_k$ are dictionary elements, the s_k are coefficients which have a sparse distribution, and $\boldsymbol{\epsilon}$ is a Gaussian noise term. The prior $p(s_k)$ is taken to be a distribution which is heavy-tailed and peaked around zero, such as the Laplacian distribution or the student-t distribution. Depending on the choice of distributions and on how one infers the s_k, the representation may be exactly or approximately sparse. If $p(s_k)$ is any continuous distribution and the s_k are sampled from the posterior, the samples are all nonzero almost surely, but hopefully most are close to zero. On the other hand, if one computes the MAP estimate and $p(s_k)$ is nondifferentiable at zero, some of the coefficients may be exactly zero.

In the case of linear dimensionality reduction, one has to choose $K < \min(N, D)$, because otherwise the model can simply memorize the data. However, with sparse coding, it is possible, and often desirable, to learn an *overcomplete* representation, where $K > D$. As a heuristic motivation, in vision, the number of possible image events is larger than the number of pixels, since each one can appear in any location within the patch. Even though the representation is higher dimensional than the original data, it can still be more compact, because most of the coefficients are zero or close to zero, and therefore it takes fewer bits on average to approximate them.

When Olshausen and Field (1996) fit this model to natural image patches, they obtained a dictionary whose elements corresponded to localized, oriented edges similar to the receptive fields in the primary visual cortex. Since then, sparse models have been successful at uncovering meaningful representations of sensory data. Sparse restricted Boltzmann machines (RBMs; Lee et al., 2008) find more interpretable representations than standard RBMs, and a convolutional extension of sparse coding (Grosse, 2007) learns semantically meaningful features from speech spectrograms. The role of overcompleteness is still controversial: Berkes et al. (2008) showed that, in a Bayesian framework, highly overcomplete representations failed to assign higher likelihood to image patches.

2.1.6 Gaussian scale mixtures

The sparse coding model of Section 2.1.5 assumes the linear reconstruction coefficients are independent, but this assumption does not hold in natural images. While related coefficients are uncorrelated, their *magnitudes* are correlated: when one coefficient is far from zero, the other is more likely to be far from zero (Simoncelli, 1997). A class of models called *Gaussian scale mixture (GSM) models* (Wainwright et al., 2001) simultaneously captures both this effect and the sparse marginal distributions of the reconstruction coefficients.

In a single dimension, a GSM is a mixture of Gaussians with different scales, all centered at zero:

$$p(s) = \int p(\sigma)\,\mathcal{N}(s; 0, \sigma)\,\mathrm{d}\sigma,$$

where $p(\sigma)$ is any distribution over the positive reals. The Gaussian distribution is a special case of GSMs where $p(\sigma)$ is a delta function. However, by placing a broader prior over σ, we obtain a heavy-tailed distribution, since large values of σ correspond to s being far from zero. An equivalent representation of GSMs takes $s = rz$, where $r \sim \mathcal{N}(0, 1)$ and z is a random variable known as a *scale variable*. This alternative representation has the advantage that unlike σ, z need not be positive.

Surprisingly, any symmetric, zero-centered heavy-tailed distribution can be represented as a GSM (Wainwright et al., 2001), so GSMs are an alternative representation of a sparse coding model. If the coefficients s_k in (2.1) are independent GSMs, the model is exactly ordinary sparse coding. However, we can place a prior or a constraint on the corresponding scale variables z_k in order to model dependencies. In such a model, different coefficients s_k are uncorrelated, but their magnitudes may be correlated (or anticorrelated). Portilla et al. (2003) placed a (fixed) tree-based prior on the z_k to obtain a highly effective denoising algorithm for natural images. Karklin and Lewicki (2005) went a step further and *learned* the dependencies between the z_k, assuming there was an underlying low-dimensional representation. When they fit the resulting model to natural images, they found components which captured higher-level textural properties of a scene.

2.2 Markov chain Monte Carlo

With the exception of PCA (Section 2.1.1), none of the models described above can be fit in closed form, and some sort of approximation is required. In this thesis, we restrict ourselves to Bayesian models, and for the most part, inference is done using Markov chain Monte Carlo (MCMC). MCMC is a broad class of techniques

for approximately sampling from (unnormalized) probability distributions. In this section, we use $\mathbf{x}$ to denote a generic state variable and $p(\mathbf{x}) = f(\mathbf{x})/\mathcal{Z}$ to denote a generic distribution we are interested in sampling from. In most of this thesis, the particular task is posterior inference, *i.e.* the goal is to sample from the posterior $p(\boldsymbol{\theta}, \mathbf{z}|\mathbf{y})$, where $\boldsymbol{\theta}$ denotes the model parameters, $\mathbf{z}$ denotes latent variables, and $\mathbf{y}$ denotes the observed data.

The idea behind MCMC is to iteratively apply any number of *transition operators* $\mathcal{T}$, each one of which preserves the stationary distribution p, *i.e.*

$$\sum_{\mathbf{x}} p(\mathbf{x})\, \mathcal{T}(\mathbf{x}'|\mathbf{x}) = p(\mathbf{x}'). \tag{2.2}$$

Under certain regularity conditions, the distribution is guaranteed to converge to the stationary distribution p in the limit. Most often, MCMC transition operators are constructed out of *reversible* operators, *i.e.* ones which satisfy the *detailed balance conditions*

$$p(\mathbf{x})\, \mathcal{T}(\mathbf{x}'|\mathbf{x}) = p(\mathbf{x}')\, \mathcal{T}(\mathbf{x}|\mathbf{x}') \quad \text{for all } \mathbf{x}, \mathbf{x}'. \tag{2.3}$$

Intuitively, any reversible transition operator satisfies (2.2) because under p, the probability mass flowing in both directions between $\mathbf{x}$ and $\mathbf{x}'$ is equal. Not all MCMC transition operators are reversible, but most of the practical ones are constructed from sequences of reversible operators. While MCMC transition operators such as the ones described below are often presented as stand-alone algorithms, they are better thought of as operators which can be strung together in sequence to perform posterior inference. (If all transition operators in a set preserve a distribution p, then any sequence of them, or linear combination of them, must preserve it as well.)

The Metropolis-Hastings (M-H) procedure is a general formua for constructing reversible transition operators. In particular, assume we have a proposal operator $q(\mathbf{x}'|\mathbf{x})$. The M-H procedure first proposes a new state $\mathbf{x}'$ from q, and then computes the following acceptance probability:

$$r = \min\left\{1,\ \frac{p(\mathbf{x}')\, q(\mathbf{x}|\mathbf{x}')}{p(\mathbf{x})\, q(\mathbf{x}'|\mathbf{x})}\right\}. \tag{2.4}$$

With probability r, the proposal is accepted, *i.e.* $\mathbf{x}'$ is taken to be the new state; otherwise, the proposal is rejected and $\mathbf{x}$ is kept. M-H is extremely general, and most practical MCMC algorithms are special cases. Unfortunately, for many problems, unless the proposal distribution q is chosen very carefully, M-H is too slow to be

practical. Either the proposals take too small a step, or the acceptance probability is too small, or both.

Gibbs sampling is a special case of M-H which is often much more practical. In the proposal distribution, one variable x_i is sampled from its conditional distribution $p(x_i|\mathbf{x}_{-i})$ given the remaining variables $\mathbf{x}_{-i}$. Because the proposal itself is reversible, the M-H acceptance ratio is 1, so the proposal is always accepted. Gibbs sampling is widely used in probabilistic modeling because it has no tunable parameters and is often pretty effective.

2.2.1 Block Gibbs sampling

Sampling a single variable at a time from its conditional distribution can be inefficient, especially if the variables are tightly coupled. If two variables x_i and x_j are tightly coupled, one solution is to resample jointly from their conditional distribution $p(x_i, x_j|\mathbf{x}_{-ij})$. For instance, consider the probabilistic matrix factorization model (Salakhutdinov and Mnih, 2008), which has the factorization $\mathbf{Y} \approx \mathbf{UV}$ where $\mathbf{U}$ and $\mathbf{V}$ each have Gaussian priors. Each of the two factors can be sampled from its posterior as a block, because the conditional decomposes into independent factors for each row of $\mathbf{U}$ or column of $\mathbf{V}$. Computing these updates requires computing a dot product with dimensions $N \times K \times K$, and another with dimensions $D \times K \times K$, and inverting a single $K \times K$ matrix (shared between all rows of $\mathbf{U}$ or $\mathbf{V}$). Therefore, the time for computing the update is $O((N+D)K^2)$. This is desirable for the motivating application of PMF, collaborative filtering, where $K \ll \min(N, D)$, and the matrix is sparse, which allows the dot products to be computed efficiently.

2.2.2 Collapsed Gibbs sampling

Another solution to the problem of tightly coupled variables is to collapse out, or marginalize out, some of them analytically. When collapsed Gibbs sampling is feasible, it can often dramatically improve the mixing rate of the sampler. The drawback is that the collapsed updates can be trickier to implement and more computationally expensive. (Collapsed Gibbs sampling is sometimes called Rao-Blackwellized Gibbs sampling, but the name is somewhat misleading because typically, the primary benefit is improved mixing rather than the increased statistical efficiency that results from the Rao-Blackwell Theorem.)

An important example of collapsed Gibbs sampling is collapsing out the parameters $\boldsymbol{\theta}$ of a latent variable model. With $\boldsymbol{\theta}$ collapsed out, the goal is to sample $\mathbf{z}$ from the posterior $p(\mathbf{z}|\mathbf{y})$. Each $\mathbf{z}_i$ is sampled from its conditional distribution given the

remaining latent variables:

$$\begin{aligned} p(\mathbf{z}_i \,|\, \mathbf{z}_{-i}, \mathbf{y}) &\propto p(\mathbf{z}_i, \mathbf{y}_i \,|\, \mathbf{z}_{-i}, \mathbf{y}_{-i}) \\ &= p(\mathbf{z}_i \,|\, \mathbf{z}_{-i})\, p(\mathbf{y}_i \,|\, \mathbf{z}_i, \mathbf{z}_{-i}, \mathbf{y}_{-i}). \end{aligned} \tag{2.5}$$

Each of these terms may require integrating out parameters:

$$p(\mathbf{z}_i \,|\, \mathbf{z}_{-i}) = \int p(\boldsymbol{\theta}^{(\mathbf{z})} \,|\, \mathbf{z}_{-i})\, p(\mathbf{z}_i \,|\, \boldsymbol{\theta}^{(\mathbf{z})})\, \mathrm{d}\boldsymbol{\theta}^{(\mathbf{z})} \tag{2.6}$$

$$p(\mathbf{y}_i \,|\, \mathbf{z}_i, \mathbf{z}_{-i}, \mathbf{y}_{-i}) = \int p(\boldsymbol{\theta}^{(\mathbf{y}\,|\,\mathbf{z})} \,|\, \mathbf{z}_{-i}, \mathbf{y}_{-i})\, p(\mathbf{y}_i \,|\, \mathbf{z}_i, \boldsymbol{\theta}^{(\mathbf{y}\,|\,\mathbf{z})})\, \mathrm{d}\boldsymbol{\theta}^{(\mathbf{y}\,|\,\mathbf{z})}, \tag{2.7}$$

where $\boldsymbol{\theta}^{(\mathbf{z})}$ and $\boldsymbol{\theta}^{(\mathbf{y}\,|\,\mathbf{z})}$ are the parameters which determine the distribution over latent variables and the observation model, respecitvely.

If the update (2.5) is applied naïvely, the two predictive distributions (2.6) and (2.7) need to be recomputed for each $\mathbf{z}_i$. This is problematic, because each predictive distribution depends on all N data points, so recomputing it from scratch for each data point requires $O(N^2)$ time, which is impractical except for small datasets. Fortunately, for many models, the posteriors over $\boldsymbol{\theta}^{(\mathbf{z})}$ and $\boldsymbol{\theta}^{(\mathbf{y}\,|\,\mathbf{z})}$ are given in terms of sufficient statistics which can be updated during Gibbs sampling. The resulting algorithm often depends only linearly on N.

For example, in the finite clustering model of Section 2.1.2, $\boldsymbol{\theta}^{(\mathbf{z})}$ corresponds to the mixture probabilities $\boldsymbol{\pi}$, and the posterior is given by $p(\boldsymbol{\pi} \,|\, \mathbf{z}_{-i}) = \mathrm{Dirichlet}(\boldsymbol{\pi}; \alpha + N_1, \ldots, \alpha + N_K)$, where N_k is the number of data points assigned to cluster k. Similarly, the posterior over the cluster center $\boldsymbol{\mu}_k$ is given by:

$$p(\mu_{kj} \,|\, \mathbf{z}_{-i}, \mathbf{y}_{-i}) = \mathcal{N}\left(\mu_{kj};\ \frac{\sigma_{kj}^{-2} S_{kj}}{s^{-2} + N_k \sigma_{kj}^{-2}},\ \left(N_k \sigma_{kj}^{-2} + s^{-2}\right)^{-1/2}\right), \tag{2.8}$$

where $S_{kj} = \sum_{\mathbf{z}_{i'}=k} y_{i'j}$ denotes the sum of the data points belonging to a given cluster. Therefore, the statistics needed to compute the posteriors over $\boldsymbol{\pi}$ and $\boldsymbol{\mu}_k$ are the counts N_k for each cluster and the sums S_{kj}.

Sometimes implementing collapsed Gibbs sampling efficiently requires additional tricks. For instance, consider the binary attribute model of Section 2.1.3. In this case, $\boldsymbol{\theta}^{(\mathbf{z})}$ corresponds to the Bernoulli parameters $\boldsymbol{\pi}$, whose posterior can be updated analogously to the clustering case. The model parameters $\boldsymbol{\theta}^{(\mathbf{y}\,|\,\mathbf{z})}$, in particular the feature matrix $\mathbf{A}$ and the bias vector $\mathbf{b}$, are trickier. For simplicity, we leave out $\mathbf{b}$ in the following discussion, but it is easy to extend the following approach to collapse

out $\mathbf{A}$ and $\mathbf{b}$ jointly. The posterior over $\mathbf{A}$ decomposes into independent terms by column $\mathbf{a}_j$ (i.e. one for each dimension of the input matrix), and the individual distributions are given by:

$$p(\mathbf{a}_j \,|\, \mathbf{Z}_{-i,:}, \mathbf{Y}_{-i,j}) = \mathcal{N}(\mathbf{a}_j; \sigma_n^{-2}\mathbf{\Lambda}^{-1}\mathbf{Z}_{-i,:}^T\mathbf{Y}_{-i,j}, \mathbf{\Lambda}^{-1}),$$

where

$$\mathbf{\Lambda} = \sigma_f^{-2}\mathbf{I} + \sigma_n^{-2}\mathbf{Z}_{-i,:}^T\mathbf{Z}_{-i,:}.$$

We see that the posterior requires the quantities $\mathbf{Z}_{-i,:}^T\mathbf{Z}_{-i,:}$ and $\mathbf{Z}_{-i,:}^T\mathbf{Y}_{-i,j}$, which are a $K \times K$ matrix (shared between all dimensions j) and a K-dimensional vector (for each dimension j), and these need only be updated once for each row of $\mathbf{Z}$. Unfortunately, the $K \times K$ matrix $\mathbf{\Lambda}$ needs to be inverted once per row, for a total cost of DK^3.

This cost is still considerably faster than the $O(N^3)$ algorithm of the original IBP collapsed Gibbs sampler of Griffiths and Ghahramani (2005), but is still fairly inefficient. Doshi-Velez and Ghahramani (2009) pointed out that because resampling a single row of $\mathbf{Z}$ causes only a rank-one update to $\mathbf{\Lambda}$, the inverses can be updated efficiently using the Woodbury formula for a cost of $O(K^2)$ per update, or $O(NK^2)$ total. Because another step of the algorithm was $O(NKD)$, the total cost per Gibbs sweep was $O(NK(K+D))$, far more efficient than the original $O(N^3)$. This exemplifies how implementing a collapsed Gibbs sampler efficiently can take substantial attention to what order to resample variables in, which statistics to update, and how to update them efficiently.

A further benefit of collapsed Gibbs sampling beyond mixing speed is that it can be extended naturally to Bayesian nonparametric models. For instance, if one wants to sample from the posterior of a CRP clustering model, the number of cluster parameters which need to be explicitly represented depends on how many clusters are used to represent the data. Therefore, one must use MCMC techniques which allow moves between spaces of varying dimensionality, such as reversible jump MCMC (Green, 1995). On the other hand, if the cluster parameters are collapsed out, the dimensionalities of the remaining variables are fixed. In such situations, a collapsed Gibbs sampler can sometimes be *easier* to implement than an uncollapsed one.

2.2.3 Slice sampling

Gibbs sampling is an effective MCMC operator which requires no parameter tuning. Unfortunately, it requires the ability to sample exactly from the conditional distribu-

tions, something which is not always feasible. There are a variety of M-H operators which enable sampling from more general distributions, but most of them are sensitive to parameters of the proposal distribution, which makes them difficult to use in a structure search setting. Slice sampling (Neal, 2003) is an MCMC transition operator which is good for sampling from one-dimensional distributions, and which requires no parameter tuning.

Suppose $p(x) = f(x)/\mathcal{Z}$ is an unnormalized distribution over a scalar x. We can augment the distribution with an auxiliary scalar variable u, and rewrite the distribution as follows:

$$f'(x,u) = \begin{cases} 1 & \text{if } 0 \leq u \leq f(x) \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

Since $f'(x) = f(x)$, we can sample from f' and ignore u. This can be done using Gibbs sampling. In particular, sampling $u \sim p(u|x)$ simply requires sampling a real value from the interval $[0, f(x)]$. (This is typically done on a log scale for numerical stability.) Then, x is resampled from $p(x|u)$, the uniform distribution over the set $\{x : f(x) > u\}$. In this thesis, slice sampling is primarily used to resample hyperparameters.

2.3 Estimating normalizing constants

Because this thesis concerns model selection, there are many instances where we need to compute partition functions. This section gives an overview of some standard approaches to the problem. Throughout this section, $\mathbf{y}$ denotes the observed data, $\mathbf{z}$ the latent variables, and $\boldsymbol{\theta}$ the parameters.

2.3.1 General principles

The marginal likelihood (ML) of a dataset can be viewed as a partition function. Therefore, the same basic algorithms are relevant to ML estimation as to other partition function estimation problems, such as held-out likelihood estimation in directed and undirected models. However, these different problems present their own challenges and have their own requirements. In this section, we outline various criteria for evaluating partition function estimators and highlight ways in which ML estimation differs from other partition function estimation problems.

First consider the general partition function estimation problem. For convenience, assume the variables are discrete. Suppose we have a probability distribution $p(\mathbf{x}) =$

$f(\mathbf{x})/\mathcal{Z}$ defined in terms of an unnormalized *pmf* f and the partition function $\mathcal{Z} = \sum_{\mathbf{x}} f(\mathbf{x})$. A partition function estimator (possibly stochastically) returns an estimate $\hat{\mathcal{Z}}$ of $\mathcal{Z}$.

Most simply, we could evaluate the estimator in terms of its bias $|\mathbb{E}[\hat{\mathcal{Z}}] - \mathcal{Z}|$ and variance $\mathrm{Var}(\hat{\mathcal{Z}})$, just like any other estimator of a scalar quantity. Many partition function estimators, such as simple importance sampling (SIS) and annealed importance sampling (AIS; Neal, 2001a), are unbiased, *i.e.* $\mathbb{E}[\hat{\mathcal{Z}}] = \mathcal{Z}$. In this context, unbiasedness can be misleading: because partition function estimates can vary over many orders of magnitude, it's common for an unbiased estimator to drastically underestimate $\mathcal{Z}$ with overwhelming probability, yet occasionally return extremely large estimates. (An extreme example is likelihood weighting (Section 2.3.3), which is unbiased, but is extremely unlikely to give an accurate answer for a high-dimensional model.) Unless the estimator is chosen very carefully, the variance is likely to be extremely large, or even infinite.

Since $\mathcal{Z}$ can vary over many orders of magnitude, it is often more meaningful to look at $\log \mathcal{Z}$. Estimators of $\mathcal{Z}$ can be equivalently viewed as estimators of $\log \mathcal{Z}$. Unfortunately, unbiasedness does not carry over; unbiased estimators of $\mathcal{Z}$ may correspond to biased estimators of $\log \mathcal{Z}$. In particular, they are stochastic lower bounds. There are two arguments for this, neither of which subsumes the other. First, because ML estimators are nonnegative estimators of a nonnegative quantity, Markov's inequality implies that $\Pr(\hat{\mathcal{Z}} > a\mathcal{Z}) < 1/a$. By taking the log, we find that

$$\Pr(\log \hat{\mathcal{Z}} > \log \mathcal{Z} + b) < e^{-b}. \tag{2.10}$$

In other words, the estimator is exceedingly unlikely to overestimate $\log \mathcal{Z}$ by more than a few nats. This also implies that the underlying cause of large or infinite $\mathrm{Var}(\hat{\mathcal{Z}})$—that $\hat{\mathcal{Z}}$ occasionally takes on extremely large values—does not occur in the log domain.

The other argument that $\log \hat{\mathcal{Z}}$ is a stochastic lower bound of $\log \mathcal{Z}$ follows from Jensen's inequality:

$$\mathbb{E}[\log \hat{\mathcal{Z}}] \leq \log \mathbb{E}[\hat{\mathcal{Z}}] = \log \mathcal{Z}. \tag{2.11}$$

Therefore, unbiased estimators can be compared in terms of their bias in the log domain: $\log \mathcal{Z} - \mathbb{E}[\log \hat{\mathcal{Z}}]$. One convenient feature of this metric is that unbiased sampling-based estimators of $\log \mathcal{Z}$ can be compared directly against variational inference algorithms such as mean field, which also yield lower bounds on $\log \mathcal{Z}$. (As discussed in Section 7.2.3, the bias of simple importance sampling (SIS) in the log domain is closely related to the error in the variational approximation.)

2.3.2 Use cases

There are several model selection criteria which often require estimating high-dimensional integrals: held-out likelihood of directed models, held-out likelihood of undirected models, and marginal likelihood of directed models. At an abstract level, all of these problems are special cases of the general problem of computing partition functions, and therefore the same methods should be relevant to each. However, these different problems have different requirements for the estimators, and we discuss the differences in this section.

2.3.2.1 Held-out likelihood of directed models

The most basic way to evaluate a generative model is to fit the parameters $\boldsymbol{\theta}$ to training data and evaluate the model likelihood

$$\prod_{i=1}^{N_{test}} p(\mathbf{y}_i; \boldsymbol{\theta}) = \prod_{i=1}^{N_{test}} \sum_{\mathbf{z}_i} p(\mathbf{z}_i; \boldsymbol{\theta}) p(\mathbf{y}_i \,|\, \mathbf{z}_i; \boldsymbol{\theta}). \tag{2.12}$$

on held-out test data. For some models, such as factor analysis and Gaussian mixture models, the sum in (2.12) can be computed exactly. In other cases, such as topic models (Wallach et al., 2009), the integral is intractable and must be approximated. If the method of approximating the integral is a stochastic lower bound, it will err on the side of conservatism, assigning the model a lower likelihood score than it should. Therefore, one is unlikely to be deceived into believing the model is better than it actually is.

2.3.2.2 Held-out likelihood of undirected models

Held-out likelihood is also used to evaluate undirected models. In this case, the model is defined as $p(\mathbf{y}, \mathbf{z}; \boldsymbol{\theta}) \triangleq f(\mathbf{y}, \mathbf{z}; \boldsymbol{\theta})/\mathcal{Z}$, where f is an unnormalized distribution and $\mathcal{Z} = \sum_{\mathbf{y},\mathbf{z}} f(\mathbf{y}, \mathbf{z}; \boldsymbol{\theta})$ is a (generally unknown and intractable) partition function. In this case, the quantity to be computed is:

$$\prod_{i=1}^{N_{test}} p(\mathbf{y}_i; \boldsymbol{\theta}) = \prod_{i=1}^{N_{test}} \frac{\sum_{\mathbf{z}} f(\mathbf{y}_i, \mathbf{z}; \boldsymbol{\theta})}{\sum_{\mathbf{z},\mathbf{y}} f(\mathbf{y}, \mathbf{z}; \boldsymbol{\theta})}. \tag{2.13}$$

The numerator requires integrating over the latent variables. This is similar to likelihood evaluation in directed models, and does not obviously raise any new issues. Often, undirected models have a particular form which makes this integral tractable

(Salakhutdinov and Murray, 2008) or easy to approximate using variational methods (Salakhutdinov and Hinton, 2009).

What makes the undirected case different from the directed case is the partition function in the denominator. This is an intractable integral over all observed and latent variables, and can be difficult to compute when the model distribution is multimodal. Fortunately, the denominator is shared between all test examples, and therefore only needs to be computed once after the parameters are fit. More problematically, because the integral appears in the denominator, underestimates of the partition correspond to overestimates of the likelihood. Therefore, poor estimates of the partition function can lead one to conclude that the model is *far better* than it actually is. Evaluating a proposed undirected model generally requires extensive experiments to test the accuracy of the partition function estimates.

2.3.2.3 Marginal likelihood of directed models

Marginal likelihood refers to the probability of the observed data with all of the model parameters and latent variables integrated out:

$$p(\mathbf{y}) = \int p(\boldsymbol{\theta}) \prod_{i=1}^{N} \sum_{\mathbf{z}_i} p(\mathbf{z}_i \,|\, \boldsymbol{\theta})\, p(\mathbf{y}_i \,|\, \mathbf{z}_i, \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}. \tag{2.14}$$

Unlike in the case of held-out likelihood, only one integral needs to be computed. Unfortunately, because this integral includes all parameters and all latent variables, it can be more computationally demanding than computing held-out likelihood.

Like in the directed case, when variational methods and unbiased estimators give poor approximations, they err on the side of pessimism. Compared to the case of held-out likelihood, the accuracy requirements can be substantially lower. For instance, suppose an algorithm consistently underestimates the held-out likelihood by 1 nat. Then each test example is seen as e times less probable than it actually is; this is quite a significant difference. However, if an algorithm underestimates the marginal likelihood of an entire dataset by 1 nat, the difference is probably not even meaningful. Recall that in the above argument from Markov's inequality, in principle the estimator could overestimate the true value by a few nats with nonnegligible probability. In the case of marginal likelihood, however, a difference of a few nats is negligible, and we can therefore view unbiased estimators as lower bounds on the marginal likelihood.

2.3.3 Likelihood weighting

Partition function estimators are often constructed from simple importance sampling (SIS). In particular, suppose we wish to compute the partition function $\mathcal{Z} = \sum_{\mathbf{x}} f(\mathbf{x})$. We generate a collection of samples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(K)}$ from a proposal distribution $q(\mathbf{x})$ whose support contains the support of p, and compute the estimate

$$\hat{\mathcal{Z}} = \frac{1}{K} \sum_{k=1}^{K} w^{(k)} \triangleq \frac{1}{K} \sum_{k=1}^{K} \frac{f(\mathbf{x}^{(k)})}{q(\mathbf{x}^{(k)})}. \tag{2.15}$$

This is an unbiased estimator of $\mathcal{Z}$, because of the identity

$$\mathbb{E}_{\mathbf{x} \sim q} \left[\frac{f(\mathbf{x})}{q(\mathbf{x})} \right] = \mathcal{Z}. \tag{2.16}$$

Likelihood weighting is a special case of this approach where the prior is used as the proposal distribution. In particular, for estimating the held-out likelihood of a directed model, latent variables $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(K)}$ are sampled from $p(\mathbf{z}; \boldsymbol{\theta})$. By inspection, the weight $w^{(k)}$ is simply the data likelihood $p(\mathbf{y} \,|\, \mathbf{z}^{(k)}; \boldsymbol{\theta})$. This method can perform well if the latent space is small enough that the posterior can be adequately covered with a large enough number of samples. Unfortunately, likelihood weighting is unlikely to be an effective method for estimating marginal likelihood, because the model parameters would have to be sampled from the prior, and the chance that a random set of parameters happens to model the data well is vanishingly small.

2.3.4 The harmonic mean of the likelihood

The harmonic mean estimator of Newton and Raftery (1994) is another estimator based on SIS. Here, the posterior is used as the proposal distribution, and the prior as the target distribution. By plugging these into (2.16), we obtain:

$$\mathbb{E}_{\boldsymbol{\theta}, \mathbf{z} \sim p(\boldsymbol{\theta}, \mathbf{z} \,|\, \mathbf{y})} \left[\frac{p(\boldsymbol{\theta}, \mathbf{z})}{p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{y})} \right] = \frac{1}{p(\mathbf{y})}. \tag{2.17}$$

This suggests the following estimator: draw samples $\boldsymbol{\theta}^{(k)}, \mathbf{z}^{(k)}$ from the posterior $p(\boldsymbol{\theta}, \mathbf{z} \,|\, \mathbf{y})$, and compute weights $w^{(k)} = p(\boldsymbol{\theta}^{(k)}, \mathbf{z}^{(k)}) / p(\boldsymbol{\theta}^{(k)}, \mathbf{z}^{(k)}, \mathbf{y}) = p(\mathbf{y} \,|\, \boldsymbol{\theta}^{(k)}, \mathbf{z}^{(k)})$.

The ML estimate, then, is computed from the harmonic mean of the likelihood values:

$$\hat{p}(\mathbf{y}) = \frac{K}{\sum_{k=1}^{K} 1/w^{(k)}} = \frac{K}{\sum_{k=1}^{K} 1/p(\mathbf{y}\,|\,\boldsymbol{\theta}^{(k)}, \mathbf{z}^{(k)})}. \tag{2.18}$$

Note that the weights $w^{(k)}$ are unbiased estimators of the *reciprocal* of the marginal likelihood. Following the discussion of Section 2.3.1, this implies that they generally underestimate the reciprocal, or equivalently, overestimate the marginal likelihood. Therefore, unlike most of the other estimators discussed in this section, the harmonic mean estimator is generally overly optimistic. Neal (2008) recommended against using the estimator for this reason. (We emphasize that the harmonic mean estimator is only a stochastic upper bound if *exact* posterior samples are used. In practice, one must rely on an approximate sampler, and the estimator can still underestimate $p(\mathbf{y})$ if the sampler fails to explore an important mode.)

2.3.5 Annealed importance sampling

The problem with both likelihood weighting and the harmonic mean estimator is that each one is based on a single importance sampling computation between two very different distributions. A more effective method is to bridge between the two distributions using a sequence of intermediate distributions. Annealed importance sampling (AIS; Neal, 2001a) is one algorithm based on this idea, and is widely used for evaluating partition functions. Mathematically, the algorithm takes as input a sequence of T distributions $p_1, \ldots, p_T$, with $p_t(\mathbf{x}) = f_t(\mathbf{x})/\mathcal{Z}_t$, where p_T is the target distribution and p_1 is a tractable initial distribution, *i.e.* one for which we can efficiently evaluate the normalizing constant and generate exact samples. Most commonly, the intermediate distributions are taken to be geometric averages of the initial and target distributions: $f_t(\mathbf{x}) = f_1(\mathbf{x})^{1-\beta_t} f_T(\mathbf{x})^{\beta_t}$, where the β_t are monotonically increasing parameters with $\beta_1 = 0$ and $\beta_T = 1$.

The AIS procedure, shown in Algorithm 1, involves applying a sequence of MCMC transition operators $\mathcal{T}_1, \ldots, \mathcal{T}_T$, where $\mathcal{T}_t$ leaves p_t invariant. The result of the algorithm is a weight w which is an unbiased estimator of the ratio of partition functions $\mathcal{Z}_T/\mathcal{Z}_1$. Since $\mathcal{Z}_1$ is typically known, $\mathcal{Z}_1 w$ can be viewed as an unbiased estimator of $\mathcal{Z}_T = \mathcal{Z}$.

For purposes of evaluating marginal likelihood, f_1 is the prior distribution $p(\boldsymbol{\theta}, \mathbf{z})$, and f_T is the joint distribution $p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{y})$ (viewed as a function of $\boldsymbol{\theta}$ and $\mathbf{z}$). Because $\mathbf{y}$ is fixed, this is proportional to the posterior $p(\boldsymbol{\theta}, \mathbf{z}\,|\,\mathbf{y})$. The intermediate distributions are given by geometric averages of the prior and the posterior, which is equivalent

Algorithm 1 Annealed Importance Sampling

for $k = 1$ to K **do**
  $\mathbf{x}_1 \leftarrow$ sample from $p_1(\mathbf{x})$
  $w^{(k)} \leftarrow \mathcal{Z}_1$
  for $t = 2$ to T **do**
    $w^{(k)} \leftarrow w^{(k)} \frac{f_t(\mathbf{x}_{t-1})}{f_{t-1}(\mathbf{x}_{t-1})}$
    $\mathbf{x}_t \leftarrow$ sample from $\mathcal{T}_t(\mathbf{x} \,|\, \mathbf{x}_{t-1})$
  end for
end for
return $\hat{\mathcal{Z}} = \sum_{k=1}^{K} w^{(k)}/K$

to raising the likelihood term to a power less than 1:

$$f_t(\boldsymbol{\theta}, \mathbf{z}) = p(\boldsymbol{\theta}, \mathbf{z})\, p(\mathbf{y} \,|\, \boldsymbol{\theta}, \mathbf{z})^{\beta_t}. \tag{2.19}$$

Note that this form of annealing can destroy the directed factorization structure which is present in the prior and the joint distribution. Conditional independencies satisfied by the original model may not hold in the intermediate distributions. Unfortunately, this can make the implementation of MCMC operators for the intermediate distributions considerably more complicated compared to the analogous operators applied to the posterior.

AIS can be justified as an instance of SIS over an extended state space. In particular, the full set of states $\mathbf{x}_1, \ldots, \mathbf{x}_T$ visited by the algorithm has a joint distribution represented by:

$$q_{for}(\mathbf{x}_1, \ldots, \mathbf{x}_T) = p_1(\mathbf{x}_1)\, \mathcal{T}_2(\mathbf{x}_2 \,|\, \mathbf{x}_1) \cdots \mathcal{T}_{T-1}(\mathbf{x}_{T-1} \,|\, \mathbf{x}_{T-2}). \tag{2.20}$$

We can also postulate a reverse chain, where $\mathbf{x}_{T-1}$ is first sampled exactly from the distribution p_T, and the transition operators are applied in the reverse order. (Note that the reverse chain cannot be explicitly simulated in general, since it requires sampling from p_T.) The joint distribution is given by:

$$q_{back}(\mathbf{x}_1, \ldots, \mathbf{x}_T) = p_T(\mathbf{x}_{T-1})\, \mathcal{T}_{T-1}(\mathbf{x}_{T-2} \,|\, \mathbf{x}_{T-1}) \cdots \mathcal{T}_2(\mathbf{x}_1 \,|\, \mathbf{x}_2). \tag{2.21}$$

If q_{for} is used as a proposal distribution for q_{back}, the importance weights come out

to:

$$\frac{q_{back}(\mathbf{x}_1, \ldots, \mathbf{x}_T)}{q_{for}(\mathbf{x}_1, \ldots, \mathbf{x}_T)} = \frac{p_T(\mathbf{x}_{T-1})}{p_1(\mathbf{x}_1)} \frac{\mathcal{T}_2(\mathbf{x}_1 \,|\, \mathbf{x}_2)}{\mathcal{T}_2(\mathbf{x}_2 \,|\, \mathbf{x}_1)} \cdots \frac{\mathcal{T}_{T-1}(\mathbf{x}_{T-2} \,|\, \mathbf{x}_{T-1})}{\mathcal{T}_{T-1}(\mathbf{x}_{T-1} \,|\, \mathbf{x}_{T-2})} \tag{2.22}$$

$$= \frac{p_T(\mathbf{x}_{T-1})}{p_1(\mathbf{x}_1)} \frac{f_2(\mathbf{x}_1)}{f_2(\mathbf{x}_2)} \cdots \frac{f_{T-1}(\mathbf{x}_{T-2})}{f_{T-1}(\mathbf{x}_{T-1})} \tag{2.23}$$

$$= \frac{\mathcal{Z}_1}{\mathcal{Z}_T} w, \tag{2.24}$$

where (2.23) follows from the reversibility of $\mathcal{T}_t$. Since this quantity corresponds to an importance weight between normalized distributions, its expectation is 1, and therefore $\mathbb{E}[w] = \mathcal{Z}_T / \mathcal{Z}_1$.

Note also that $q_{back}(\mathbf{x}_{T-1}) = p_T(\mathbf{x}_{T-1})$. Therefore, AIS can also be used as an importance sampler for p_T:

$$\mathbb{E}_{q_{for}}[w h(\mathbf{x}_{T-1})] = \frac{\mathcal{Z}_T}{\mathcal{Z}_1} \mathbb{E}_{p_T}[h(\mathbf{x})] \tag{2.25}$$

for any statistic h. (The partition function estimator corresponds to the special case where $h(\mathbf{x}) = 1$.)

2.3.6 Sequential Monte Carlo

Observe that the marginal distribution $p(\mathbf{y})$ can be decomposed into a series of predictive distributions:

$$p(\mathbf{y}_{1:N}) = p(\mathbf{y}_1)\, p(\mathbf{y}_2 \,|\, \mathbf{y}_1) \cdots p(\mathbf{y}_N \,|\, \mathbf{y}_{1:N-1}). \tag{2.26}$$

Since the predictive likelihood terms can't be computed exactly, approximations are required. Sequential Monte Carlo (SMC) methods (del Moral et al., 2006) use particles to represent the parameters and/or the latent variables. In each step, as a new data point is observed, the particles are generally updated to take into account the new information. While SMC is most closely associated with filtering problems where there are explicit temporal dynamics, it has also been successfully applied to models with no inherent temporal structure, such as the ones considered in this work. This is the setting that we focus on here.

SMC is a very broad family of algorithms, so we cannot summarize all of the advances. Instead, we give a generic implementation in Algorithm 2 where several decisions are left unspecified. In each step, the latent variables are sampled according

to a proposal distribution q, which may optionally take into account the current data point. (More details are given below.) The weights are then updated according to the evidence, and the model parameters (and possibly latent variables) are updated based on the new evidence.

This procedure corresponds the most closely to the particle learning approach of Carvalho et al. (2010), where $\mathbf{z}$ is approximated in the typical particle filter framework, and $\boldsymbol{\theta}$ is resampled from the posterior after each update. Our formulation is slightly more general: since it may not be possible to sample $\boldsymbol{\theta}$ exactly from the posterior, we allow any MCMC operator to be used which preserves the posterior distribution. Furthermore, we allow $\mathbf{z}$ to be included in the MCMC step as well. Carvalho et al. (2010) do not allow this, because it would require revisiting all of the data after every sampling step. However, we consider it because it may be advantageous to pay the extra cost in the interest of more accurate results.

Algorithm 2 leaves open the choice of $q(\mathbf{z}_i \,|\, \mathbf{y}_i, \boldsymbol{\theta}^{(k)})$, the proposal distribution for the latent variables at the subsequent time step. The simplest method is to ignore the observations and sample $\mathbf{z}_i^{(k)}$ from the predictive distribution, *i.e.*

$$q(\mathbf{z}_i \,|\, \mathbf{y}_i, \boldsymbol{\theta}^{(k)}) = p(\mathbf{z}_i \,|\, \boldsymbol{\theta}^{(k)}). \tag{2.27}$$

The particles are then weighted according to the observation likelihood:

$$w^{(k)} \leftarrow w^{(k)} p(\mathbf{y}_i \,|\, \mathbf{z}_i^{(k)}, \boldsymbol{\theta}^{(k)}). \tag{2.28}$$

A more accurate method, used in the posterior particle filter, is to sample $\mathbf{z}_i^{(k)}$ from the posterior:

$$q(\mathbf{z}_i \,|\, \mathbf{y}_i, \boldsymbol{\theta}^{(k)}) = p(\mathbf{z}_i \,|\, \mathbf{y}_i, \boldsymbol{\theta}^{(k)}). \tag{2.29}$$

In this case, the weight update corresponds to the predictive likelihood:

$$w^{(k)} \leftarrow w^{(k)} p(\mathbf{y}_i \,|\, \boldsymbol{\theta}^{(k)}) \tag{2.30}$$

$$= w^{(k)} \sum_{\mathbf{z}_i} p(\mathbf{z}_i \,|\, \boldsymbol{\theta}^{(k)}) p(\mathbf{y}_i \,|\, \mathbf{z}_i, \boldsymbol{\theta}^{(k)}). \tag{2.31}$$

Posterior particle filtering can result in considerably lower variance of the weights compared to standard particle filtering, and therefore better marginal likelihood estimates. For some models, such as clustering (Section 2.1.2) and factor analysis (Section 2.1.1), these computations can be performed exactly. For other models, such as binary attribute models (Section 2.1.3), it is intractable to sample exactly

Algorithm 2 Particle learning

for $k = 1$ to K **do**
  $\boldsymbol{\theta}^{(k)} \leftarrow$ sample from $p(\boldsymbol{\theta})$
  $w^{(k)} \leftarrow 1$
end for
for $i = 1$ to T **do**
  for $k = 1$ to K **do**
    $\mathbf{z}_i^{(k)} \leftarrow$ sample from $q(\mathbf{z}_i \,|\, \mathbf{y}_i, \boldsymbol{\theta}^{(k)})$
    $w^{(k)} \leftarrow w^{(k)} p(\mathbf{z}_i^{(k)} \,|\, \boldsymbol{\theta}) p(\mathbf{y}_i \,|\, \mathbf{z}_i^{(k)}, \boldsymbol{\theta}^{(k)}) / q(\mathbf{z}_i \,|\, \mathbf{y}_i, \boldsymbol{\theta}^{(k)})$
    $(\mathbf{z}_{1:i}^{(k)}, \boldsymbol{\theta}^{(k)}) \leftarrow$ MCMC transition which leaves $p(\mathbf{z}_{1:i}, \boldsymbol{\theta} \,|\, \mathbf{y}_{1:i})$ invariant
  end for
  if resampling criterion met **then**
    Sample $(\mathbf{z}_{1:i}^{(k)}, \boldsymbol{\theta}^{(k)})$ proportionally to $w^{(k)}$
    $S \leftarrow \sum_{k=1}^{K} w^{(k)}$
    for $k = 1$ to K **do**
      $w^{(k)} \leftarrow S/K$
    end for
  end if
end for
return $\hat{\mathcal{Z}} = \frac{1}{K} \sum_{k=1}^{K} w^{(k)}$

from the posterior or to compute the sum in (2.31).

For simplicity of notation, Algorithm 2 explicitly samples the model parameters $\boldsymbol{\theta}$. However, for models where $\boldsymbol{\theta}$ has a simple closed form depending on certain sufficient statistics of $\mathbf{y}$ and $\mathbf{z}$, it can be collapsed out analytically, giving a Rao-Blackwellized particle filter. The algorithm is the same as Algorithm 2, except that steps involving $\boldsymbol{\theta}$ are ignored and the updates for $\mathbf{z}$ and w are modified:

$$\mathbf{z}_i^{(k)} \leftarrow \text{sample from } q(\mathbf{z}_i^{(k)} \,|\, \mathbf{y}_{1:i}, \mathbf{z}_{1:i-1}^{(k)}) \tag{2.32}$$

$$w^{(k)} \leftarrow w^{(k)} \frac{p(\mathbf{z}_i^{(k)} \,|\, \mathbf{z}_{1:i-1}^{(k)})\, p(\mathbf{y}_i \,|\, \mathbf{z}_{1:i}^{(k)}, \mathbf{y}_{1:i-1})}{q(\mathbf{z}_i^{(k)} \,|\, \mathbf{y}_{1:i}, \mathbf{z}_{1:i-1}^{(k)})} \tag{2.33}$$

2.3.6.1 Relationship with AIS

While SMC is based on a different intuition from AIS, the underlying mathematics is equivalent. In particular, we discuss the unifying view of del Moral et al. (2006). For simplicity, assume there is only a single particle, *i.e.* $K = 1$. While Algorithm 2 incrementally builds up the latent representation one data point at a time, we

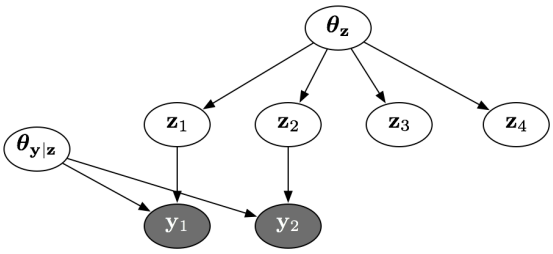


Figure 2-1: The intermediate distribution $f_2(\mathbf{y}, \mathbf{z}) = p(\boldsymbol{\theta})\, p(\mathbf{z}\,|\,\mathbf{y})\, p(\mathbf{y}_{1:2}\,|\,\boldsymbol{\theta}, \mathbf{z})$.

can imagine that all of the latent variables are explicitly represented at every step. Recall that AIS was defined in terms of a sequence of unnormalized distributions f_t and MCMC transition operators $\mathcal{T}_t$ which leave each distribution invariant. In this section, t ranges from 0 to T, rather than 1 to T as in Section 2.3.5.

The intermediate distributions are constructed by including only a subset of the data likelihood terms:

$$f_t(\boldsymbol{\theta}, \mathbf{z}) = p(\boldsymbol{\theta}) \prod_{i=1}^{N} p(\mathbf{z}_i) \prod_{i=1}^{t} p(\mathbf{y}_i\,|\,\boldsymbol{\theta}, \mathbf{z}_i). \tag{2.34}$$

This distribution is shown in Figure 2-1. Since each distribution in the sequence differs from its predecessor simply by adding an additional observation likelihood term,

$$\frac{f_t(\boldsymbol{\theta}, \mathbf{z})}{f_{t-1}(\boldsymbol{\theta}, \mathbf{z})} = p(\mathbf{y}_t\,|\,\boldsymbol{\theta}, \mathbf{z}_t). \tag{2.35}$$

The transition operator first samples $\boldsymbol{\theta}$ from the conditional distribution $p(\boldsymbol{\theta}\,|\,\mathbf{y}_{1:t}, \mathbf{z}_{1:t})$, and then resamples $\mathbf{z}_{t+1:N}$ from $p(\mathbf{z}_{t+1:N}\,|\,\boldsymbol{\theta})$.

At an abstract level, all of the latent variables are represented throughout the algorithm. However, by inspection, the latent variables corresponding to unobserved data points have no effect on any of the other values which are sampled, and therefore they need not be explicitly computed. The algorithm, therefore, is equivalent to Algorithm 2 for the case of one particle.

2.3.7 Variational Bayes

All of the methods described above are sampling-based estimators. Variational Bayes is an alternative set of techniques based on optimization. In particular, the aim is to approximate the intractable posterior distribution $p(\mathbf{z}, \boldsymbol{\theta} \,|\, \mathbf{y})$ with a tractable approximation $q(\mathbf{z}, \boldsymbol{\theta})$, *i.e.* one whose structure is simple enough to represent explicitly. Typically, $\mathbf{z}$ and $\boldsymbol{\theta}$ are assumed to be independent, *i.e.* $q(\mathbf{z}, \boldsymbol{\theta}) = q(\mathbf{z})q(\boldsymbol{\theta})$, and the individual terms may have additional factorization assumptions as well. The objective function being optimized is the following:

$$\mathcal{F}(q) \triangleq \mathbb{E}_{q(\mathbf{z},\boldsymbol{\theta})} \left[\log p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{y})\right] + \mathcal{H}\left[q(\mathbf{z}, \boldsymbol{\theta})\right], \tag{2.36}$$

where $\mathcal{H}$ denotes entropy. This functional is typically optimized using a coordinate ascent procedure, whereby each factor of q is optimized given the other factors. Assuming the factorization given above, the update rules which optimize (2.36) are:

$$q(\mathbf{z}) \propto \exp\left(\mathbb{E}_{q(\boldsymbol{\theta})} \left[\log p(\mathbf{z}, \boldsymbol{\theta}, \mathbf{y})\right]\right) \tag{2.37}$$

$$q(\boldsymbol{\theta}) \propto \exp\left(\mathbb{E}_{q(\mathbf{z})} \left[\log p(\mathbf{z}, \boldsymbol{\theta}, \mathbf{y})\right]\right) \tag{2.38}$$

Variational Bayes is used for both posterior inference and marginal likelihood estimation, and the two tasks are equivalent, according to the following identity:

$$\log \mathcal{F}(q) = \log p(\mathbf{y}) - \mathrm{D}_{\mathrm{KL}}(q(\mathbf{z}, \boldsymbol{\theta}) \,\|\, p(\mathbf{z}, \boldsymbol{\theta} \,|\, \mathbf{y})). \tag{2.39}$$

I.e., variational Bayes underestimates the true log marginal likelihood, and the gap is determined by the KL divergence from the true posterior.

2.3.8 Chib-style estimators

Another estimator which is popular because of its simplicity is Chib's method (Chib, 1995). This method is based on the identity

$$p(\mathbf{y}) = \frac{p(\mathbf{z}^\star, \boldsymbol{\theta}^\star, \mathbf{y})}{p(\mathbf{z}^\star, \boldsymbol{\theta}^\star \,|\, \mathbf{y})} \tag{2.40}$$

for any particular values $(\mathbf{z}^\star, \boldsymbol{\theta}^\star)$ of the latent variables and parameters. While (2.40) holds for any choice of $(\mathbf{z}^\star, \boldsymbol{\theta}^\star)$, they are usually taken to be high probability locations, such as the maximum a posteriori (MAP) estimate. The numerator can generally be computed in closed form. The denominator is based on a Monte Carlo estimate of the conditional probability obtained from posterior samples $(\mathbf{z}^{(1)}, \boldsymbol{\theta}^{(1)}), \ldots, (\mathbf{z}^{(K)}, \boldsymbol{\theta}^{(K)})$.

In particular, let $\mathcal{T}$ represent an MCMC operator which leaves $p(\mathbf{z}, \boldsymbol{\theta} \,|\, \mathbf{y})$ invariant; the basic version of the algorithm assumes a Gibbs sampler. For models where the Gibbs transitions can't be computed exactly, another variant uses Metropolis-Hastings instead (Chib and Jeliazkov, 2001). (The posterior samples may be obtained from a Markov chain using $\mathcal{T}$, but this is not required.) The denominator is estimated as:

$$\hat{p}(\mathbf{z}^\star, \boldsymbol{\theta}^\star \,|\, \mathbf{y}) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{T}(\mathbf{z}^\star, \boldsymbol{\theta}^\star \,|\, \mathbf{z}^{(k)}, \boldsymbol{\theta}^{(k)}, \mathbf{y}). \tag{2.41}$$

How should the estimator be expected to perform? Observe that if exact samples are used, (2.41) is an unbiased estimate of the denominator of (2.40). Therefore, following the analysis of Section 2.3.1, it would tend to underestimate the denominator, and therefore overestimate the true marginal likelihood value. If approximate posterior samples are used, nothing can be said about its relationship with the true value. In this review, we focus on latent variable models, which generally have symmetries corresponding to relabeling of latent components or dimensions. Since transition probabilities between these modes are very small, the estimator could drastically overestimate the marginal likelihood unless the posterior samples happen to include the correct mode. Accounting for the symmetries in the algorithm itself can be tricky, and can cause subtle bugs (Neal, 1999).

Murray and Salakhutdinov (2009) proposed a variant on Chib's method which yields an unbiased estimate of the marginal likelihood. We will refer to the modified version as the Chib-Murray-Salakhutdinov (CMS) estimator. The difference is that, rather than allowing an arbitrary initialization for the Markov chain over $(\mathbf{z}, \boldsymbol{\theta})$, they initialize the chain with a sample from $\tilde{\mathcal{T}}(\mathbf{z}, \boldsymbol{\theta} \,|\, \mathbf{z}^\star, \boldsymbol{\theta}^\star)$, where

$$\tilde{\mathcal{T}}(\mathbf{z}', \boldsymbol{\theta}' \,|\, \mathbf{z}, \boldsymbol{\theta}) \triangleq \frac{\mathcal{T}(\mathbf{z}, \boldsymbol{\theta} \,|\, \mathbf{z}', \boldsymbol{\theta}')\, p(\mathbf{z}', \boldsymbol{\theta}' \,|\, \mathbf{y})}{\sum_{\mathbf{z}', \boldsymbol{\theta}'} \mathcal{T}(\mathbf{z}, \boldsymbol{\theta} \,|\, \mathbf{z}', \boldsymbol{\theta}')\, p(\mathbf{z}', \boldsymbol{\theta}' \,|\, \mathbf{y})} \tag{2.42}$$

is the reverse operator of $\mathcal{T}$.

Chapter 3

A grammar for matrix decompositions

Matrix decompositions are a class of models where an input matrix is represented in terms of sums and products of matrices with simple priors or constraints. This chapter presents a compact notation for matrix decomposition models in terms of algebraic expressions. The models are organized into a context-free grammar, where the productions correspond to simple probabilistic modeling motifs such as clustering or dimensionality reduction. Complex structures are generated through the iterated application of these productions. We discuss several examples of existing models which can be (approximately) represented in this framework.

3.1 Motivation

As a motivating example, consider a matrix where the rows correspond to 50 different kinds of mammals, and the columns correspond to individual features they might have, such as "can swim" or "has teeth" (Osherson et al., 1991). Each entry has a binary label depending on the presence of the attribute. (In this chapter, we follow the machine learning convention where rows of the matrix correspond to data points or entities, and the columns correspond to feature dimensions. However, the model space we present is symmetric with respect to rows and columns.) We may be interested in modeling the underlying structure in order to understand the relationships between different mammals or to make predictions about unknown attributes.

One way to analyze this dataset is to cluster the animals into meaningful groups, such as herbivores or predators. (See Section 2.1.2.) We could model such structure

using a mixture of Bernoullis model, where each mammal is drawn from a mixture of K components with probabilities $\boldsymbol{\pi}$, and each cluster has a vector of probabilities $\mathbf{p}_k$ for each of the features. In other words, the distribution over features $\mathbf{y}$ for a new mammal is given by:

$$\begin{aligned} z &\sim \text{Multinomial}(\boldsymbol{\pi}) \\ y_j &\sim \text{Bernoulli}(p_{zj}). \end{aligned}$$

When talking about matrix decomposition models, it is more convenient to work with real-valued matrices than binary ones, so we model the observation matrix $\mathbf{Y}$ in terms of a real-valued latent matrix $\mathbf{X}$ and an observation model connecting $\mathbf{X}$ to $\mathbf{Y}$. To model binary observations, we can suppose $\mathbf{Y}$ was generated by thresholding $\mathbf{X}$ at zero:

$$\begin{aligned} z &\sim \text{Multinomial}(\boldsymbol{\pi}) \\ x_j &\sim \mathcal{N}(\mu_{zj}, 1) \\ y_j &= \mathbf{1}_{x_j > 0}. \end{aligned}$$

This model allows us to perform certain kinds of inferences. For instance, if we spot a new mammal which is large and has claws and sharp teeth, we may conclude it is a predator, and be able to make some further predictions about its attributes. However, if we see a new kind of mammal which doesn't appear to fit into any of the previously seen categories, we're unable to draw any conclusions at all. A more structured model known as co-clustering, or biclustering, clusters both the rows and the columns of the matrix, allowing us to model the relationships between different dimensions (Kemp et al., 2006). For instance, "has claws" and "has sharp teeth" probably cluster together, so an animal in a previously unknown category which has one attribute is likely to have the other as well.

Both the clustering and co-clustering models can be represented as matrix decompositions. In the clustering model of Figure 3-1 (left), the latent matrix $\mathbf{X}$ is decomposed into a product of a binary matrix representing the cluster assignments and a real-valued matrix representing the cluster centers, and another real-valued matrix is added to the result to capture the within-cluster variation. The co-clustering model of Figure 3-1 (right) is similar, except that the matrix of cluster centers is decomposed further into a real-valued matrix and a binary matrix representing the cluster assignment of each feature dimension. We can think of this as a recursive decomposition: a matrix decomposition model is applied to one of the components of a matrix decomposition model. The rest of this chapter makes this notion more

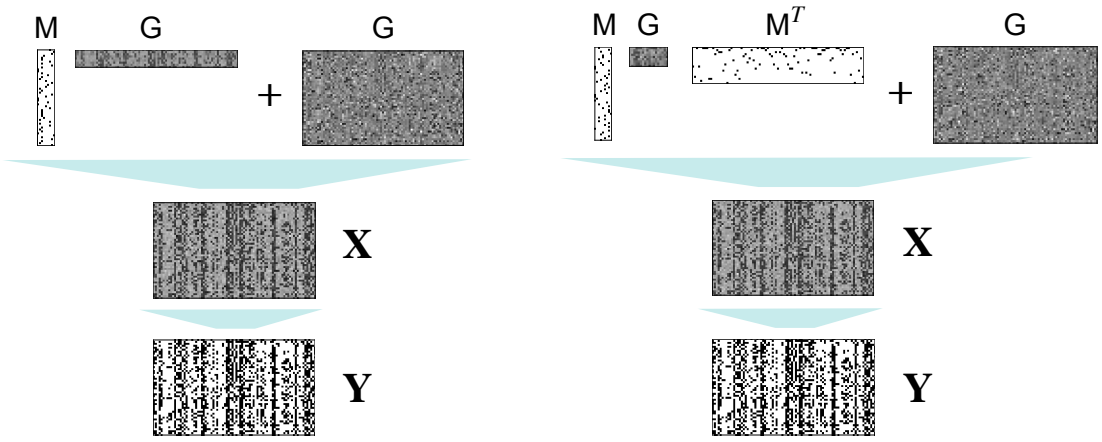


Figure 3-1: **Left:** a flat clustering model fit to the mammals dataset. Rows = animals, columns = attributes. $\mathbf{Y}$ is the observation matrix, and $\mathbf{X}$ are the latent variables postulated to explain the observations. The three matrices at the top denote cluster assignments, cluster centers, and within-cluster variability. M and G are component priors (see Section 3.2). **Right:** a co-clustering model fit to the same data. The four matrices at the top denote the row cluster assignments, block parameters, column cluster assignments, and within-block variability.

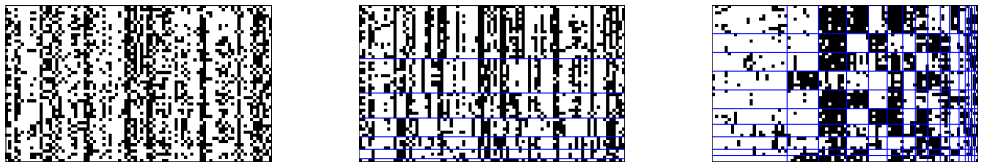


Figure 3-2: Visualizations of the animals data by sorting the rows and columns according to learned structure. **Left:** the raw observation matrix. **Middle:** Rows are sorted by the cluster assignment in a flat clustering model. **Right:** Rows and columns are each sorted by cluster assignment in a co-clustering model.

precise.

In addition to making predictions, probabilistic models can be used to look for patterns in a dataset. Figure 3-2 shows visulizations of the animals dataset corresponding to the clustering and co-clustering models, where the rows and columns of the input matrix are sorted according to the learned structure. The co-clustering model gives a block structure, where entries within a block are likely to share the same value.

One advantage of viewing this model as a matrix decomposition is that it suggests ways to generalize the model. For instance, why should we restrict ourselves to discrete, non-overlapping categories? Perhaps the animals are better described as binary vectors representing membership in overlapping categories. Or perhaps the feature dimensions are more compactly described using low-dimensional real-valued vectors. The rest of this chapter defines an open-ended space of matrix decomposition models, with the goal of making such decisions automatically.

3.2 A notation for matrix decompositions

So far, we have discussed decomposing a matrix into sums and products of simpler matrices, either real-valued or binary-valued. In order to specify a probabilistic model, we need to specify distributions over each of these component matrices. Surprisingly, a handful of distributions suffice to specify a wide range of powerful matrix decomposition models. In this section, we describe the set of component priors and how they can be combined. We denote the matrix as $\mathbf{U}$, with entries u_{ij}. Boldface letters (*e.g.* $\mathbf{A}$) represent matrices, whereas sans-serif letters (*e.g.* G) represent component priors.

1. **Gaussian (G).** In probabilistic modeling, when we want to avoid assuming any particular structure for a set of real-valued variables, we often model them as independent Gaussians. This motivates our first component prior, where the entries are all independent Gaussians:

$$u_{ij} \sim \mathcal{N}(0, \sigma_i \sigma_j).$$

This equation assumes each row and column has an associated scale parameter, denoted σ_i and σ_j, respectively. Whether these parameters are shared between different columns is a modeling choice. For simplicity, in our framework, we assume the variances are shared between all dimensions corresponding to dimensions of the input matrix, but each latent dimension has its own variance parameter. In this way, the variances can be generalized to new rows and columns, while the variances of individual latent dimensions can be estimated.[1] We place an inverse gamma prior on σ_i^2 and σ_j^2 where applicable. This component is represented with the letter G, for "generic" or "Gaussian."

2. **Multinomial (M).** When we fit mixture models, we often assume the assignments are drawn independently from a multinomial distribution, and the parameters of the multinomial are shared between all rows. This can be represented with a Dirichlet-Multinomial distribution:

$$\begin{aligned} \boldsymbol{\pi} &\sim \text{Dirichlet}(\alpha) \\ \mathbf{u}_i &\sim \text{Multinomial}(\boldsymbol{\pi}) \end{aligned}$$

Here $\mathbf{u}_i$ is represented as a binary vector with exactly one 1, which represents the cluster assignment. This component is denoted M, for "mixture" or "multinomial."

3. **Binary (B).** The next component is a distribution over binary matrices where the entries are independent Bernoullis, and the mean parameters are shared between different rows. Specifically, we assume a beta-Bernoulli distribution:

$$\begin{aligned} \pi_j &\sim \text{Beta}(a, b) \\ u_{ij} &\sim \text{Bernoulli}(\pi_j) \end{aligned}$$

[1] A more flexible approach, which we have not implemented, would be to give every row or column its own scale parameter, but learn the prior over scale parameters, so that the scales can be generalized to new rows or columns.

This component is used to build binary attribute models. We denote it B, for "binary" or "Bernoulli."

4. **Integration matrix (C).** Finally, we include an integration matrix, which is a deterministic matrix with ones below the diagonal:

$$u_{ij} = \mathbf{1}_{i \geq j}.$$

This is useful for modeling temporal structure, as multiplying by this matrix has the effect of cumulatively summing the rows. This component is denoted C, for "chain" or "cumulative."

In our experiments, the parameters α, a, and b are all fixed to reasonable defaults.

We write matrix decomposition models using algebraic expressions containing these four symbols. The operations we allow are addition, matrix multiplication, matrix transpose, elementwise product (denoted $\circ$), and elementwise exponentiation (denoted $\exp(\cdot)$). The generative model is simple: generate each of the matrices from its corresponding prior, and evaluate the expression. For instance, the clustering model of Figure 3-1 (left) would be denoted $\mathsf{MG} + \mathsf{G}$, and the co-clustering model of Figure 3-1 (right) would be denoted $\mathsf{MGM}^T + \mathsf{G}$. Note that each occurrence of a given letter in the formula represents a *different* matrix drawn from that prior.

So far, we have not specified the dimensions of the component matrices. How do we know, for instance, that the matrices in $\mathsf{MG} + \mathsf{G}$ represent a tall matrix, a fat matrix, and a matrix the same size as the observation matrix $\mathbf{Y}$? Some of the dimensions are determined by the dimensions of $\mathbf{Y}$. The remaining dimensions need to be fit to the data; we defer discussion of this to Section 4.1, which discusses the procedure for inferring the latent component matrices. For now, we simply note that an algebraic expression, combined with the sizes of all component matrices, defines a generative model over $\mathbf{Y}$.

3.3 Grammar

In the previous section, we introduced a notation for matrix decomposition models. However, fitting models corresponding to arbitrary expressions is difficult. Furthermore, the set of expressions of a given size grows exponentially, precluding exhaustive search. In order to perform posterior inference and structure search, we must further restrict the space of models.

Observe that our notation for matrix decompositions is recursive: expressions can be broken down into subexpressions, each of which is itself a matrix decompo-

sition model. Context-free grammars are a useful formalism for capturing this sort of recursive structure. We define a context free grammar whose starting symbol is G, the structureless model where the entries are assumed to be independent Gaussians. Other models (expressions) are generated by iteratively applying the following production rules:

$$\begin{aligned}
&\text{low-rank approximation} && \mathsf{G} \to \mathsf{GG} + \mathsf{G} && (1)\\
&\text{clustering} && \mathsf{G} \to \mathsf{MG} + \mathsf{G} \mid \mathsf{GM}^T + \mathsf{G} && (2)\\
& && \mathsf{M} \to \mathsf{MG} + \mathsf{G} && (3)\\
&\text{binary attributes} && \mathsf{G} \to \mathsf{BG} + \mathsf{G} \mid \mathsf{GB}^T + \mathsf{G} && (4)\\
& && \mathsf{B} \to \mathsf{BG} + \mathsf{G} && (5)\\
& && \mathsf{M} \to \mathsf{B} && (6)\\
&\text{linear dynamics} && \mathsf{G} \to \mathsf{CG} + \mathsf{G} \mid \mathsf{GC}^T + \mathsf{G} && (7)\\
&\text{sparsity} && \mathsf{G} \to \exp(\mathsf{G}) \circ \mathsf{G} && (8)
\end{aligned}$$

For instance, any occurrence of G in a model may be replaced by $\mathsf{GG} + \mathsf{G}$ or $\mathsf{MG} + \mathsf{G}$. Iterated application of these productions allows us to build hierarchical models by specifying additional dependencies between variables which were previously modeled as independent.

Each of the productions corresponds to a motif of probabilistic modeling. With the exception of the sparsity rule, they can each be viewed as stand-alone models. We describe all of the production rules as factorizations of an $N \times D$ matrix $\mathbf{X}$. The production rules are as follows:

- **Low-rank approximation.** The rule $\mathsf{G} \to \mathsf{GG} + \mathsf{G}$ takes a Gaussian matrix $\mathbf{X}$ and approximates it as the product $\mathbf{UV}$, where $\mathbf{U}$ and $\mathbf{V}$ are two Gaussian matrices of sizes $N \times K$ and $K \times D$. Typically, $K < \min(N, D)$, so the product has rank K. This factorization corresponds to a linear dimensionality reduction model, such as PCA, factor analysis, or probabilistic matrix factorization.[2] Each of the latent dimensions has an associated variance parameter, with an inverse Gamma prior. Therefore, as part of fitting the model, we can infer the

[2]The precise model depends on the variance model; see Section 3.2. If the variances are shared between input rows and columns but not latent dimensions, it corresponds to probabilistic PCA (Tipping and Bishop, 1999). If each feature dimension also has its own variance parameter, it corresponds to factor analysis (Basilevsky, 1994). If variances are shared between all input dimensions and all latent dimensions, it corresponds to probabilistic matrix factorization (Salakhutdinov and Mnih, 2008).

effective dimensionality by learning small variances for the unused dimensions. Low rank approximations are described in Section 2.1.1 of the background.

- **Clustering.** The rule $\mathsf{G} \to \mathsf{MG} + \mathsf{G}$ replaces a Gaussian matrix with a mixture of Gaussians. This corresponds to an approximation $\mathbf{X} \approx \mathbf{ZA}$, where $\mathbf{Z}$ represents the cluster assignments and $\mathbf{A}$ represents the cluster centers, as described in Section 3.1. The mirror image $\mathsf{G} \to \mathsf{GM}^T + \mathsf{G}$ is the same, except that columns are clustered rather than rows. Clustering models are discussed further in Section 2.1.2. The rule $\mathsf{M} \to \mathsf{MG} + \mathsf{G}$ replaces a Multinomial matrix with a mixture of Gaussians. The intuition is that if we've previously modeled a dataset in terms of a hard clustering, we may decide that a soft clustering is more appropriate. This production rule adds that extra flexibility.

- **Binary attributes.** The next production rule, $\mathsf{G} \to \mathsf{BG} + \mathsf{G}$, replaces a Gaussian matrix with a binary attribute model $\mathbf{X} \approx \mathbf{ZA}$, where $\mathbf{Z}$ is a binary matrix and $\mathbf{A}$ is a real-valued matrix. Each row of $\mathbf{Z}$ can be thought of as the binary vector representation of one data point, and the kth row of $\mathbf{A}$ can be thought of as a feature vector which is added to the data vector if the kth attribute is present. The rule $\mathsf{G} \to \mathsf{GB}^T + \mathsf{G}$ is the same, except that it fits a binary vector representation of columns rather than rows. The rules $\mathsf{B} \to \mathsf{BG} + \mathsf{G}$ and $\mathsf{M} \to \mathsf{B}$ add more flexibility to a model. Binary attribute models are discussed in Section 2.1.3 of the background chapter.

- **Linear dynamics.** So far, all of the production rules have been exchangeable: they are invariant to the ordering of rows and columns. If we wish to capture temporal continuity in the data, where the rows correspond to time steps, we can apply the rule $\mathsf{G} \to \mathsf{CG} + \mathsf{G}$, which models the columns as independent Markov chains. Recall that multiplying by the lower triangular matrix C has the effect of cumulatively summing the rows. In this representation, the first G matrix gives the *innovations*, or the differences in values between consecutive time steps. Linear dynamical systems are discussed in Section 2.1.4 of the background.

- **Sparsity.** Sometimes we would like real-valued matrices with heavy-tailed distributions. This can be achieved with the production rule $\mathsf{G} \to \exp(\mathsf{G}) \circ \mathsf{G}$. As discussed in Section 2.1.5 of the background, symmetric heavy-tailed distributions centered at 0 can be represented as Gaussian scale mixtures, where a Gaussian variable r is multiplied by a scale variable z. Here, $\exp(\mathsf{G})$ corresponds to the scale variables, and the second G corresponds to the Gaussian variables. Since these are multiplied elementwise, this construction corresponds

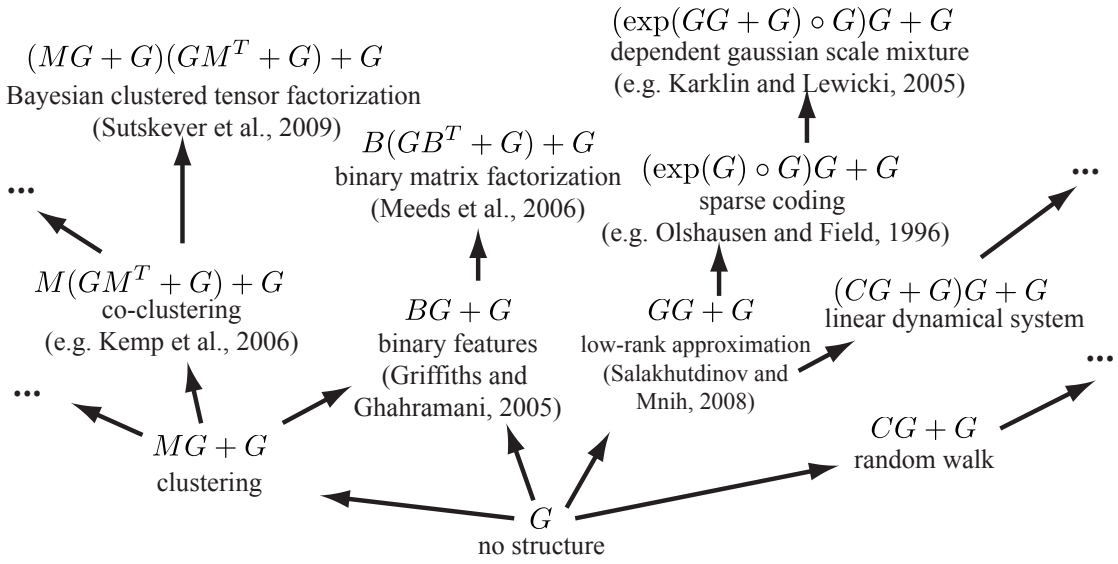


Figure 3-3: Examples of existing machine learning models which fall under our framework. Arrows represent models reachable using a single production rule. Only a small fraction of the 2496 models reachable within 3 steps are shown, and not all possible arrows are shown.

to independent draws from a heavy-tailed distribution. This is often referred to as a sparse distribution, since the values are unlikely to be far from 0. The degree of sparsity can be controlled by setting the variance of the scale variables; if the variance is small, the distribution will be nearly Gaussian, whereas if the variance is large, it will be very heavy-tailed.

3.4 Examples

We now turn to several examples in which our simple components and production rules give rise to a rich variety of models from unsupervised learning. While the model space is symmetric with respect to rows and columns, we adopt the convention that the rows of $\mathbf{Y}$ correspond to data points and columns corresponds to observed features. For simplicity, we assume that $\mathbf{Y} = \mathbf{X}$, *i.e.* that the matrix decomposition model directly generates the observed data. (For binary data, one could instead assume $\mathbf{Y}$ results from applying a threshold to $\mathbf{X}$, as in Section 3.1.) Figure 3-3 gives several additional examples of matrix decomposition models and highlights the relationships between them.

We note that we omit certain design choices which were significant in the original presentation of some of these models. Our aim is not to reproduce the existing models exactly, but to capture their overall structure in a compact framework which highlights the relationships between the models. One particular way in which our grammar differs from previous work is that many models we discuss were originally formulated as Bayesian nonparametric models, whereas we limit ourselves to finite models. (However, as discussed in Section 4.1.1, we use Bayesian nonparametric techniques as a heuristic to infer the latent dimensions.)

3.4.1 Clustering

We always begin with the structureless model G, which assumes the entries of the matrix are *i.i.d.* Gaussian. By applying the rule $\mathsf{G} \to \mathsf{MG}+\mathsf{G}$, we obtain the clustering model $\mathsf{MG}+\mathsf{G}$ discussed in Section 3.1, which groups the rows (entities) into clusters.

Our grammar does not license the co-clustering model $\mathsf{MGM}+\mathsf{G}$ of Section 3.1. However, by applying the rule $\mathsf{G} \to \mathsf{GM}^T + \mathsf{G}$ to the matrix of cluster centers, we obtain another model which captures the same structure, $\mathsf{M}(\mathsf{GM}^T + \mathsf{G}) + \mathsf{G}$.[3] Specifically, the model clusters both the rows and the columns of $\mathbf{X}$. The difference is that $\mathsf{MGM}+\mathsf{G}$ requires a hard clustering of the columns, while $\mathsf{M}(\mathsf{GM}^T + \mathsf{G}) + \mathsf{G}$ allows a soft clustering. If a hard clustering is in fact a better match to the data, this can be learned by fitting a small variance parameter for the inner factorization. The decision to use a soft clustering will be further explained in Section 4.1, where we discuss algorithms for inference.

There may be additional structure not captured by the co-clustering model. For instance, there may be correlations within the blocks, or the mean parameters of the blocks may themselves have structure. Bayesian clustered tensor factorization (BCTF) (Sutskever et al., 2009) is a model intended to capture these correlations. While the original model was proposed for general tensors, we focus on the special case of matrices. BCTF assumes the matrix has a low-rank representation, and the low-rank factors each have a mixture of Gaussians distribution. In our grammar, we obtain this model by applying the rule $\mathsf{M} \to \mathsf{MG}+\mathsf{G}$ to the first M matrix in the co-clustering model to get $(\mathsf{MG}+\mathsf{G})(\mathsf{GM}^T + \mathsf{G}) + \mathsf{G}$. This rule replaces the hard clustering with a soft clustering.

Graphical model representations are shown in Figure 3-4. The graphical models leave implicit the forms of the distributions, which are one of the most important

[3] We note that in our grammar, a co-clustering model can alternatively be obtained starting from a clustering of columns: first apply $\mathsf{G} \to \mathsf{GM}^T + \mathsf{G}$, followed by $\mathsf{G} \to \mathsf{MG}+\mathsf{G}$ to get $(\mathsf{MG}+\mathsf{G})\mathsf{M}^T + \mathsf{G}$. We will not distinguish these models even though they are not strictly identical.

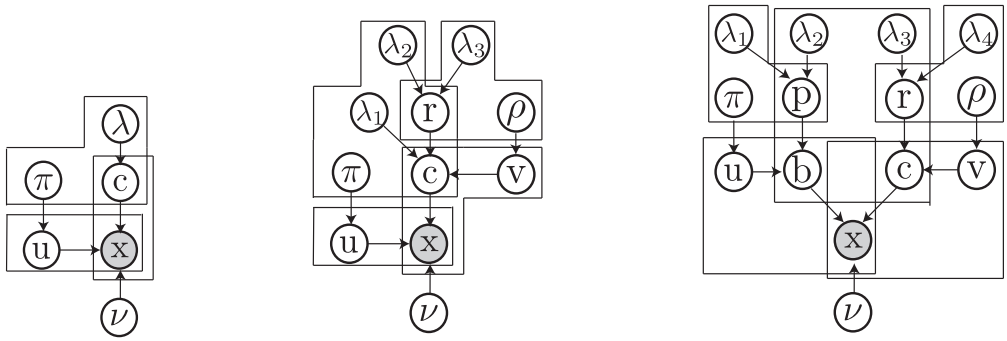


Figure 3-4: Representations of some models from our grammar as graphical models. From left to right: the flat clustering model $\mathsf{MG}+\mathsf{G}$, the co-clustering model $\mathsf{M}(\mathsf{GM}^T+\mathsf{G})+\mathsf{G}$, and Bayesian clustered tensor factorization, $(\mathsf{MG}+\mathsf{G})(\mathsf{GM}^T+\mathsf{G})+\mathsf{G}$. The graphical models show which variables directly influence each other, but obscure much of the relevant structure.

pieces of information about a given model. Our algebraic expression notation, while it leaves out some details of the original models, is compact and highlights the relevant structure.

The data points or attributes may be better represented in terms of overlapping clusters. This can be represented using binary attribute models, where M components are replaced with B components in the above models. In particular, by starting with the production rule $\mathsf{G} \to \mathsf{BG} + \mathsf{G}$, we obtain a finite version of the IBP linear-Gaussian model (Griffiths and Ghahramani, 2005), where data points are represented using binary vectors. By applying $\mathsf{G} \to \mathsf{GB}^T + \mathsf{G}$ to the feature matrix, we obtain $\mathsf{B}(\mathsf{GB}^T + \mathsf{G}) + \mathsf{G}$, where both the rows and the columns have binary representations. This is similar to the binary matrix factorization (BMF) model (Meeds et al., 2006), which can be represented as $\mathsf{BGB}^T + \mathsf{G}$. While it might not be obvious from the original formulations, the matrix decomposition representation highlights the fact that BMF is essentially the binary analogue of co-clustering.

This framework immediately suggests ways to extend these models. For instance, $\mathsf{B}(\mathsf{GM}^T + \mathsf{G}) + \mathsf{G}$ represents rows as binary vectors and columns as discrete clusters. Alternatively, $(\mathsf{BG} + \mathsf{G})(\mathsf{GB}^T + \mathsf{G}) + \mathsf{G}$ would be the analogue of BCTF which uses binary vectors rather than clusters.

3.4.2 Linear dynamics

As discussed in Section 3.3, the production rule $\mathsf{G} \to \mathsf{CG} + \mathsf{G}$ gives a model where the columns of $\mathbf{X}$ are independent Markov chains. Features are often correlated, however, and this can be captured with a linear dynamical system. Specifically, by starting with the production rule $\mathsf{G} \to \mathsf{GG} + \mathsf{G}$ and applying $\mathsf{G} \to \mathsf{CG} + \mathsf{G}$, we arrive at the model $(\mathsf{CG} + \mathsf{G})\mathsf{G} + \mathsf{G}$, where the data are explained in terms of a low-dimensional representation which evolves over time. The model can be written in the more familiar form:

$$\begin{aligned} \mathbf{z}_t &= \mathbf{z}_{t-1} + \boldsymbol{\epsilon}_t \\ \mathbf{y}_t &= \mathbf{C}_t \mathbf{z}_t + \boldsymbol{\delta}_t. \end{aligned}$$

See Section 2.1.4 of the background for more information on linear dynamical systems.

3.4.3 Statistics of natural images

For an example from vision, suppose each row of $\mathbf{Y}$ corresponds to a small (*e.g.* 12×12) patch sampled from an image and vectorized. Image patches can be viewed as lying near a low-dimensional subspace spanned by the lowest frequency Fourier coefficients (Bossomaier and Snyder, 1986). This can be captured by the low-rank model $\mathsf{GG} + \mathsf{G}$, which is analogous to probabilistic PCA (Tipping and Bishop, 1999). (See Section 2.1.1 of the background for more details.)

In a landmark paper, Olshausen and Field (1996) found that image patches are better modeled as a linear combination of a small number of components drawn from a larger dictionary. In other words, $\mathbf{Y}$ is approximated as the product $\mathbf{WA}$, where each row of $\mathbf{A}$ is a basis function, and $\mathbf{W}$ is a sparse matrix giving the linear reconstruction coefficients for each patch. By fitting this "sparse coding" model, they obtained a dictionary of oriented edges similar to the receptive fields of neurons in the primary visual cortex. (See Section 2.1.5 for more details.) If we apply the rule $\mathsf{G} \to \exp(\mathsf{G}) \circ \mathsf{G}$, we obtain a Bayesian version of sparse coding, $(\exp(\mathsf{G}) \circ \mathsf{G})\mathsf{G} + \mathsf{G}$. This is similar to the model proposed by Berkes et al. (2008). Intuitively, the latent Gaussian coefficients are multiplied elementwise by "scale" variables to give a heavy-tailed distribution.

Many researchers have designed models to capture the dependencies between these scale variables, and such "Gaussian scale mixture" models represent the state-of-the art for low-level vision tasks such as denoising (Portilla et al., 2003) and texture synthesis (Portilla and Simoncelli, 2000). (See Section 2.1.6 for more details.) One

such GSM model is that of Karklin and Lewicki (2008), who fit a low-rank model to the scale variables. By applying Rule (1) to the sparse coding structure, we can represent their model in our framework as $(\exp(\mathsf{GG} + \mathsf{G}) \circ \mathsf{G})\mathsf{G} + \mathsf{G}$. This model has been successful at capturing higher-level textural properties of a scene and has properties similar to complex cells in the primary visual cortex.

3.4.4 Deep learning

The previous sections have all discussed models which were implemented as part of the system described in the following chapter. In this section, we consider how the grammar could be extended with a handful of additional productions in order to capture models from deep learning. These models also help motivate the inference procedure described in Section 4.1.

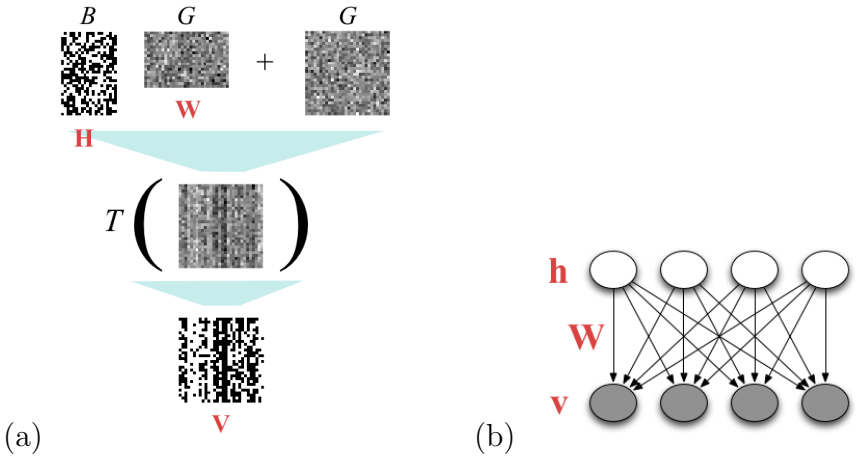
The specific grammar presented in Section 3.3 is not quite powerful enough to represent deep learning models. With the exception of the GSM production rule, all of the rules in the grammar involve only addition, matrix multiplication, and matrix transpose, so each component contributes linearly to the final reconstruction. In order to capture nonlinear structure, we can add a threshold operator T:

$$[\mathrm{T}(\mathbf{A})]_{ij} = \begin{cases} 1 & \text{if } \mathbf{A}_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Suppose we add production rules to our grammar which use the threshold operator to expand binary matrices:

$$\begin{aligned} \mathsf{B} &\to \mathrm{T}(\mathsf{GG} + \mathsf{G}) \\ \mathsf{B} &\to \mathrm{T}(\mathsf{MG} + \mathsf{G}) \\ \mathsf{B} &\to \mathrm{T}(\mathsf{BG} + \mathsf{G}) \end{aligned}$$

Suppose we're given a binary input matrix; for instance, each row might represent an image of a handwritten digit. The structureless model can be represented as a Bernoulli matrix, B. Applying the binary features production rule, we get the model $\mathrm{T}(\mathsf{BG} + \mathsf{G})$. Call the left factor $\mathbf{H}$, the right factor $\mathbf{W}$, and the input matrix $\mathbf{V}$. To see how this relates to belief networks, hold $\mathbf{W}$ fixed and consider the predictive distribution over subsequent rows, $p(\mathbf{v} \,|\, \mathbf{W})$. This corresponds to a sigmoid belief network with a single hidden layer, as shown in Figure 3-5. $\mathbf{H}$ can be seen as a binary representation, and $\mathbf{W}$ can be seen as a weight matrix. The probability of



(a) (b)

Figure 3-5: A sigmoid belief network with one hidden layer, represented as $\mathsf{T}(\mathsf{BG}+\mathsf{G})$. (a) The factorization of the input matrix. (b) The predictive distribution over rows, represented as a graphical model.

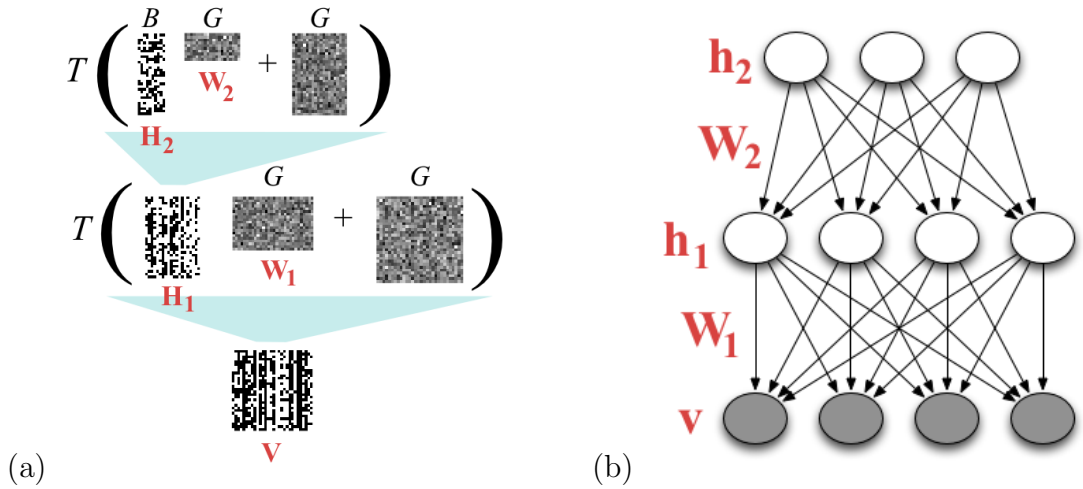
the visible units given the hidden units is given by:

$$p(\mathbf{v}\,|\,\mathbf{h},\mathbf{W}) = \Phi(\mathbf{W}^T\mathbf{h}),$$

where Φ is the probit function. Hence, the generative model for digits is as follows: the activations of $\mathbf{h}$ are sampled as independent Bernoullis, they send messages to the visible units $\mathbf{v}$ using the connection weights $\mathbf{W}$, and each v_i activates independently with a probability that's sigmoidal in its inputs.

We can add another hidden layer by applying the production rule $\mathsf{B} \to \mathsf{T}(\mathsf{BG}+\mathsf{G})$, which yields the model $\mathsf{T}(\mathsf{T}(\mathsf{BG}+\mathsf{G})\mathsf{G}+\mathsf{G})$, shown in Figure 3-6. This is a deep sigmoid belief network: it is similar to the shallow network, except there is an additional hidden layer to capture the dependencies between the hidden units. Clearly, this process can be iterated indefinitely until there's no more exploitable structure between the top layer's hidden units. Interestingly, the greedy initialization procedure presented in the following chapter parallels Hinton et al. (2006)'s greedy training procedure for deep belief nets.

There is no reason to limit ourselves to adding layers of binary features. It might turn out, for instance, that a good model of the MNIST handwritten digits dataset (LeCun et al., 1998) is to have two layers of binary features, followed by a clustering model. This would correspond to the structure $\mathsf{T}(\mathsf{T}(\mathsf{T}(\mathsf{MG}+\mathsf{G})\mathsf{G}+\mathsf{G})\mathsf{G}+\mathsf{G})$.



(a) (b)

Figure 3-6: A deep sigmoid belief network with one hidden layer, represented as T(T(BG+G)G + G). (a) The factorization of the input matrix. (b) The predictive distribution over rows, represented as a graphical model.

3.5 Discussion

We note that many of the above models are not typically viewed as matrix decomposition structures. Describing them as such results in a compact notation and makes clearer the relationships between the different models. The above examples have in common that complex models can be derived by incrementally adding structure to a sequence of simpler models (in a way that parallels the path researchers took to discover them). This observation motivates our proposed procedures for inference and structure learning, which are discussed in the following chapter.

Chapter 4

Compositional structure search

Chapter 3 introduced a grammar for matrix decomposition models where the productions correspond to simple probabilistic models. In this chapter, we describe algorithms for inference and model selection in this space of models inspired by the recursive structure discovery work outlined in Section 1.3. Inference is performed in composite decomposition models using efficient samplers specialized to each of the productions. A best-first search over the grammar was empirically successful at determining the correct structures for synthetic data and plausible structures for a variety of real-world datasets, all using exactly the same code and with no hand-tuned parameters.

4.1 Posterior inference of component matrices

Searching over matrix decomposition structures requires a generic and unified approach to posterior sampling of the latent matrices. Unfortunately, for most of the structures we consider, the posterior is complicated and multimodal, and escaping from local modes requires carefully chosen special-purpose sampling operators. Engineering such operators for thousands of different models would be undesirable.

Fortunately, the compositional nature of our model space allows us to focus the engineering effort on the relatively small number of production rules. In particular, observe that in a realization of the generative process, the value of an expression depends only on the values of its sub-expressions. This suggests the following initialization procedure: when applying a production rule $\mathcal{P}$ to a matrix $\mathbf{S}$, sample from the posterior for $\mathcal{P}$'s generative model conditioned on it evaluating (exactly) to $\mathbf{S}$. Many of our production rules correspond to simple machine learning models for which researchers have already expended much time developing efficient inference

algorithms:

1. **Low rank.** To apply the rule $\mathsf{G} \to \mathsf{GG} + \mathsf{G}$, we fit the probabilistic matrix factorization (Salakhutdinov and Mnih, 2008) model using block Gibbs sampling over the two factors (see Section 2.2.1). While PMF assumes a fixed latent dimension, we choose the dimension automatically by placing a Poisson prior on the dimension and moving between states of differing dimension using reversible jump MCMC (Green, 1995).

2. **Clustering.** To apply the clustering rule to rows: $\mathsf{G} \to \mathsf{MG} + \mathsf{G}$, or to columns: $\mathsf{G} \to \mathsf{GM}^T + \mathsf{G}$, we perform collapsed Gibbs sampling over the cluster assignments in a Dirichlet process mixture model (see Section 2.2.2).

3. **Binary factors.** To apply the rule $\mathsf{G} \to \mathsf{BG} + \mathsf{G}$ or $\mathsf{G} \to \mathsf{GB}^T + \mathsf{G}$, we perform accelerated collapsed Gibbs sampling (Doshi-Velez and Ghahramani, 2009) over the binary variables in a linear-Gaussian Indian Buffet Process (Griffiths and Ghahramani, 2005) model, using split-merge proposals (Meeds et al., 2006) to escape local modes. (Also see Section 2.2.2.)

4. **Markov chains.** The rule $\mathsf{G} \to \mathsf{CG} + \mathsf{G}$ is equivalent to estimating the state of a random walk given noisy observations, which is done using Rauch-Tung-Striebel (RTS) smoothing.

The remaining production rules begin with a random decomposition of $\mathbf{S}$. The initialization step is followed by generic Gibbs sampling over the entire model. We note that our initialization procedure generalizes "tricks of the trade" whereby complex models are initialized from simpler ones (Kemp et al., 2006; Miller et al., 2009).

In addition to simplifying the engineering, this procedure allows us to reuse computations between different structures. Most of the computation time is in the initialization steps. Each of these steps only needs to be run once on the full matrix, specifically when the first production rule is applied. Subsequent initialization steps are performed on the component matrices, which are considerably smaller. This allows a large number of high level structures to be fit for a fraction of the cost of fitting them from scratch.

4.1.1 Determining the latent dimensions

While some of the above algorithms involve fitting Bayesian nonparametric models, once the dimensionality is chosen, the model is converted to a finite model of fixed dimensionality (as defined in Section 3.3). This does not necessarily yield a correct

or optimal number of components: Miller and Harrison (2013) showed that the Dirichlet process tends to overestimate the number of clusters underlying a dataset generated from a finite clustering model. There is also a bias in the opposite direction: when the model is misspecified, marginalizing out the model parameters tends to overpenalize model complexity (Kass and Raftery, 1995), causing a bias towards too few components. In the context of compositional structure search, the lower level models are overly simplified *by design*, so misspecification effects are likely to be significant. We emphasize that converting a Bayesian nonparametric model to a finite model is merely a heuristic, and should not be relied upon to determine the "true" number of components.

4.2 Scoring candidate structures

Performing model selection requires a criterion for scoring individual structures which is informative yet tractable. To motivate our method, we first discuss two popular choices: marginal likelihood of the input matrix and entrywise mean squared error (MSE). Marginal likelihood (Section 1.2) is widely used in Bayesian model comparison. Unfortunately, this requires integrating out all of the latent component matrices, whose posterior distribution is highly complex and multimodal. While elegant solutions exist for particular models, generic marginal likelihood estimation remains extremely difficult. (Chapters 6 and 7 discuss ongoing work towards computing marginal likelihoods of matrix decomposition models.) At the other extreme, one can hold out a subset of the entries of the matrix and compute the mean squared error for predicting these entries. MSE is easier to implement, but we found that it was unable to distinguish many of the the more complex structures in our grammar.

As a middle ground between these two approaches, we chose to evaluate predictive likelihood of held-out rows and columns. That is, for each row (or column) $\mathbf{y}$ of the matrix, we evaluate $p(\mathbf{y} \,|\, \mathbf{Y}_{\text{obs}})$, where $\mathbf{Y}_{\text{obs}}$ denotes an "observed" sub-matrix. Like marginal likelihood, this tests the model's ability to predict entire rows or columns. However, it can be efficiently approximated in our class of models using a small but carefully chosen toolbox corresponding to the component matrix priors in our grammar. We discuss the case of held-out rows; columns are handled analogously.

First, by expanding out the products in the expression, we can write the decomposition uniquely in the form

$$\mathbf{Y} = \mathbf{U}_1\mathbf{V}_1 + \cdots + \mathbf{U}_n\mathbf{V}_n + \mathbf{E}, \tag{4.1}$$

where $\mathbf{E}$ is an *i.i.d.* Gaussian "noise" matrix and the $\mathbf{U}_i$'s are any of the following:

(1) a component matrix G, M, or B, (2) some number of Cs followed by G, (3) a Gaussian scale mixture. The held-out row $\mathbf{y}$ can therefore be represented as:

$$\mathbf{y} = \mathbf{V}_1^T \mathbf{u}_1 + \cdots + \mathbf{V}_n^T \mathbf{u}_n + \mathbf{e}. \tag{4.2}$$

The predictive likelihood is given by:

$$p(\mathbf{y} \,|\, \mathbf{Y}_{\text{obs}}) = \int p(\mathbf{U}_{\text{obs}}, \mathbf{V} \,|\, \mathbf{Y}_{\text{obs}}) p(\mathbf{u} \,|\, \mathbf{U}_{\text{obs}}) p(\mathbf{y} \,|\, \mathbf{u}, \mathbf{V}) \, \mathrm{d}\mathbf{U}_{\text{obs}} \, \mathrm{d}\mathbf{U} \, \mathrm{d}\mathbf{V} \tag{4.3}$$

where $\mathbf{U}_{\text{obs}}$ is shorthand for $(\mathbf{U}_{\text{obs}\,1}, \ldots, \mathbf{U}_{\text{obs}\,n})$ and $\mathbf{u}$ is shorthand for $(\mathbf{u}_1, \ldots, \mathbf{u}_n)$.

In order to evaluate this integral, we generate samples from the posterior $p(\mathbf{U}_{\text{obs}}, \mathbf{V} \,|\, \mathbf{Y})$ using the techniques described in Section 4.1, and compute the sample average of

$$p_{\text{pred}}(\mathbf{y}) \triangleq \int p(\mathbf{u} \,|\, \mathbf{U}_{\text{obs}}) p(\mathbf{y} \,|\, \mathbf{u}, \mathbf{V}) \, \mathrm{d}\mathbf{u} \tag{4.4}$$

If the term $\mathbf{U}_i$ is a Markov chain, the predictive distribution $p(\mathbf{u}_i \,|\, \mathbf{U}_{\text{obs}})$ can be computed using Rauch-Tung-Striebel smoothing; in the other cases, $\mathbf{u}$ and $\mathbf{U}_{\text{obs}}$ are related only through the hyperparameters of the component prior. Either way, each term $p(\mathbf{u}_i \,|\, \mathbf{U}_{\text{obs}})$ can be summarized as a Gaussian, multinomial, Bernoulli, or Gaussian scale mixture distribution.

It remains to marginalize out the latent representation $\mathbf{u}$ of the held-out row. While this can be done exactly in some simple models, it is intractable in general (for instance, if $\mathbf{u}$ is Bernoulli or a Gaussian scale mixture). It is important that the approximation to the integral be a lower bound, because otherwise an overly optimistic model could be chosen even when it is completely inappropriate.

Our approach is a hybrid of variational and sampling techniques. We first lower bound the integral (4.4) in an approximate model $\tilde{p}_{\text{pred}}$ where the Gaussian scale mixture components are approximated as Gaussians. This is done using using the variational Bayes bound

$$\log \tilde{p}_{\text{pred}}(\mathbf{y}) \geq \mathbb{E}_q[\log \tilde{p}_{\text{pred}}(\mathbf{y}, \mathbf{u})] + \mathcal{H}(q).$$

The approximating distribution $q(\mathbf{u})$ is such that all of the discrete components are independent, while the Gaussian components are marginalized out. The ratio $p_{\text{pred}}(\mathbf{y})/\tilde{p}_{\text{pred}}(\mathbf{y})$ is then estimated using annealed importance sampling (AIS) (Neal, 2001a). Because AIS is an unbiased estimator which always takes positive values, by Markov's inequality we can regard it as a stochastic lower bound. Therefore, this small toolbox of techniques allows us to (stochastically) lower bound the predictive

likelihood across a wide variety of matrix decomposition models.

4.3 Search over structures

We aim to find a matrix decomposition structure which is a good match to a dataset, as measured by the predictive likelihood criterion of Section 4.2. Since the space of models is large and inference in many of the models is expensive, we wish to avoid exhaustively evaluating every model. Instead, we adopt a greedy search procedure inspired by the process of scientific discovery and by the recursive structure discovery approaches outlined in Section 1.3. In particular, consider a common heuristic researchers use to build probabilistic models: we begin with a model which has already been applied to a problem, look for additional dependencies not captured by the model, and refine the model to account for those dependencies.

In our approach, refining a model corresponds to applying one of the productions. This suggests the following greedy search procedure, which iteratively "expands" the best-scoring unexpanded models by applying all possible production rules and scoring the resulting models. In particular we first expand the structureless model G. Then, in each step, we expand the K best-performing models from the previous step by applying all possible productions. We then score all the resulting models. The procedure stops when no model achieves sufficient improvement over the best model from the previous step. We refer to the models reached in i productions as the Level i models; for instance, $\mathsf{GG} + \mathsf{G}$ is a Level 1 model and $(\mathsf{MG} + \mathsf{G})\mathsf{G} + \mathsf{G}$ is a Level 2 model.

The effectiveness of this search procedure depends whether the score of a simple structure is a strong indicator of the best score which can be obtained from the structures derived from it. In our experiments, the scores of the simpler structures turned out to be a powerful heuristic: while our experiments used $K = 3$, in most cases, the correct (or best-scoring) structure would have been found with a purely greedy search ($K = 1$). This results in enormous savings because of the compositional nature of our search space: while the number of possible structures (up to a given level) grows quickly in the number of production rules, the number of structures evaluated by this search procedure is merely linear.

The search procedure returns a high-scoring structure for each level in our grammar. There remains a question of when to stop. Choosing between structures of differing complexity imposes a tradeoff between goodness of fit and other factors such as interpretability and tractability of inference, and inevitably the choice is somewhat subjective. In practice, a user may wish to run our procedure up to a fixed level and analyze the sequence of models chosen, as well as the predictive likeli-

hood improvement at each level. However, for the purposes of evaluating our system, we need it to return a single answer. In all of our experiments, we adopt the following arbitrary but consistent criterion: prefer the higher level structure if its predictive log-likelihood score improves on the previous level by at least one nat per row and column.[1]

4.4 Experiments

4.4.1 Synthetic data

We first validated our structure learning procedure on synthetic data where the correct model was known. We generated matrices of size 200×200 from all of the models in Figure 3-3, with 10 latent dimensions. The noise variance σ^2 was varied from 0.1 to 10, while the signal variance was fixed at 1.[2] The structures selected by our procedure are shown in Table 4.1.

	— Increasing noise ⟶			
	$\sigma^2 = 0.1$	$\sigma^2 = 1$	$\sigma^2 = 3$	$\sigma^2 = 10$
low-rank	$\mathsf{GG} + \mathsf{G}$	$\mathsf{GG} + \mathsf{G}$	$\mathsf{GG} + \mathsf{G}$	①G
clustering	$\mathsf{MG} + \mathsf{G}$	$\mathsf{MG} + \mathsf{G}$	$\mathsf{MG} + \mathsf{G}$	$\mathsf{MG} + \mathsf{G}$
binary latent features	①$(\mathsf{BG} + \mathsf{G})\mathsf{G} + \mathsf{G}$	$\mathsf{BG} + \mathsf{G}$	$\mathsf{BG} + \mathsf{G}$	$\mathsf{BG} + \mathsf{G}$
co-clustering	$\mathsf{M}(\mathsf{GM}^T + \mathsf{G}) + \mathsf{G}$	$\mathsf{M}(\mathsf{GM}^T + \mathsf{G}) + \mathsf{G}$	$\mathsf{M}(\mathsf{GM}^T + \mathsf{G}) + \mathsf{G}$	①$\mathsf{GM}^T + \mathsf{G}$
binary matrix factorization	①$(\mathsf{BG} + \mathsf{G})(\mathsf{GB}^T + \mathsf{G}) + \mathsf{G}$	$(\mathsf{BG} + \mathsf{G})\mathsf{B}^T + \mathsf{G}$	②$\mathsf{GG} + \mathsf{G}$	②$\mathsf{GG} + \mathsf{G}$
BCTF	$(\mathsf{MG} + \mathsf{G})(\mathsf{GM}^T + \mathsf{G}) + \mathsf{G}$	$(\mathsf{MG} + \mathsf{G})(\mathsf{GM}^T + \mathsf{G}) + \mathsf{G}$	②$\mathsf{GM}^T + \mathsf{G}$	●G
sparse coding	$(\exp(\mathsf{G}) \circ \mathsf{G})\mathsf{G} + \mathsf{G}$	$(\exp(\mathsf{G}) \circ \mathsf{G})\mathsf{G} + \mathsf{G}$	$(\exp(\mathsf{G}) \circ \mathsf{G})\mathsf{G} + \mathsf{G}$	②G
dependent GSM	①$(\exp(\mathsf{G}) \circ \mathsf{G})\mathsf{G} + \mathsf{G}$	①$(\exp(\mathsf{G}) \circ \mathsf{G})\mathsf{G} + \mathsf{G}$	①$(\exp(\mathsf{G}) \circ \mathsf{G})\mathsf{G} + \mathsf{G}$	●$\mathsf{BG} + \mathsf{G}$
random walk	$\mathsf{CG} + \mathsf{G}$	$\mathsf{CG} + \mathsf{G}$	$\mathsf{CG} + \mathsf{G}$	①G
linear dynamical system	$(\mathsf{CG} + \mathsf{G})\mathsf{G} + \mathsf{G}$	$(\mathsf{CG} + \mathsf{G})\mathsf{G} + \mathsf{G}$	$(\mathsf{CG} + \mathsf{G})\mathsf{G} + \mathsf{G}$	②$\mathsf{BG} + \mathsf{G}$

Table 4.1: The structures learned from 200×200 matrices generated from various distributions, with signal variance 1 and noise variance σ^2. Incorrect structures are marked with a 1, 2, or 3, depending how many decisions would need to be changed to find the correct structure. We observe that our approach typically finds the correct answer in low noise settings and backs off to simpler models in high noise settings.

[1] More precisely, if $\frac{S_i - S_{i-1}}{N+D} > 1$, where S_i is the total predictive log-likelihood for the level i model summed over all rows and columns, and N and D are the numbers of rows and columns, respectively. We chose to normalize by $N + D$ because the predictive likelihood improvements between more abstract models tend to grow with the number of rows and columns in the input matrix, rather than the number of entries.

[2] Our grammar generates expressions of the form $\cdots + \mathsf{G}$. We consider this final G term to be the "noise" and the rest to be the "signal," even though the models and algorithms do not distinguish the two.

We observe that seven of the ten structures were identified perfectly in both trials where the noise variance was no larger than the data variance ($\sigma^2 \leq 1$). When $\sigma^2 = 0.1$, the system incorrectly chose $(\mathsf{BG}+\mathsf{G})\mathsf{G}+\mathsf{G}$ for the binary latent feature data, rather than $\mathsf{BG}+\mathsf{G}$. Similarly, it chose $(\mathsf{BG}+\mathsf{G})(\mathsf{GB}^T+\mathsf{G})+\mathsf{G}$ rather than $(\mathsf{BG}+\mathsf{G})\mathsf{B}^T+\mathsf{G}$ for binary matrix factorization. In both cases, the sampler learned an incorrect set of binary features, and the additional flexibility of the more complex model compensated for this. This phenomenon, where more structured models compensate for algorithmic failures in simpler models, has also been noted in the context of deep learning (Salakhutdinov and Murray, 2008).

Our system also did not succeed in learning the dependent Gaussian scale mixture structure $(\exp(\mathsf{GG}+\mathsf{G})\circ\mathsf{G})\mathsf{G}+\mathsf{G}$ from synthetic data, instead generally falling back to the simpler sparse coding model $(\exp(\mathsf{G})\circ\mathsf{G})\mathsf{G}+\mathsf{G}$. For $\sigma^2 = 0.1$ the correct structure was in fact the highest scoring structure, but did not cross our threshold of 1 nat improvement over the previous level. We note that in every case, there were nearly 2500 incorrect structures to choose from, so it is notable that the correct model structure can be recovered most of the time.

In general, when the noise variance was much larger than the signal variance, the system gracefully fell back to simpler models, such as $\mathsf{GM}^T+\mathsf{G}$ instead of the BCTF model $(\mathsf{MG}+\mathsf{G})(\mathsf{GM}^T+\mathsf{G})+\mathsf{G}$ (see Section 3.4.1). At the extreme, in the maximum noise condition, it chose the structureless model G much of the time. Overall, our procedure reliably learned most of the model structures in low-noise settings (impressive considering the extremely large space of possible wrong answers) and gracefully fell back to simpler models when necessary.

4.4.2 Real-world data

Next, we evaluated our system on several real-world datasets. We first consider two domains, motion capture and image statistics, where the core statistical assumptions are widely agreed upon, and verify that our learned structures are consistent with these assumptions. We then turn to domains where the correct structure is more ambiguous and analyze the representations our system learns.

In general, we do not expect every real-world dataset to have a unique best structure. In cases where the predictive likelihood score differences between multiple top-scoring models were not statistically significant, we report the set of top-scoring models and analyze what they have in common.

Motion capture. We first consider a human motion capture dataset (Hsu et al., 2005; Taylor et al., 2007) consisting of a person walking in a variety of styles. Each row of the matrix gives the person's orientation and displacement in one frame, as

	Level 1	Level 2	Level 3
Motion capture	$\mathsf{CG}+\mathsf{G}$	$\mathsf{C}(\mathsf{GG}+\mathsf{G})+\mathsf{G}$	—
Image patches	$\mathsf{GG}+\mathsf{G}$	$(\exp(\mathsf{G})\circ\mathsf{G})\mathsf{G}+\mathsf{G}$	$(\exp(\mathsf{GG}+\mathsf{G})\circ\mathsf{G})\mathsf{G}+\mathsf{G}$
20 Questions	$\mathsf{MG}+\mathsf{G}$	$\mathsf{M}(\mathsf{GG}+\mathsf{G})+\mathsf{G}$	—
Senate votes	$\mathsf{GM}^T+\mathsf{G}$	$(\mathsf{MG}+\mathsf{G})\mathsf{M}^T+\mathsf{G}$	—

Table 4.2: The best performing models at each level of our grammar for real-world datasets. These correspond to plausible structures for the datasets, as discussed in the text.

well as various joint angles. We used 200 frames (6.7 seconds), and 45 state variables. In the first step, the system chose the Markov chain model $\mathsf{CG}+\mathsf{G}$, which assumes that the components of the state evolve continuously but independently over time. Since a person's different joint angles are clearly correlated, the system next captured these correlations with the model $\mathsf{C}(\mathsf{GG}+\mathsf{G})+\mathsf{G}$. This is slightly different from the popular linear dynamical system model $(\mathsf{CG}+\mathsf{G})\mathsf{G}+\mathsf{G}$, but it is more physically correct in the sense that the LDS assumes the deviations of the observations from the low-dimensional subspace must be independent in different time steps, while our learned structure captures the temporal continuity in the deviations.

Natural image patches. We tested the system on the Sparsenet dataset of Olshausen and Field (1996), which consists of 10 images of natural scenes which were blurred and whitened. The rows of the input matrix corresponded to 1,000 patches of size 12×12. In the first stage, the model learned the low-rank representation $\mathsf{GG}+\mathsf{G}$, and in the second stage, it sparsified the linear reconstruction coefficients to give the sparse coding model $(\exp(\mathsf{G})\circ\mathsf{G})\mathsf{G}+\mathsf{G}$. In the third round, it modeled the dependencies between the scale variables by recursively giving them a low-rank representation, giving a dependent Gaussian scale mixture (GSM) model $(\exp(\mathsf{GG}+\mathsf{G})\circ\mathsf{G})\mathsf{G}+\mathsf{G}$ reminiscent of Karklin and Lewicki (2008). A closely related model, $(\exp(\mathsf{GB}^T+\mathsf{G})\circ\mathsf{G})\mathsf{G}+\mathsf{G}$, also achieved a score not significantly lower. Both of these structures resulted in a rank-one factorization of the scale matrix, similar to the radial Gaussianization model of Lyu and Simoncelli (2009) for neighboring wavelet coefficients.

Dependent GSM models (see Section 3.4.3) are the state-of-the-art for a variety of image processing tasks, so it is interesting that this structure can be learned merely from the raw data. We note that a single pass through the grammar reproduces an analogous sequence of models to those discovered by the image statistics research community as discussed in Section 3.4.3.

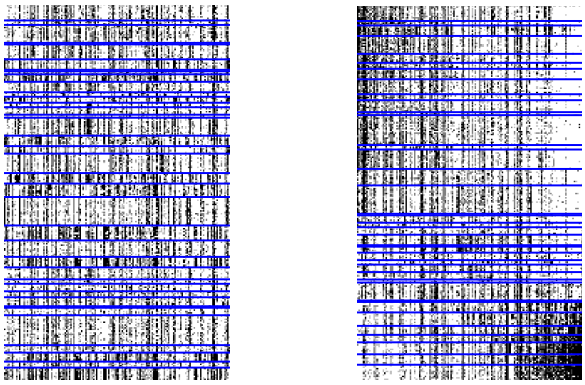
20 Questions. We now consider a dataset collected by Pomerleau et al. (2009) of Mechanical Turk users' responses to 218 questions from the 20 Questions game about

1. Miscellaneous. key, chain, powder, aspirin, umbrella, quarter, cord, sunglasses
2. Clothing. coat, dress, pants, shirt, skirt, backpack, tshirt, quilt, carpet, pillow, clothing
3. Artificial foods. pizza, soup, meat, breakfast, stew, lunch, gum, bread, fries, coffee
4. Machines. bell, telephone, watch, typewriter, lock, channel, tuba, phone, fan, ipod, flute
5. Natural foods. carrot, celery, corn, lettuce, artichoke, pickle, walnut, mushroom, beet
6. Buildings. apartment, barn, church, house, chapel, store, library, camp, school
7. Printed things. card, notebook, ticket, note, napkin, money, journal, menu, letter, mail
8. Body parts. arm, eye, foot, hand, leg, chin, shoulder, lip, teeth, toe, eyebrow, feet, hair
9. Containers. bottle, cup, glass, spoon, pipe, gallon, pan, straw, bin, clipboard, carton
10. Outdoor places. trail, island, earth, yard, town, harbour, river, planet, pond, lawn
11. Tools. knife, chisel, hammer, pliers, saw, screwdriver, screw, dagger, spear, hoe, needle
12. Stuff. speck, gravel, soil, tear, bubble, slush, rust, fat, garbage, crumb, eyelash
13. Furniture. bed, chair, desk, dresser, table, sofa, seat, ladder, mattress, handrail, bench
14. Liquids. wax, honey, pint, disinfectant, gas, drink, milk, water, cola, paste, lemonade
15. Structural features. bumper, cast, fence, billboard, guardrail, axle, deck, dumpster
16. Non-solid things. surf, fire, lightning, sky, steam, cloud, dance, wind, breeze, tornado
17. Transportation. airplane, car, train, truck, jet, sedan, submarine, jeep, boat, tractor
18. Herbivores. cow, horse, lamb, camel, pig, hog, calf, elephant, cattle, giraffe, yak, goat
19. Internal organs. rib, lung, vein, stomach, heart, brain, smile, blood, lap, nerve, lips
20. Carnivores. bear, walrus, shark, crocodile, dolphin, hippo, gorilla, hyena, rhinocerous

Figure 4-1: The 20 largest clusters discovered by our Level 2 model $\mathsf{M}(\mathsf{GG}+\mathsf{G})+\mathsf{G}$ for the 20 Questions dataset. Each line gives **our interpretation**, followed by random items from the cluster.

1000 concrete nouns (e.g. animals, foods, tools). The system began by clustering the entities using the flat clustering model $\mathsf{MG}+\mathsf{G}$. In the second stage, it found low-rank structure in the matrix of cluster centers, resulting in the model $\mathsf{M}(\mathsf{GG}+\mathsf{G})+\mathsf{G}$. No third-level structure achieved more than 1 nat improvement beyond this. The low-rank representation had 8 dimensions, where the largest variance dimension corresponded to living vs. nonliving and the second largest corresponded to large vs. small. The 39 clusters, the 20 largest of which are shown in Figure 4-1, correspond to semantically meaningful categories. Visualizations of the learned models are shown in Figure 4-2.

We note that two other models expressing similar assumptions, $\mathsf{M}(\mathsf{GB}^T+\mathsf{G})+\mathsf{G}$ and $(\mathsf{MG}+\mathsf{G})\mathsf{G}+\mathsf{G}$, achieved scores only slightly lower. What these models have in common is a clustering of entities (but not questions) coupled with low-rank structure between entities and questions. The learned clusters and dimensions are qualitatively similar in each case.



(a) $\mathsf{MG}+\mathsf{G}$ (b) $\mathsf{M}(\mathsf{GG}+\mathsf{G})+\mathsf{G}$

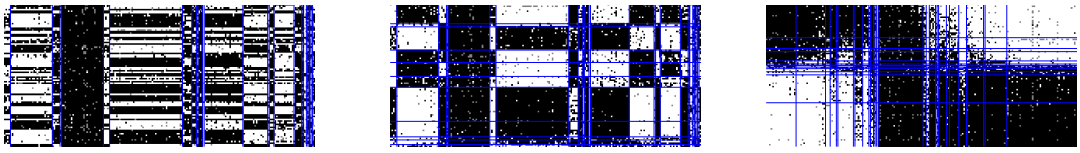
Figure 4-2: Visualizations of the Level 1 representation $\mathsf{MG}+\mathsf{G}$ and the Level 2 representation $\mathsf{M}(\mathsf{GG}+\mathsf{G})+\mathsf{G}$. Rows = entities, columns = questions. 250 rows and 150 columns were selected at random from the original matrix. Rows and columns are sorted first by cluster, then by the highest variance dimension of the low-rank representation (if applicable). Clusters were sorted by the same dimension as well. Blue = cluster boundaries.

Senate voting records. Finally, we consider a dataset of roll call votes from the 111th United States Senate (2009-2010). Rows correspond to Senators, and the columns correspond to all 696 votes, most of which were on procedural motions and amendments to bills. Yea votes were mapped to 1, Nay and Present were mapped to -1, and absences were treated as unobserved. In the first two stages, our procedure clustered the votes and Senators, giving the clustering model $\mathsf{GM}^T + \mathsf{G}$ and the co-clustering model $(\mathsf{MG} + \mathsf{G})\mathsf{M}^T + \mathsf{G}$, respectively. Senators clustered along party lines, as did most of the votes, according to the party of the proposer. The learned representations are all visualized in Figure 4-3.

In the third stage, one of the best performing models was Bayesian clustered tensor factorization (BCTF) (see section 3.4.1), where Senators and votes are each clustered inside a low-rank representation.[3] This low-rank representation was rank 5, with one dominant dimension corresponding to the liberal-conservative axis. The BCTF model makes it clearer that the clusters of Senators and votes are not independent, but can be seen as occupying different points in a low-dimensional representation. This model improved on the previous level by less than our 1 nat cutoff.[4]

[3]The other models whose scores were not significantly different were: $(\mathsf{MG} + \mathsf{G})\mathsf{M}^T + \mathsf{BG} + \mathsf{G}$, $(\mathsf{MG}+\mathsf{G})\mathsf{M}^T + \mathsf{GM}^T + \mathsf{G}$, $\mathsf{G}(\mathsf{GM}^T + \mathsf{G}) + \mathsf{GM}^T + \mathsf{G}$, and $(\mathsf{BG}+\mathsf{G})(\mathsf{GM}^T + \mathsf{G}) + \mathsf{G}$. All of these models include the clustering structure but account for additional variability within clusters.

[4]BCTF results in a more compact representation than the co-clustering model, but our predic-



(a) $\mathsf{GM}^T + \mathsf{G}$ (b) $(\mathsf{MG} + \mathsf{G})\mathsf{M}^T + \mathsf{G}$ (c) $(\mathsf{MG} + \mathsf{G})(\mathsf{GM}^T + \mathsf{G}) + \mathsf{G}$

Figure 4-3: Visualization of the representations learned from the Senate voting data. Rows = Senators, columns = votes. 200 columns were selected at random from the original matrix. Black = yes, white = no, gray = absence. Blue = cluster boundaries. Rows and columns are sorted first by cluster (if applicable), then by the highest variance dimension of the low-rank representation (if applicable). Clusters are sorted by the same dimension as well. The models in the sequence increasingly reflect the polarization of the Senate.

The models in this sequence increasingly highlight the polarization of the Senate.

4.5 Discussion

We have presented an effective and practical method for automatically determining the model structure in a particular space of models, matrix decompositions, by exploiting compositionality. However, we believe our approach can be extended beyond the particular space of models presented here. Most straightforwardly, additional components can be added to capture other motifs of probabilistic modeling, such as tree embeddings and low-dimensional embeddings. More generally, it should be fruitful to investigate other model classes with compositional structure, such as tensor decompositions.

In either case, exploiting the structure of the model space becomes increasingly essential. For instance, the number of models reachable in 3 steps is cubic in the number of production rules, whereas the complexity of the greedy search is linear. For tensors, the situation is even more overwhelming: even if we restrict our attention to analogues of $\mathsf{GG} + \mathsf{G}$, a wide variety of provably distinct generalizations have been identified, including the widely used Tucker3 and PARAFAC decompositions (Kolda and Bader, 2007).

What is the significance of the grammar being context-free? While it imposes

tive likelihood criterion doesn't reward this except insofar as overfitting hurts a model's ability to generalize to new rows and columns. We speculate that a fully Bayesian approach using marginal likelihood may lead to more compact structures.

no restriction on the models themselves, it has the effect that the grammar "over-generates" model structures. Our grammar licenses some nonsensical models: for instance, $\mathsf{G}(\mathsf{MG} + \mathsf{G}) + \mathsf{G}$, which attempts to cluster dimensions of a latent space which is defined only up to affine transformation. Reassuringly, we have never observed such models being selected by our search procedure — a useful sanity check on the output of the algorithm. The only drawback is that the system wastes some time evaluating meaningless models. Just as context-free grammars for English can be augmented with attributes to enforce contextual restrictions such as agreement, our grammar could be similarly extended to rule out unidentifiable models. Such extensions may become important if our approach is applied to a much larger space of models.

Our context-free grammar formalism unifies a wide variety of matrix decomposition models in terms of compositional application of a few production rules. We exploited this compositional structure to efficiently and generically sample from and evaluate a wide variety of latent variable models, both continuous and discrete, flat and hierarchical. Greedy search over our grammar allows us to select a model structure from raw data by evaluating only a small fraction of all models. This search procedure was effective at recovering the correct structure for synthetic data and sensible structures for real-world data. More generally, we believe this work is a proof-of-concept for the practicality of selecting complex model structures in a compositional manner. Since many model spaces other than matrix factorizations are compositional in nature, we hope to spur additional research on automatically searching large, compositional spaces of models.

Chapter 5

Learning composite covariance kernels

This chapter is taken from Duvenaud et al. (2013). While David Duvenaud and James Lloyd were the primary investigators in this work, I include it with the permission of both authors because it is closely related to the main theme of this thesis.

So far, we have seen compositional structure search applied to a space of matrix decomposition structures. We now turn our attention to Gaussian process covariance kernels, another space where models can be constructed compositionally out of a small number of operators and primitives. Similarly to Chapter 4, a recursive search over the grammar yields correct decompositions on synthetic data and interpretable structure on a variety of real-world datasets.

5.1 Introduction

Kernel-based nonparametric models, such as support vector machines and Gaussian processes (GPs), have been one of the dominant paradigms for supervised machine learning over the last 20 years. These methods depend on defining a kernel function, $k(x, x')$, which specifies how similar or correlated outputs y and y' are expected to be at two inputs x and x'. By defining the measure of similarity between inputs, the kernel determines the pattern of inductive generalization.

Most existing techniques pose kernel learning as a (possibly high-dimensional) parameter estimation problem. Examples include learning hyperparameters (Rasmussen and Williams, 2006), linear combinations of fixed kernels (Bach, 2009), and mappings from the input space to an embedding space (Salakhutdinov and Hinton,

2008).

However, to apply existing kernel learning algorithms, the user must specify the parametric form of the kernel, and this can require considerable expertise, as well as trial and error.

To make kernel learning more generally applicable, we reframe the kernel learning problem as one of structure discovery, and automate the choice of kernel form. In particular, we formulate a space of kernel structures defined compositionally in terms of sums and products of a small number of base kernel structures. This provides an expressive modeling language which concisely captures many widely used techniques for constructing kernels. We focus on Gaussian process regression, where the kernel specifies a covariance function, because the Bayesian framework is a convenient way to formalize structure discovery. Borrowing discrete search techniques which have proved successful in equation discovery (Todorovski and Dzeroski, 1997) and unsupervised learning (Grosse et al., 2012), we automatically search over this space of kernel structures using marginal likelihood as the search criterion.

We found that our structure discovery algorithm is able to automatically recover known structures from synthetic data as well as plausible structures for a variety of real-world datasets. On a variety of time series datasets, the learned kernels yield decompositions of the unknown function into interpretable components that enable accurate extrapolation beyond the range of the observations. Furthermore, the automatically discovered kernels outperform a variety of widely used kernel classes and kernel combination methods on supervised prediction tasks.

While we focus on Gaussian process regression, we believe our kernel search method can be extended to other supervised learning frameworks such as classification or ordinal regression, or to other kinds of kernel architectures such as kernel SVMs. We hope that the algorithm developed in this chapter will help replace the current and often opaque art of kernel engineering with a more transparent science of automated kernel construction.

5.2 Expressing structure through kernels

Gaussian process models use a kernel to define the covariance between any two function values: $\mathrm{Cov}(y, y') = k(x, x')$. The kernel specifies which structures are likely under the GP prior, which in turn determines the generalization properties of the model. In this section, we review the ways in which kernel families[1]can be composed

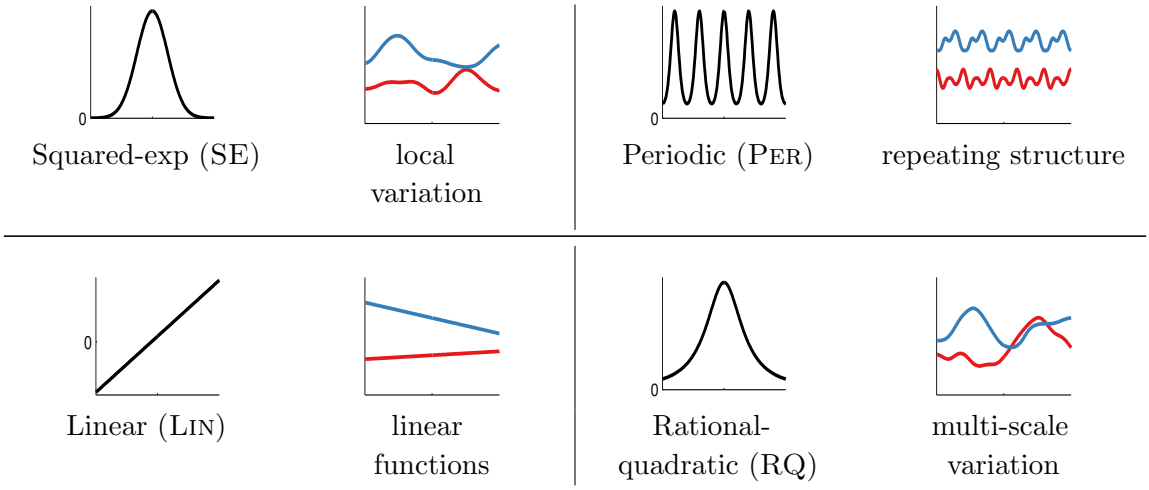


Figure 5-1: Visualizations of some simple kernels. **(left)** the kernel function $k(\cdot, 0)$. **(right)** draws from a GP with each repective kernel. The x-axis has the same range on all plots.

to express diverse priors over functions.

There has been significant work on constructing GP kernels and analyzing their properties, summarized in Chapter 4 of Rasmussen and Williams (2006). Commonly used kernels families include the squared exponential (SE), periodic (PER), linear (LIN), and rational quadratic (RQ). For scalar-valued inputs, these kernels are defined as follows:

$$
\begin{aligned}
k_{\mathrm{SE}}(x, x') &= \sigma^2 \exp\left(-\tfrac{(x-x')^2}{2\ell^2}\right) \\
k_{\mathrm{PER}}(x, x') &= \sigma^2 \exp\left(-\tfrac{2\sin^2(\pi(x-x')/p)}{\ell^2}\right) \\
k_{\mathrm{LIN}}(x, x') &= \sigma_b^2 + \sigma_v^2 (x-\ell)(x'-\ell) \\
k_{\mathrm{RQ}}(x, x') &= \sigma^2 \left(1 + \tfrac{(x-x')^2}{2\alpha\ell^2}\right)^{-\alpha}
\end{aligned}
$$

See Figure 5-1 for visualizations of these kernel families.

Composing Kernels Positive semidefinite kernels (i.e. those which define valid covariance functions) are closed under addition and multiplication. This allows one to create richly structured and interpretable kernels from well understood base components.

[1]When unclear from context, we use 'kernel family' to refer to the parametric forms of the functions given in the appendix. A kernel is a kernel family with all of the parameters specified.

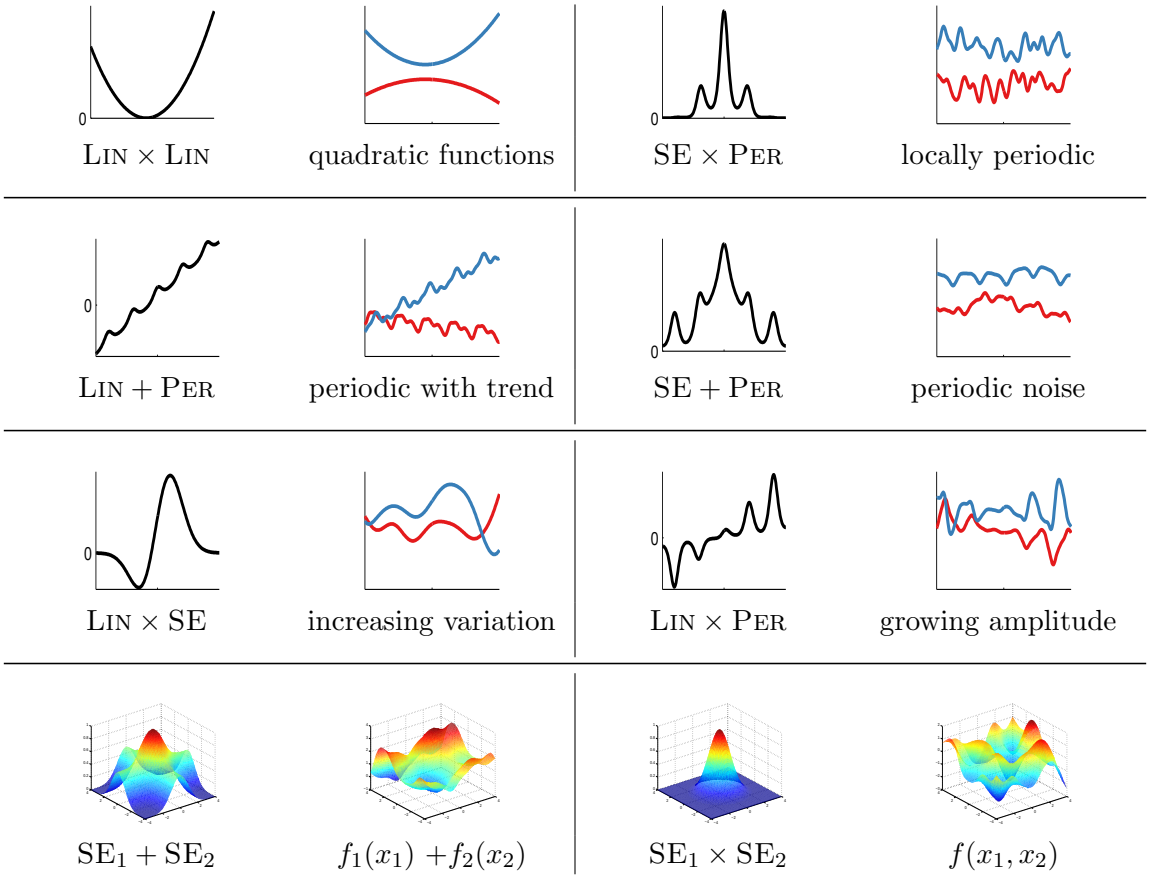


Figure 5-2: Examples of structures expressible by composite kernels. **(left)** composite kernel function $k(\cdot, 0)$. **(right)** samples from the respective GP.

All of the base kernels we use are one-dimensional; kernels over multidimensional inputs are constructed by adding and multiplying kernels over individual dimensions. These dimensions are represented using subscripts, e.g. SE_2 represents an SE kernel over the second dimension of x.

Summation By summing kernels, we can model the data as a superposition of independent functions, possibly representing different structures. Suppose functions f_1, f_2 are draw from independent GP priors, $f_1 \sim \mathcal{GP}(\mu_1, k_1)$, $f_2 \sim \mathcal{GP}(\mu_2, k_2)$. Then $f := f_1 + f_2 \sim \mathcal{GP}(\mu_1 + \mu_2, k_1 + k_2)$.

In time series models, sums of kernels can express superposition of different processes, possibly operating at different scales. In multiple dimensions, summing ker-

nels gives additive structure over different dimensions, similar to generalized additive models (Hastie and Tibshirani, 1990). These two kinds of structure are demonstrated in rows 2 and 4 of Figure 5-2, respectively.

Multiplication Multiplying kernels allows us to account for interactions between different input dimensions or different notions of similarity. For instance, in multidimensional data, the multiplicative kernel $\mathrm{SE}_1 \times \mathrm{SE}_3$ represents a smoothly varying function of dimensions 1 and 3 which is not constrained to be additive. In univariate data, multiplying a kernel by SE gives a way of converting global structure to local structure. For example, PER corresponds to globally periodic structure, whereas $\textsc{Per} \times \mathrm{SE}$ corresponds to locally periodic structure, as shown in row 1 of Figure 5-2.

Many architectures for learning complex functions, such as convolutional networks (LeCun et al., 1989) and sum-product networks (Poon and Domingos, 2011), include units which compute AND-like and OR-like operations. Composite kernels can be viewed in this way too. A sum of kernels can be understood as an OR-like operation: two points are considered similar if either kernel has a high value. Similarly, multiplying kernels is an AND-like operation, since two points are considered similar only if both kernels have high values. Since we are applying these operations to the similarity functions rather than the regression functions themselves, compositions of even a few base kernels are able to capture complex relationships in data which do not have a simple parametric form.

Example expressions In addition to the examples given in Figure 5-2, many common motifs of supervised learning can be captured using sums and products of one-dimensional base kernels:

Bayesian linear regression	LIN
Bayesian polynomial regression	$\textsc{Lin} \times \textsc{Lin} \times \ldots$
Generalized Fourier decomposition	$\textsc{Per} + \textsc{Per} + \ldots$
Generalized additive models	$\sum_{d=1}^{D} \mathrm{SE}_d$
Automatic relevance determination	$\prod_{d=1}^{D} \mathrm{SE}_d$
Linear trend with local deviations	$\textsc{Lin} + \mathrm{SE}$
Linearly growing amplitude	$\textsc{Lin} \times \mathrm{SE}$

We use the term 'generalized Fourier decomposition' to express that the periodic functions expressible by a GP with a periodic kernel are not limited to sinusoids.

5.3 Searching over structures

As discussed above, we can construct a wide variety of kernel structures compositionally by adding and multiplying a small number of base kernels. In particular, we consider the four base kernel families discussed in Section 5.2: SE, PER, LIN, and RQ. Any algebraic expression combining these kernels using the operations + and × defines a kernel family, whose parameters are the concatenation of the parameters for the base kernel families.

Our search procedure begins by proposing all base kernel families applied to all input dimensions. We allow the following search operators over our set of expressions:

(1) Any subexpression $\mathcal{S}$ can be replaced with $\mathcal{S} + \mathcal{B}$, where $\mathcal{B}$ is any base kernel family.

(2) Any subexpression $\mathcal{S}$ can be replaced with $\mathcal{S} \times \mathcal{B}$, where $\mathcal{B}$ is any base kernel family.

(3) Any base kernel $\mathcal{B}$ may be replaced with any other base kernel family $\mathcal{B}'$.

These operators can generate all possible algebraic expressions. To see this, observe that if we restricted the + and × rules only to apply to base kernel families, we would obtain a context-free grammar (CFG) which generates the set of algebraic expressions. However, the more general versions of these rules allow more flexibility in the search procedure, which is useful because the CFG derivation may not be the most straightforward way to arrive at a kernel family.

Our algorithm searches over this space using a greedy search: at each stage, we choose the highest scoring kernel and expand it by applying all possible operators.

Our search operators are motivated by strategies researchers often use to construct kernels. In particular,

- One can look for structure, e.g. periodicity, in the residuals of a model, and then extend the model to capture that structure. This corresponds to applying rule (1).

- One can start with structure, e.g. linearity, which is assumed to hold globally, but find that it only holds locally. This corresponds to applying rule (2) to obtain the structure shown in rows 1 and 3 of Figure 5-2.

- One can add features incrementally, analogous to algorithms like boosting, backfitting, or forward selection. This corresponds to applying rules (1) or (2) to dimensions not yet included in the model.

Scoring kernel families Choosing kernel structures requires a criterion for evaluating structures. We choose marginal likelihood as our criterion, since it balances the fit and complexity of a model (Rasmussen and Ghahramani, 2001). Conditioned on kernel parameters, the marginal likelihood of a GP can be computed analytically. However, to evaluate a kernel family we must integrate over kernel parameters. We approximate this intractable integral with the Bayesian information criterion (Schwarz, 1978) after first optimizing to find the maximum-likelihood kernel parameters.

Unfortunately, optimizing over parameters is not a convex optimization problem, and the space can have many local optima. For example, in data with periodic structure, integer multiples of the true period (i.e. harmonics) are often local optima. To alleviate this difficulty, we take advantage of our search procedure to provide reasonable initializations: all of the parameters which were part of the previous kernel are initialized to their previous values. All parameters are then optimized using conjugate gradients, randomly restarting the newly introduced parameters. This procedure is not guaranteed to find the global optimum, but it implements the commonly used heuristic of iteratively modeling residuals.

5.4 Related Work

Nonparametric regression in high dimensions Nonparametric regression methods such as splines, locally weighted regression, and GP regression are popular because they are capable of learning arbitrary smooth functions of the data. Unfortunately, they suffer from the curse of dimensionality: it is very difficult for the basic versions of these methods to generalize well in more than a few dimensions. Applying nonparametric methods in high-dimensional spaces can require imposing additional structure on the model.

One such structure is additivity. Generalized additive models (GAM) assume the regression function is a transformed sum of functions defined on the individual dimensions: $\mathbb{E}[f(\mathbf{x})] = g^{-1}(\sum_{d=1}^{D} f_d(x_d))$. These models have a limited compositional form, but one which is interpretable and often generalizes well. In our grammar, we can capture analogous structure through sums of base kernels along different dimensions.

It is possible to add more flexibility to additive models by considering higher-order interactions between different dimensions. Additive Gaussian processes (Duvenaud et al., 2011) are a GP model whose kernel implicitly sums over all possible products of one-dimensional base kernels. Plate (1999) constructs a GP with a composite kernel, summing an SE kernel along each dimension, with an SE-ARD kernel (i.e. a product

of SE over all dimensions). Both of these models can be expressed in our grammar.

A closely related procedure is smoothing-splines ANOVA (Wahba, 1990; Gu, 2002). This model is a linear combination of splines along each dimension, all pairs of dimensions, and possibly higher-order combinations. Because the number of terms to consider grows exponentially in the order, in practice, only terms of first and second order are usually considered.

Semiparametric regression (e.g. Ruppert et al., 2003) attempts to combine interpretability with flexibility by building a composite model out of an interpretable, parametric part (such as linear regression) and a 'catch-all' nonparametric part (such as a GP with an SE kernel). In our approach, this can be represented as a sum of SE and LIN.

Kernel learning There is a large body of work attempting to construct a rich kernel through a weighted sum of base kernels (e.g. Christoudias et al., 2009; Bach, 2009). While these approaches find the optimal solution in polynomial time, speed comes at a cost: the component kernels, as well as their hyperparameters, must be specified in advance.

Another approach to kernel learning is to learn an embedding of the data points. Lawrence (2005) learns an embedding of the data into a low-dimensional space, and constructs a fixed kernel structure over that space. This model is typically used in unsupervised tasks and requires an expensive integration or optimisation over potential embeddings when generalizing to test points. Salakhutdinov and Hinton (2008) use a deep neural network to learn an embedding; this is a flexible approach to kernel learning but relies upon finding structure in the input density, p(x). Instead we focus on domains where most of the interesting structure is in f(x).

Wilson and Adams (2013) derive kernels of the form SE $\times \cos(x - x')$, forming a basis for stationary kernels. These kernels share similarities with SE $\times$ PER but can express negative prior correlation, and could usefully be included in our grammar.

Diosan et al. (2007) and Bing et al. (2010) learn composite kernels for support vector machines and relevance vector machines, using genetic search algorithms. Our work employs a Bayesian search criterion, and goes beyond this prior work by demonstrating the interpretability of the structure implied by composite kernels, and how such structure allows for extrapolation.

Structure discovery There have been several attempts to uncover the structural form of a dataset by searching over a grammar of structures. For example, Schmidt and Lipson (2009), Todorovski and Dzeroski (1997) and Washio et al. (1999) attempt to learn parametric forms of equations to describe time series, or relations between

quantities. Because we learn expressions describing the covariance structure rather than the functions themselves, we are able to capture structure which does not have a simple parametric form.

Kemp and Tenenbaum (2008) learned the structural form of a graph used to model human similarity judgments. Examples of graphs included planes, trees, and cylinders. Some of their discrete graph structures have continous analogues in our own space; e.g. $\mathrm{SE}_1 \times \mathrm{SE}_2$ and $\mathrm{SE}_1 \times \mathrm{PER}_2$ can be seen as mapping the data to a plane and a cylinder, respectively.

5.5 Structure discovery in time series

To investigate our method's ability to discover structure, we ran the kernel search on several time series.

As discussed in Section 2, a GP whose kernel is a sum of kernels can be viewed as a sum of functions drawn from component GPs. This provides another method of visualizing the learned structures. In particular, all kernels in our search space can be equivalently written as sums of products of base kernels by applying distributivity. For example,

$$\mathrm{SE} \times (\mathrm{RQ} + \mathrm{LIN}) = \mathrm{SE} \times \mathrm{RQ} + \mathrm{SE} \times \mathrm{LIN}.$$

We visualize the decompositions into sums of components using the following formula for the conditional distribution of one component given the sum:

$$\mathbf{f}_1|\mathbf{f} \sim \mathcal{N}\big(\mu_1 + \mathbf{K}_1{}^{\mathsf{T}}(\mathbf{K}_1 + \mathbf{K}_2)^{-1}\left(\mathbf{f} - \mu_1 - \mu_2\right), \\ \mathbf{K}_1 - \mathbf{K}_1{}^{\mathsf{T}}(\mathbf{K}_1 + \mathbf{K}_2)^{-1}\mathbf{K}_1\big).$$

The search was run to depth 10, using the base kernels from Section 5.2.

Mauna Loa atmospheric $\mathbf{CO_2}$ Using our method, we analyzed records of carbon dioxide levels recorded at the Mauna Loa observatory. Since this dataset was analyzed in detail by Rasmussen and Williams (2006), we can compare the kernel chosen by our method to a kernel constructed by human experts.

Figure 5-3 shows the posterior mean and variance on this dataset as the search depth increases. While the data can be smoothly interpolated by a single base kernel model, the extrapolations improve dramatically as the increased search depth allows more structure to be included.

Figure 5-4 shows the final model chosen by our method, together with its decomposition into additive components. The final model exhibits both plausible ex-

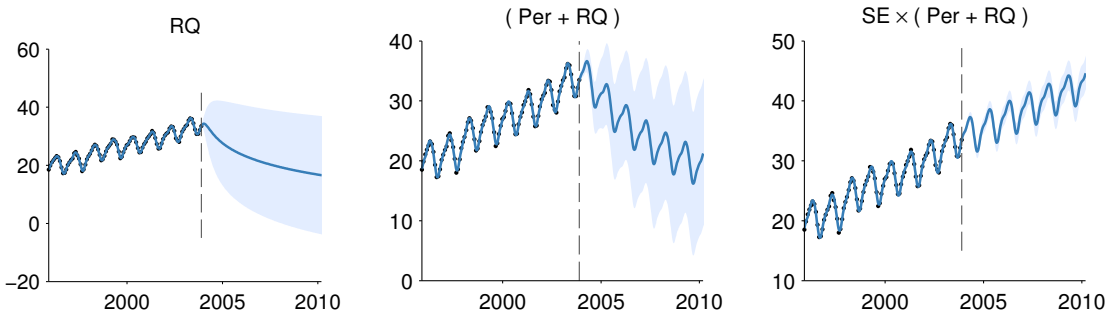


Figure 5-3: Posterior mean and variance for different depths of kernel search. The dashed line marks the extent of the dataset. In the first column, the function is only modeled as a locally smooth function, and the extrapolation is poor. Next, a periodic component is added, and the extrapolation improves. At depth 3, the kernel can capture most of the relevant structure, and is able to extrapolate reasonably.

trapolation and interpretable components: a long-term trend, annual periodicity and medium-term deviations; the same components chosen by Rasmussen and Williams (2006). We also plot the residuals, observing that there is little obvious structure left in the data.

Airline passenger data Figure 5-6 shows the decomposition produced by applying our method to monthly totals of international airline passengers (Box et al., 1976). We observe similar components to the previous dataset: a long-term trend, annual periodicity, and medium-term deviations. In addition, the composite kernel captures the near-linearity of the long-term trend and the linearly growing amplitude of the annual oscillations.

Solar irradiance Data Finally, we analyzed annual solar irradiation data from 1610 to 2011 (Lean et al., 1995). The posterior and residuals of the learned kernel are shown in Figure 5-5. None of the models in our search space are capable of parsimoniously representing the lack of variation from 1645 to 1715. Despite this, our approach fails gracefully: the learned kernel still captures the periodic structure, and the quickly growing posterior variance demonstrates that the model is uncertain about long term structure.

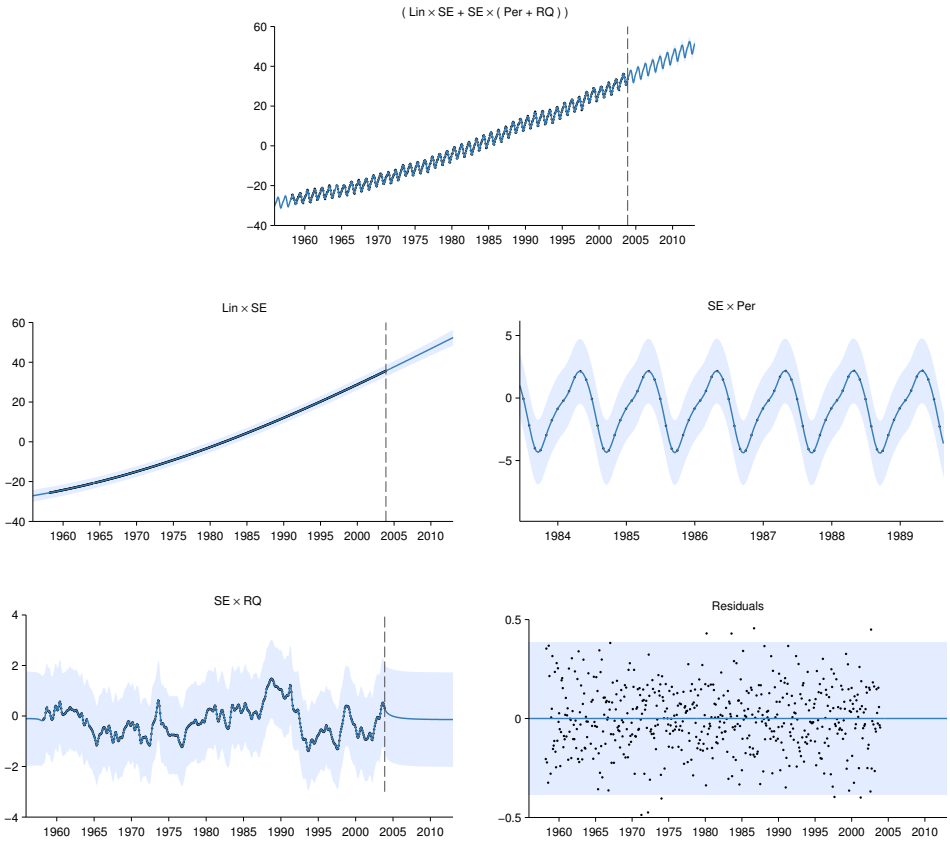


Figure 5-4: **Top:** The posterior on the Mauna Loa dataset, after a search of depth 10. **Bottom:** The posterior decomposition into a sum of four components. The decompositions shows long-term, yearly periodic, medium-term anomaly components, and residuals, respectively. In the top right plot, the scale has been changed in order to clearly show the yearly periodic structure.

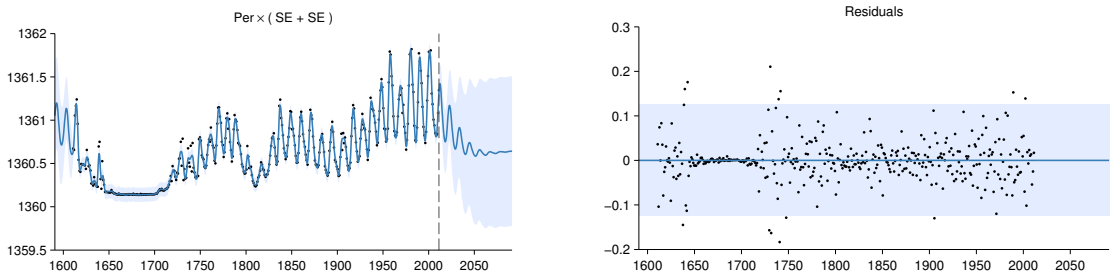


Figure 5-5: Full posterior (left) and residuals (right) on the solar irradiance dataset.

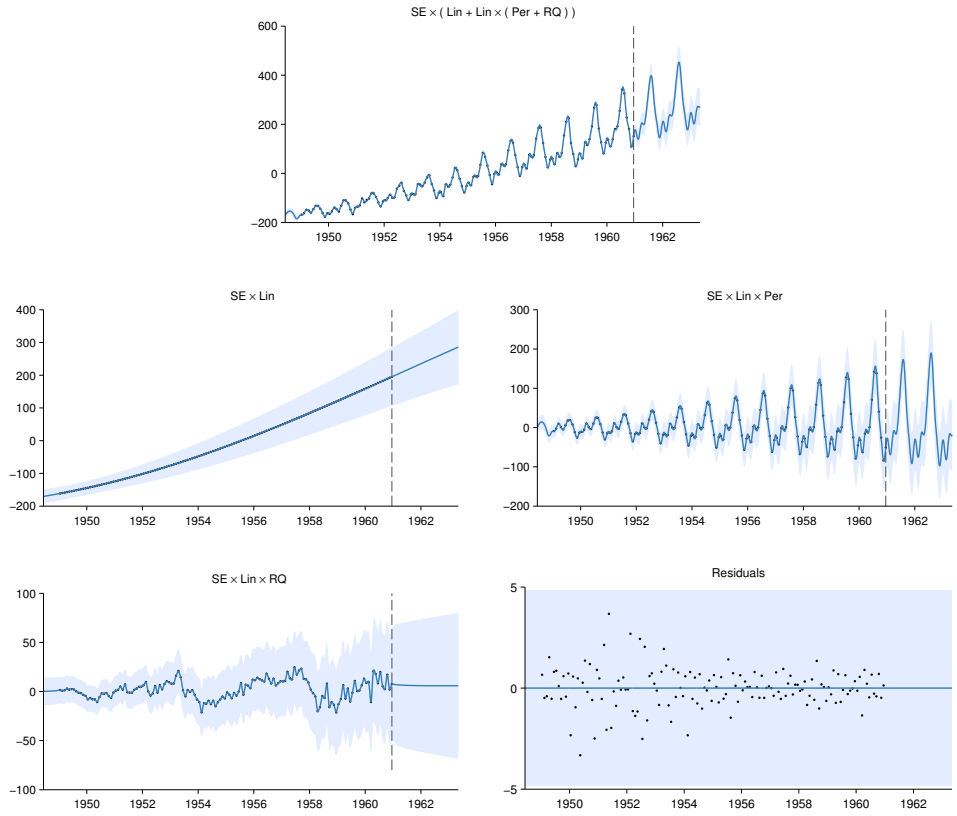


Figure 5-6: **Top:** The airline dataset and posterior after a search of depth 10. **Bottom:** Additive decomposition of the posterior into long-term smooth trend, yearly variation, and short-term deviations. Due to the linear kernel, the marginal variance grows over time, making this a heteroskedastic model.

True Kernel	D	SNR = 10	SNR = 1	SNR = 0.1
$\mathrm{SE} + \mathrm{RQ}$	1	SE	$\mathrm{SE} \times \textsc{Per}$	SE
$\textsc{Lin} \times \textsc{Per}$	1	$\textsc{Lin} \times \textsc{Per}$	$\textsc{Lin} \times \textsc{Per}$	SE
$\mathrm{SE}_1 + \mathrm{RQ}_2$	2	$\mathrm{SE}_1 + \mathrm{SE}_2$	$\textsc{Lin}_1 + \mathrm{SE}_2$	$\textsc{Lin}_1$
$\mathrm{SE}_1 + \mathrm{SE}_2 \times \textsc{Per}_1 + \mathrm{SE}_3$	3	$\mathrm{SE}_1 + \mathrm{SE}_2 \times \textsc{Per}_1 + \mathrm{SE}_3$	$\mathrm{SE}_2 \times \textsc{Per}_1 + \mathrm{SE}_3$	-
$\mathrm{SE}_1 \times \mathrm{SE}_2$	4	$\mathrm{SE}_1 \times \mathrm{SE}_2$	$\textsc{Lin}_1 \times \mathrm{SE}_2$	$\textsc{Lin}_2$
$\mathrm{SE}_1 \times \mathrm{SE}_2 + \mathrm{SE}_2 \times \mathrm{SE}_3$	4	$\mathrm{SE}_1 \times \mathrm{SE}_2 + \mathrm{SE}_2 \times \mathrm{SE}_3$	$\mathrm{SE}_1 + \mathrm{SE}_2 \times \mathrm{SE}_3$	SE_1
$(\mathrm{SE}_1 + \mathrm{SE}_2) \times (\mathrm{SE}_3 + \mathrm{SE}_4)$	4	$(\mathrm{SE}_1 + \mathrm{SE}_2) \times (\mathrm{SE}_3 \times \textsc{Lin}_3 \times \textsc{Lin}_1 + \mathrm{SE}_4)$	$(\mathrm{SE}_1 + \mathrm{SE}_2) \times \mathrm{SE}_3 \times \mathrm{SE}_4$	-

Table 5.1: Kernels chosen by our method on synthetic data generated using known kernel structures. D denotes the dimension of the functions being modeled. SNR indicates the signal-to-noise ratio. Dashes - indicate no structure.

5.6 Validation on synthetic data

We validated our method's ability to recover known structure on a set of synthetic datasets. For several composite kernel expressions, we constructed synthetic data by first sampling 300 points uniformly at random, then sampling function values at those points from a GP prior. We then added i.i.d. Gaussian noise to the functions, at various signal-to-noise ratios (SNR).

Table 5.1 lists the true kernels we used to generate the data. Subscripts indicate which dimension each kernel was applied to. Subsequent columns show the dimensionality D of the input space, and the kernels chosen by our search for different SNRs. Dashes - indicate that no kernel had a higher marginal likelihood than modeling the data as i.i.d. Gaussian noise.

For the highest SNR, the method finds all relevant structure in all but one test. The reported additional linear structure is explainable by the fact that functions sampled from SE kernels with long length scales occasionally have near-linear trends. As the noise increases, our method generally backs off to simpler structures.

5.7 Quantitative evaluation

In addition to the qualitative evaluation in Section 5.5, we investigated quantitatively how our method performs on both extrapolation and interpolation tasks.

	Mean Squared Error (MSE)					Negative Log-Likelihood				
Method	bach	concrete	puma	servo	housing	bach	concrete	puma	servo	housing
Linear Regression	1.031	0.404	0.641	0.523	0.289	2.430	1.403	1.881	1.678	1.052
GAM	1.259	0.149	0.598	0.281	0.161	1.708	0.467	1.195	0.800	0.457
HKL	**0.199**	0.147	0.346	0.199	0.151	-	-	-	-	-
GP SE-ARD	**0.045**	0.157	0.317	0.126	**0.092**	**−0.131**	0.398	0.843	0.429	0.207
GP Additive	**0.045**	**0.089**	**0.316**	**0.110**	0.102	**−0.131**	**0.114**	**0.841**	**0.309**	0.194
Structure Search	**0.044**	**0.087**	**0.315**	**0.102**	**0.082**	**−0.141**	**0.065**	**0.840**	**0.265**	**0.059**

Table 5.2: Comparison of multidimensional regression performance. Bold results are not significantly different from the best-performing method in each experiment, in a paired t-test with a p-value of 5%.

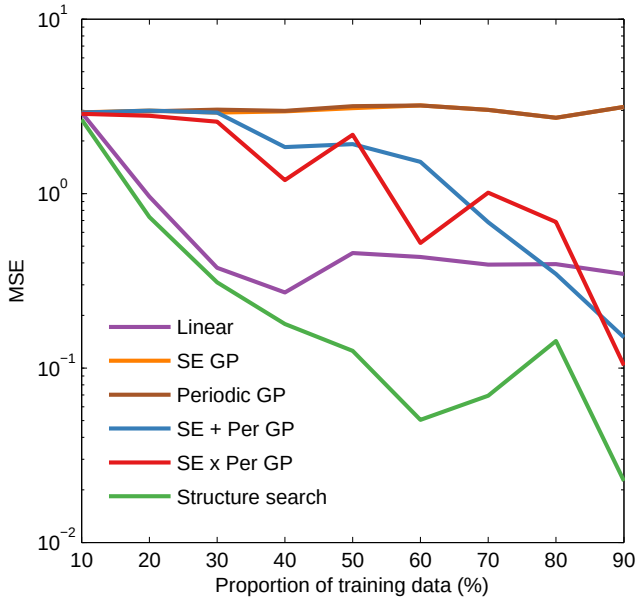


Figure 5-7: Extrapolation performance on the airline dataset. We plot test-set MSE as a function of the fraction of the dataset used for training.

5.7.1 Extrapolation

We compared the extrapolation capabilities of our model against standard baselines[2]. Dividing the airline dataset into contiguous training and test sets, we computed the predictive mean-squared-error (MSE) of each method. We varied the size of the training set from the first 10% to the first 90% of the data.

Figure 5-7 shows the learning curves of linear regression, a variety of fixed kernel family GP models, and our method. GP models with only SE and PER kernels did not capture the long-term trends, since the best parameter values in terms of GP marginal likelihood only capture short term structure. Linear regression approximately captured the long-term trend, but quickly plateaued in predictive performance. The more richly structured GP models (SE + PER and SE × PER) eventually captured more structure and performed better, but the full structures discovered by our search outperformed the other approaches in terms of predictive performance for all data amounts.

5.7.2 High-dimensional prediction

To evaluate the predictive accuracy of our method in a high-dimensional setting, we extended the comparison of Duvenaud et al. (2011) to include our method. We performed 10-fold cross-validation on 5 datasets[3] comparing 5 methods in terms of MSE and predictive likelihood. Our structure search was run up to depth 10, using the SE and RQ base kernel families.

The comparison included three methods with fixed kernel families: Additive GPs, Generalized Additive Models (GAM), and a GP with a standard SE kernel using Automatic Relevance Determination (GP SE-ARD). Also included was the related kernel-search method of Hierarchical Kernel Learning (HKL).

Results are presented in Table 5.2. Our method outperformed the next-best method in each test, although not substantially.

All GP hyperparameter tuning was performed by automated calls to the GPML toolbox[4]; Python code to perform all experiments is available on github[5].

[2]In one dimension, the predictive means of all baseline methods in table 5.2 are identical to that of a GP with an SE kernel.

[3]The data sets had dimensionalities ranging from 4 to 13, and the number of data points ranged from 150 to 450.

[4]Available at `www.gaussianprocess.org/gpml/code/`

[5]`github.com/jamesrobertlloyd/gp-structure-search`

5.8 Discussion

> "It would be very nice to have a formal apparatus that gives us some 'optimal' way of recognizing unusual phenomena and inventing new classes of hypotheses that are most likely to contain the true one; but this remains an art for the creative human mind."
>
> E. T. Jaynes, 1985

Towards the goal of automating the choice of kernel family, we introduced a space of composite kernels defined compositionally as sums and products of a small number of base kernels. The set of models included in this space includes many standard regression models. We proposed a search procedure for this space of kernels which parallels the process of scientific discovery.

We found that the learned structures are often capable of accurate extrapolation in complex time-series datasets, and are competitive with widely used kernel classes and kernel combination methods on a variety of prediction tasks. The learned kernels often yield decompositions of a signal into diverse and interpretable components, enabling model-checking by humans. We believe that a data-driven approach to choosing kernel structures automatically can help make nonparametric regression and classification methods accessible to non-experts.

Chapter 6

Compositional importance sampling

Chapter 4 presented a technique for greedily initializing the inference procedure in matrix decomposition models. The initialization procedure was entirely compositional: it required only writing special-purpose inference procedures for the individual factorization models. However, other aspects of the algorithms were not compositional. First, the recursive initialization was followed by full joint inference over the entire model. Second, the models were evaluated in terms of predictive likelihood computed on the raw observations, which required marginalizing over multiple components jointly.

Aesthetically, it seems odd that these non-compositional operators should be required. As people, we continually learn higher level abstractions, and we don't need to continually assess how these high-level abstractions relate to the light hitting our retinas. Can we perform inference and structure learning using *only* algorithms corresponding to the production rules?

Compositionality would have practical benefits when implementing a structure search system. The compositional nature of the recursive initialization procedure allowed samplers to be written independently for different production rules. By contrast, implementing predictive likelihood scoring required considering combinations of disparate models, such as Gaussian scale mixtures and Markov chains. We speculate that predictive likelihood scoring would constitute the largest obstacle to extending the grammar of Chapter 3 with additional production rules, or with non-Gaussian observation models.

In this chapter, we present compositional importance sampling (CIS), an inference and model scoring algorithm defined *entirely* in terms of recursive application of a

handful of algorithms. CIS returns both a marginal likelihood estimate and an approximate posterior sample from the model. Its implementation is modular: it requires procedures which yield posterior samples and marginal likelihood estimates for the individual production rules. (If one only requires posterior samples, CIS is equivalent to the recursive initialization procedure of Chapter 4, and one need not implement marginal likelihood estimators.)

The main technical contribution of this chapter is a theoretical analysis of the performance of CIS in the case of a clustering-within-low-rank model similar to $\mathsf{G}(\mathsf{G}\mathsf{M}^T + \mathsf{G}) + \mathsf{G}$. Using a Gaussian approximation to the posterior distribution, we bound the bias of the marginal likelihood estimates and the KL divergence of the samples from the true posterior. Under certain assumptions about the data, this translates into consistency guarantees for inference and model selection.

In the context of this thesis, this chapter serves two purposes. First, it presents a fully compositional procedure for posterior inference and marginal likelihood estimation. Second, the theoretical analysis also informs the accuracy of the greedy initialization procedure of Chapter 4, and can therefore be seen as a theoretical explanation for the empirical results of that chapter.

6.1 Compositional importance sampling

We begin by repeating the observation from Section 2.3.3 that importance sampling yields an unbiased estimator of a partition function:

$$\hat{\mathcal{Z}} \triangleq \frac{1}{S} \sum_{s=1}^{S} \frac{f(\mathbf{x}^{(s)})}{q(\mathbf{x}^{(s)})}, \tag{6.1}$$

where q is a normalized proposal distribution, f is the unnormalized target distribution, and the $\mathbf{x}^{(s)}$ denote samples from q. Compositional importance sampling (CIS) is simply the special case of this estimator where the proposal and target distributions are two different models from the grammar which differ only through the application of a single production rule.

For concreteness, consider the case where q denotes the low rank model $\mathsf{G}\mathsf{G} + \mathsf{G}$ and p denotes the clustering-within-low-rank model $\mathsf{G}(\mathsf{G}\mathsf{M}^T + \mathsf{G}) + \mathsf{G}$. Let $\mathbf{Y} \approx \mathbf{U}\mathbf{V}$ denote the factorization of the input matrix, where $\mathbf{U}$ has a Gaussian prior, and $\mathbf{V}$ has a Gaussian prior in q and a mixture of Gaussians prior in p. (This model is

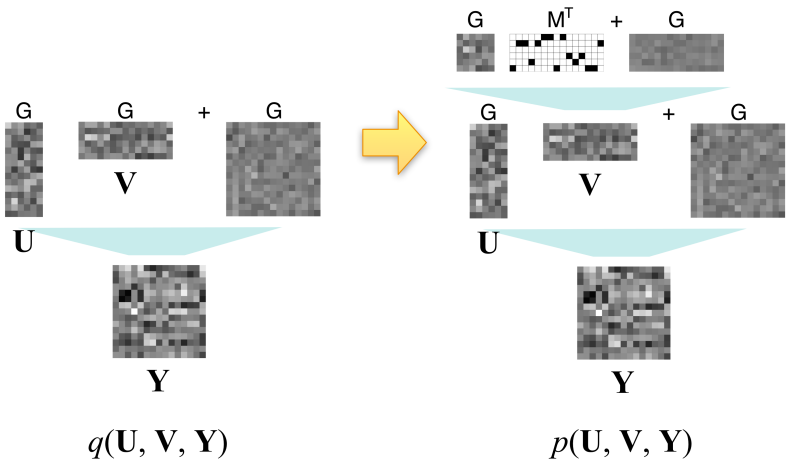


Figure 6-1: Estimating the marginal likelihood under the clustering-within-low-rank model $\mathsf{G}(\mathsf{G}\mathsf{M}^T + \mathsf{G}) + \mathsf{G}$ using CIS.

shown in Figure 6-1.) The two models q and p differ only in their prior over $\mathbf{V}$:

$$\begin{aligned} q(\mathbf{U}, \mathbf{V}, \mathbf{Y}) &= p(\mathbf{U})\, q(\mathbf{V})\, p(\mathbf{Y} \mid \mathbf{U}, \mathbf{V}) \\ p(\mathbf{U}, \mathbf{V}, \mathbf{Y}) &= p(\mathbf{U})\, p(\mathbf{V})\, p(\mathbf{Y} \mid \mathbf{U}, \mathbf{V}) \end{aligned} \tag{6.2}$$

In order to evaluate the marginal likelihood $p(\mathbf{Y})$, we apply the estimator (6.1) with $q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})$ as the proposal distribution:

$$\begin{aligned} p(\mathbf{Y}) &= q(\mathbf{Y})\, \mathbb{E}_{\mathbf{U},\mathbf{V} \sim q(\mathbf{U},\mathbf{V} \mid \mathbf{Y})} \left[\frac{p(\mathbf{U}, \mathbf{V}, \mathbf{Y})}{q(\mathbf{U}, \mathbf{V}, \mathbf{Y})} \right] \\ &= q(\mathbf{Y})\, \mathbb{E}_{\mathbf{U},\mathbf{V} \sim q(\mathbf{U},\mathbf{V} \mid \mathbf{Y})} \left[\frac{p(\mathbf{U})\, p(\mathbf{V})\, p(\mathbf{Y} \mid \mathbf{U}, \mathbf{V})}{p(\mathbf{U})\, q(\mathbf{V})\, p(\mathbf{Y} \mid \mathbf{U}, \mathbf{V})} \right] \\ &= q(\mathbf{Y})\, \mathbb{E}_{\mathbf{U},\mathbf{V} \sim q(\mathbf{U},\mathbf{V} \mid \mathbf{Y})} \left[\frac{p(\mathbf{V})}{q(\mathbf{V})} \right] \end{aligned} \tag{6.3}$$

The CIS estimator is

$$\hat{p}(\mathbf{Y}) = q(\mathbf{Y}) \frac{1}{S} \sum_{s=1}^{S} \frac{p(\mathbf{V}^{(s)})}{q(\mathbf{V}^{(s)})}, \tag{6.4}$$

where the $\mathbf{V}^{(s)}$ are posterior samples from $q(\mathbf{V} \mid \mathbf{Y})$.

Now consider how to compute this quantity. First, $q(\mathbf{Y})$ is the marginal likelihood

of $\mathbf{Y}$ under the low-rank model $\mathsf{GG}+\mathsf{G}$. We sample $\mathbf{U}$ and $\mathbf{V}$ from $q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})$, which entails performing posterior inference in the low rank model. $q(\mathbf{V})$ is the probability of $\mathbf{V}$ under an *i.i.d.* Gaussian prior, which has a closed-form solution. Finally, $p(\mathbf{V})$ is the marginal likelihood of $\mathbf{V}$ under the clustering model $\mathsf{GM}^T + \mathsf{G}$. Hence, we have reduced the problem of estimating marginal likelihood in the clustering-within-low-rank model to that of inference and marginal likelihood computation in simple factorization models. More generally, applying CIS to the models of Chapter 3 requires only inference and marginal likelihood estimation algorithms corresponding to the production rules of the grammar.

Note that (6.4) represents an idealized version of the estimator where posterior inference and marginal likelihood computation are performed exactly in the simple models. In practice, one would use approximate algorithms for both. If annealed importance sampling (AIS; Neal, 2001a) is used for both the inference and marginal likelihood estimation, CIS remains an unbiased estimator of $p(\mathbf{Y})$ even with the approximation.

By itself, the fact that CIS is unbiased is a very weak guarantee. It is easy to construct an unbiased estimator of the marginal likelihood; one example is importance sampling from the prior. The important question is: how accurate are the results? This question is analyzed in Section 6.2.

6.1.1 Relationship with model checking

One motivation for CIS is that it serves as a form of model checking. Specifically, as discussed in Gelman et al. (2014, chap. 7), one heuristic for checking models is to look at the estimated parameters, latent variables, or residuals, and look for structure not already encoded by the model. If there is such structure, it suggests that the original model is inadequate. The CIS estimator (6.4) can be thought of as "looking for more structure" in $\mathbf{V}$ by comparing its marginal likelihood under a structured prior p to the marginal likelihood under the original generic prior q. If $\log p(\mathbf{V})$ exceeds $\log q(\mathbf{V})$ by more than a few nats, then p almost certainly[1] achieves a higher marginal likelihood on the original data. In this way, CIS could be part of a model checking procedure which automatically suggests elaborations of a model.

[1]The ratio in CIS is an unbiased estimator of the ratio of the two marginal likelihoods. As discussed in Section 2.3.1, unbiased estimators of positive quantities are unlikely to overestimate the true value by very much.

6.2 Theoretical analysis

As described in Section 6.1, CIS is an unbiased estimator of the marginal likelihood $p(\mathbf{Y})$. By the analysis of Section 2.3.1, it can therefore be viewed as a stochastic lower bound on $\log p(\mathbf{Y})$:

$$\mathbb{E}[\log \hat{p}(\mathbf{Y})] < \log p(\mathbf{Y}). \tag{6.5}$$

A natural way to evaluate the estimator is in terms of the bias

$$\delta \triangleq \log p(\mathbf{Y}) - \mathbb{E}[\log \hat{p}(\mathbf{Y})]. \tag{6.6}$$

The bias is relevant to model comparison: suppose we are interested in comparing two models p and p' such that $\log p(\mathbf{Y})$ and $\log p'(\mathbf{Y})$ differ by at least r nats. Then a given estimator is able to distinguish them if $\delta < r$ for each model, assuming one averages over enough trials.[2] Therefore, the bias gives a measure of the resolution of a model selection scheme. (See Section 2.3.1 for more background on evaluating partition function estimators.)

The bias for a single sample (*i.e.* $S = 1$) is given by

$$\begin{aligned}
\delta &= \log p(\mathbf{Y}) - \mathbb{E}_{q(\mathbf{U},\mathbf{V} \mid \mathbf{Y})} \left[\log p(\mathbf{U}, \mathbf{V}, \mathbf{Y}) - \log q(\mathbf{U}, \mathbf{V}, \mathbf{Y}) + \log q(\mathbf{Y})\right] \\
&= \mathbb{E}_{q(\mathbf{U},\mathbf{V} \mid \mathbf{Y})} \left[\log q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y}) - \log p(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})\right] \\
&= \mathrm{D}_{\mathrm{KL}}(q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y}) \,\|\, p(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})).
\end{aligned} \tag{6.7}$$

Using multiple samples may improve the estimates, but our analysis assumes $S = 1$.

Due to (6.7), our analysis of the bias doubles as an analysis of the recursive initialization procedure of Chapter 4. Specifically, suppose we first sample $(\mathbf{U}, \mathbf{V}) \sim q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})$, and then recursively sample a factorization $\mathcal{F}$ of $\mathbf{V}$ from $p(\mathcal{F} \mid \mathbf{V})$. The joint distribution is given by $q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})\, p(\mathcal{F} \mid \mathbf{V})$, and

$$\begin{aligned}
\mathrm{D}_{\mathrm{KL}}\left(q(\mathbf{U}, \mathbf{V}, \mathcal{F} \mid \mathbf{Y}) \,\|\, p(\mathbf{U}, \mathbf{V}, \mathcal{F} \mid \mathbf{Y})\right) &= \mathrm{D}_{\mathrm{KL}}\left(q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})\, p(\mathcal{F} \mid \mathbf{V}) \,\|\, p(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})\, p(\mathcal{F} \mid \mathbf{V})\right) \\
&= \mathrm{D}_{\mathrm{KL}}\left(q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y}) \,\|\, p(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})\right) \\
&= \delta.
\end{aligned} \tag{6.8}$$

Therefore, the KL divergence of the recursive initialization from the true posterior is *exactly* the bias of the CIS estimator. The rest of this section is devoted to analyzing δ for the case of a clustering-within-low-rank model.

[2] It is also necessary that the estimator not have infinite or unreasonably large variance. While unbiased estimators of marginal likelihood typically have extremely large variance, this is not true of the corresponding log marginal likelihood estimators, as discussed in Section 2.3.1.

6.2.1 The model

For concreteness, we focus on the case of a clustering-within-low-rank model which roughly corresponds to $\mathsf{G}(\mathsf{GM}^T + \mathsf{G}) + \mathsf{G}$ in the grammar of Chapter 3. Section 6.2.5 discusses the significance of the differences and speculates about how the analysis could be extended to other models.

We define q to be a low rank factorization model which is a slight variant of $\mathsf{GG} + \mathsf{G}$, and p to be a clustering-within-low-rank model which is a slight variant of $\mathsf{G}(\mathsf{GM}^T + \mathsf{G}) + \mathsf{G}$. Both q and p agree on the generative process for $\mathbf{U}$: each row is sampled independently from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Under the prior $q(\mathbf{V})$, the columns are independent multivariate Gaussians:

1. Sample a precision matrix $\mathbf{\Lambda_V}$ from the weakly informative prior

$$p_{wi}(\mathbf{\Lambda_V}) \propto \begin{cases} |\mathbf{\Lambda_V}|^{-(K+1)/2} & \text{if } \epsilon\mathbf{I} \preceq \mathbf{\Lambda_V} \preceq \epsilon^{-1}\mathbf{I} \\ 0 & \text{otherwise,} \end{cases} \tag{6.9}$$

   where ϵ is small. (This choice of prior is motivated in Appendix A.3.1.)

2. Sample each column of $\mathbf{V}$ as $\mathbf{v}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{\Lambda_V}^{-1})$.

The prior $p(\mathbf{V})$ is a clustering model:

1. Sample a parameter r from an inverse gamma distribution. This represents the ratio of between-cluster to within-cluster variance.

2. Sample the within-cluster precision matrix $\mathbf{\Lambda_V}$ from p_{wi}.

3. Sample a cluster assignments vector $\mathbf{z}$ from a Dirichlet-multinomial distribution with K' possible assignments.

4. Sample K' mean vectors $\boldsymbol{\mu}_k$ *i.i.d.* from $\mathcal{N}(\mathbf{0}, r\mathbf{\Lambda_V}^{-1})$.

5. Sample each column $\mathbf{v}_j$ from $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{z}_j}, \mathbf{\Lambda_V}^{-1})$.

Finally, the observation model is spherical Gaussian noise with variance λ^{-1}. As a simplifying assumption, we assume that λ and the dimension K of the low rank factorization are fixed. (All other parameters are sampled from the posterior.)

The distributions q and p are nearly the same as $\mathsf{GG} + \mathsf{G}$ and $\mathsf{G}(\mathsf{GM}^T + \mathsf{G}) + \mathsf{G}$, respectively. The differences are:

- $\mathbf{U}$ is assumed to have spherical, rather than diagonal, covariance. This does not make the model any less expressive, because the variances of the latent dimensions can be absorbed into $\mathbf{\Lambda_V}$. (This is analogous to factor analysis, where the latent factors are assumed WLOG to have spherical covariance.)
- $\mathbf{V}$ has full, rather than diagonal, covariance.
- The within-cluster and between-cluster variances for $\mathbf{V}$ are constrained to be identical up to a scale factor. By contrast, in the model of Chapter 3, the two were independent.

The significance of these differences is discussed in Section 6.2.5.

6.2.2 Additional assumptions

We do not wish to assume the data was generated by the model, because in the context of compositional model selection, the models are believed to be oversimplified. However, we do assume a weaker form of correctness: that the residuals are uncorrelated. In particular, if $(\mathbf{U}, \mathbf{V})$ is a sample from the posterior $p(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})$, and $\mathbf{R} = \mathbf{UV} - \mathbf{Y}$ is the matrix of residuals, then we assume that $\|\mathbf{R}\|_2 = O(\sqrt{N + D})$, where $\| \cdot \|_2$ denotes the matrix 2-norm (largest singular value). This is the rate of growth which obtains *a.s.* for random matrices with *i.i.d.* entries and a weak condition on the moments (Geman, 1980). (If there were correlations, the growth rate could be $O(\sqrt{ND})$.)

The assumption of uncorrelated residuals is not overly restrictive, because any significant correlations in $\mathbf{R}$ can be explained by increasing K. We note that looking for correlations in residuals is a common model checking heuristic. Beyond this, we do not make any additional assumptions about the data. In particular, we don't assume that the fixed parameters λ or K are correct, and we don't assume the data were generated by the model of Section 6.2.1.

6.2.3 Bias of the estimator

We now state our main result; the derivation is given in the next section. For the clustering-within-low-rank model given above, with the assumption about the residuals, and using Gaussian approximations to the posterior distributions,

$$\delta \leq \frac{3D}{2\lambda N} \mathbb{E}[\operatorname{tr} \mathbf{\Lambda_V}] + o(N^{-1}D). \tag{6.10}$$

Here, N is the number of rows, D is the number of columns, λ is the precision of the observation noise (in the model, rather than the data), and $\mathbf{\Lambda_V}$ is the within-cluster precision of the columns of $\mathbf{V}$. (The expectation is with respect to the posterior $p(\mathbf{\Lambda_V} \mid \mathbf{Y})$.)

Consider the dependence on each of the factors. If N is increased, the number of observations increases, while the number of unknowns remains fixed. Therefore, we would expect the KL divergence to shrink with N. Similarly, if D is increased, the number of unknowns grows, while the number of observations per unknown is fixed. Therefore, we should expect the KL divergence to grow with D. Increasing λ increases the weighting of the evidence relative to the prior. Because q and p share the evidence term, increasing λ should increase the overlap between the posteriors. Finally, a larger $\mathbf{\Lambda_V}$ corresponds to higher demands on the accuracy of $\mathbf{V}$. All of these are reflected in (6.10).

To better understand this result, suppose $\mathbf{\Lambda_V}$ is chosen such that:

- all K latent dimensions have equal variance
- the signal variance is fixed to 1
- the ratio of total variance to within-cluster variance is γ.

This corresponds to $\mathbf{\Lambda_V} = \gamma K \mathbf{I}$, and therefore $\operatorname{tr} \mathbf{\Lambda_V} = \gamma K^2$. Under this assumption, the bias is

$$\frac{3\gamma K^2 D}{2\lambda N}. \tag{6.11}$$

Is this accuracy sufficient for model selection? This depends on the difference $\log p(\mathbf{Y}) - \log q(\mathbf{Y})$. Observe that the matrix $\mathbf{V}$ has KD entries. Therefore, if each entry of $\mathbf{V}$ is explained better under p than under q, we would expect

$$\log p(\mathbf{Y}) = \log q(\mathbf{Y}) + \Theta(KD).$$

Consider the following asymptotic regimes:

- $N \to \infty$, D fixed. The bias decays to zero while the log marginal likelihood difference approaches a constant. Therefore, CIS will be able to distinguish the two models.
- N fixed, $D \to \infty$. The bound (6.11) and the log marginal likelihood difference are both proportional to D, so it is not obvious from the analysis whether CIS can distinguish the models in this case.

- $N \to \infty, D \to \infty, D/N$ constant. The bound (6.11) approaches a constant while the log marginal likelihood difference is proportional to D. CIS is able to distinguish the models.

Alternatively, we can ask whether the greedy initialization procedure yields good top-level representations. Suppose that the inference procedure makes an error such as merging two distinct clusters. Since the number of columns belonging to these two clusters is proportional to D, the KL divergence would need to grow as $\Omega(D)$ for this error to remain in the limit. However, in the regime where $N \to \infty$, $D \to \infty$, and D/N is constant, the bound (6.11) approaches a constant. Therefore, in this asymptotic regime, the greedy initialization procedure will learn the correct set of clusters given a large enough input matrix.

6.2.4 Derivation

The derivation of (6.10) proceeds in four steps. First, we reduce the problem to the case of fixed cluster assignments and variance parameters for $p(\mathbf{V})$. Second, we determine based on symmetries of the model that certain directions of variability can be ignored when analyzing the KL divergence. Third, we compute the second-order Taylor approximations of the log-posterior distributions around a posterior sample from $p(\cdot \mid \mathbf{Y})$. Finally, we analyze the KL divergence in the second-order approximation to derive the bound given above.

6.2.4.1 Reduction to fixed hyperparameters

Let $\boldsymbol{\eta} = (\mathbf{z}, \boldsymbol{\Lambda}_{\mathbf{V}}, r)$ denote all parameters and latent variables associated with $p(\mathbf{V})$. We find that:

$$\begin{aligned}
\mathrm{D}_{\mathrm{KL}}\left(q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y}) \,\|\, p(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})\right) &= \mathbb{E}_{q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})}\left[\log q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y}) - \log p(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})\right] \\
&= \mathbb{E}_{q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})}\left[\log q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y}) - \log \mathbb{E}_{p(\boldsymbol{\eta} \mid \mathbf{Y})}\left[p(\mathbf{U}, \mathbf{V} \mid \mathbf{Y}, \boldsymbol{\eta})\right]\right] \\
&\leq \mathbb{E}_{q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})}\left[\log q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y}) - \mathbb{E}_{p(\boldsymbol{\eta} \mid \mathbf{Y})}\left[\log p(\mathbf{U}, \mathbf{V} \mid \mathbf{Y}, \boldsymbol{\eta})\right]\right] \\
&= \mathbb{E}_{p(\boldsymbol{\eta} \mid \mathbf{Y}), q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})}\left[\log q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y}) - \log p(\mathbf{U}, \mathbf{V} \mid \mathbf{Y}, \boldsymbol{\eta})\right] \\
&= \mathbb{E}_{p(\boldsymbol{\eta} \mid \mathbf{Y})}\left[\mathrm{D}_{\mathrm{KL}}\left(q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y}) \,\|\, p(\mathbf{U}, \mathbf{V} \mid \mathbf{Y}, \boldsymbol{\eta})\right)\right]
\end{aligned}$$

Therefore, we can limit ourselves to considering a fixed $\boldsymbol{\eta}$ corresponding to a posterior sample from $p(\boldsymbol{\eta} \mid \mathbf{Y})$.

6.2.4.2 Model symmetries

Our immediate goal is to take a second-order Taylor approximation to $\log p(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})$ and $\log q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y}, \boldsymbol{\eta})$ around a sample $(\mathbf{U}_0, \mathbf{V}_0)$. Typically, this is justified through an asymptotic argument that for large N and D, most posterior mass is concentrated near some pair $(\mathbf{U}_0, \mathbf{V}_0)$. Unfortunately, this is not true in the present case, because the model has a rotational symmetry. In particular, if $\mathbf{Q}$ is a rotation matrix, one can apply a transformation of the form

$$\begin{aligned} \mathbf{U} &\mapsto \mathbf{U}\mathbf{Q} \\ \mathbf{V} &\mapsto \mathbf{Q}^T\mathbf{V} \end{aligned} \tag{6.12}$$

and obtain an equivalent explanation of the data, because $\mathbf{U}\mathbf{Q}\mathbf{Q}^T\mathbf{V} = \mathbf{U}\mathbf{V}$. All such explanations have equal probability under the prior. Therefore, the posterior probability mass does not concentrate around a point.

However, q and p were chosen such that they agree on all transformations of this form. In particular, for any invertible matrices $\mathbf{T}_1$ and $\mathbf{T}_2$,

$$\frac{p(\mathbf{U}\mathbf{T}_1, \mathbf{T}_2\mathbf{V} \mid \mathbf{Y})}{q(\mathbf{U}\mathbf{T}_1, \mathbf{T}_2\mathbf{V} \mid \mathbf{Y})} = \frac{p(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})}{q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})}. \tag{6.13}$$

See Appendix A.2 for details. Recall that we are interested in determining $\delta = \mathbb{E}_{q(\mathbf{U},\mathbf{V} \mid \mathbf{Y})}[\log q(\mathbf{U}, \mathbf{V} \mid \mathbf{Y}) - \log p(\mathbf{U}, \mathbf{V} \mid \mathbf{Y})]$. Eqn 6.13 implies that variations corresponding to linear transformations of the representations do not contribute to the KL divergence. Therefore, we "factor out" this variability when we compute the second-order approximation.

In particular, let $\mathrm{d}\mathbf{V} = \mathbf{V} - \mathbf{V}_0$. We can uniquely decompose $\mathrm{d}\mathbf{V}$ into a component $\mathrm{d}\mathbf{V}_{\text{row}}$ whose rows lie in the row space of $\mathbf{V}_0$, and a component $\mathrm{d}\mathbf{V}_\perp$ whose rows lie in the null space of $\mathbf{V}_0$:

$$\begin{aligned} \mathrm{d}\mathbf{V}_{\text{row}} &= \mathrm{d}\mathbf{V}\mathbf{V}_0{}^T(\mathbf{V}_0\mathbf{V}_0{}^T)^{-1}\mathbf{V}_0 \\ \mathrm{d}\mathbf{V}_\perp &= \mathrm{d}\mathbf{V} - \mathrm{d}\mathbf{V}_{\text{row}}. \end{aligned} \tag{6.14}$$

From the definition of $\mathrm{d}\mathbf{V}_{\text{row}}$,

$$\mathbf{V}_0 + \mathrm{d}\mathbf{V}_{\text{row}} = \mathbf{V}_0 + \mathrm{d}\mathbf{T}_2\mathbf{V}_0 = (\mathbf{I} + \mathrm{d}\mathbf{T}_2)\mathbf{V}_0$$

for some $\mathrm{d}\mathbf{T}_2$. Therefore, $\mathrm{d}\mathbf{V}_{\text{row}}$ corresponds to a linear transformation of the latent space, and can therefore be ignored. The two components are visualized in Figure 6-

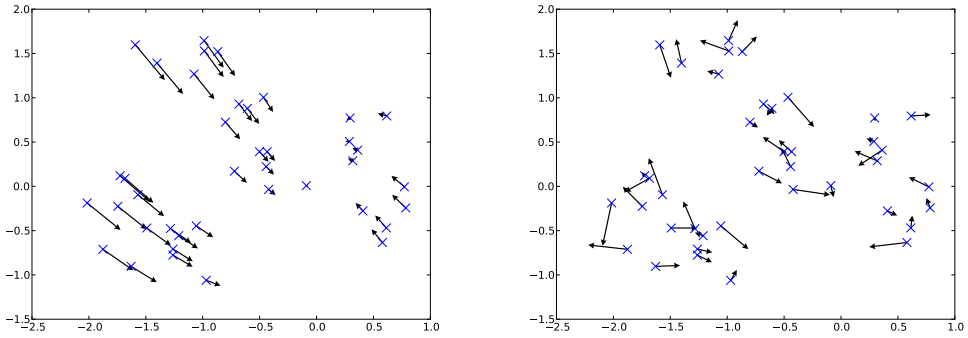


Figure 6-2: Visualization of the components $\mathrm{d}\mathbf{V}_{\text{row}}$ (left) and $\mathrm{d}\mathbf{V}_{\perp}$ (right) for $K = 2$. Each x represents one column of $\mathbf{V}_0$, and the corresponding arrows represent the columns of $\mathrm{d}\mathbf{V}$. The component $\mathrm{d}\mathbf{V}_{\text{row}}$ corresponds to linear transformations of the latent space, which can be ignored because of the symmetry in the priors $q(\mathbf{V})$ and $p(\mathbf{V})$.

2. We now restrict $\mathbf{V}$ to vary only within the null space of $\mathbf{V}_0$, *i.e.* we constrain

$$(\mathbf{V} - \mathbf{V}_0)\mathbf{V}_0^T = \mathbf{0}. \tag{6.15}$$

6.2.4.3 Second-order Taylor approximation

In the true posterior, $\mathbf{U}$ and $\mathbf{V}$ interact nonlinearly, which makes them difficult to analyze. In this section, we construct an approximation to the posterior where $\mathbf{U}$ and $\mathbf{V}$ are jointly Gaussian. In particular, let $(\mathbf{U}_0, \mathbf{V}_0, \boldsymbol{\eta})$ denote a posterior sample from p. We take a second-order Taylor approximation around $(\mathbf{U}_0, \mathbf{V}_0)$ with respect to $\mathbf{U}$ and $\mathbf{V}$, where $\boldsymbol{\eta}$ is held fixed. (See Section 6.2.4.1 for the justification for fixing $\boldsymbol{\eta}$ to a posterior sample.)

As discussed in the previous section, we only allow $\mathbf{V}$ to vary within the null space of $\mathbf{V}_0$. Let $\mathbf{V}_\perp = \mathbf{V} - \mathbf{V}_0$. We give some potentials in terms of $\mathbf{V}$ and some in terms of $\mathbf{V}_\perp$, depending which is clearer in context; however, the Taylor approximation is always with respect to $\mathbf{U}$ and $\mathbf{V}_\perp$.

In order to define the approximation, we need the Kronecker product notation. In particular, vec($\mathbf{U}$) denotes stacking the columns of $\mathbf{U}$ into a vector, and $\mathbf{U} \otimes \mathbf{V}$

is a matrix whose blocks are given by

$$\begin{pmatrix} u_{11}\mathbf{V} & u_{12}\mathbf{V} & & u_{1K}\mathbf{V} \\ u_{21}\mathbf{V} & u_{22}\mathbf{V} & & u_{2K}\mathbf{V} \\ & & \ddots & \\ u_{N1}\mathbf{V} & u_{N2}\mathbf{V} & & u_{NK}\mathbf{V} \end{pmatrix}$$

Let $\mathbf{F}$ denote the permutation matrix which implements matrix transpose in the vectorized representation:

$$\mathbf{F}\,\mathrm{vec}(\mathbf{X}) = \mathrm{vec}(\mathbf{X}^T). \tag{6.16}$$

First, $\mathbf{U}$ is distributed as a spherical Gaussian:

$$q(\mathbf{U}) = p(\mathbf{U}) = \mathcal{N}^{-1}(\mathrm{vec}(\mathbf{U}); \mathbf{0}, \mathbf{I} \otimes \mathbf{I}). \tag{6.17}$$

The second-order Taylor approximation to $\log q(\mathbf{V})$ is given by:

$$q_{\mathrm{so}}(\mathbf{V}_\perp) = \mathcal{N}^{-1}\left(\mathrm{vec}(\mathbf{V}_\perp); \mathbf{0}, \mathbf{I} \otimes D(\mathbf{V}_0\mathbf{V}_0^T)^{-1}\right). \tag{6.18}$$

The details are in Appendix A.3.1. Intuitively, this corresponds to all columns of $\mathbf{V}$ being drawn *i.i.d.* from zero-mean Gaussians whose covariance $\hat{\mathbf{\Lambda}}_{\mathbf{V}}^{-1} = \frac{1}{D}\mathbf{V}\mathbf{V}^T$ is the empirical covariance of the columns of $\mathbf{V}$. (Note, however, that this approximation holds *only* in the null space of $\mathbf{V}_0$.)

As for $p(\mathbf{V} \mid \boldsymbol{\eta})$, we can write $\mathbf{V}$ as

$$\mathbf{V} = \mathbf{\Xi}\mathbf{Z}^T + \mathbf{E}, \tag{6.19}$$

where $\mathbf{\Xi}$ denotes the cluster centers, which have precision $r^{-1}\mathbf{\Lambda}_{\mathbf{V}}$, $\mathbf{Z}$ denotes the cluster assignments (as a binary matrix), and $\mathbf{E}$ denotes the within-cluster variation, which has precision $\mathbf{\Lambda}_{\mathbf{V}}$. The assignments $\mathbf{Z}$ are fixed as part of $\boldsymbol{\eta}$ (see Section 6.2.4.1), while $\mathbf{\Xi}$ and $\mathbf{E}$ are marginalized out. The covariance of $\mathrm{vec}(\mathbf{V})$ is $\boldsymbol{S}^{-1} \otimes \mathbf{\Lambda}_{\mathbf{V}}^{-1}$, where $\boldsymbol{S}^{-1} = r\mathbf{Z}\mathbf{Z}^T + \mathbf{I}$. *I.e.*,

$$p(\mathbf{V} \mid \boldsymbol{\eta}) = \mathcal{N}^{-1}\left(\mathrm{vec}(\mathbf{V}); \mathbf{0}, \boldsymbol{S} \otimes \mathbf{\Lambda}_{\mathbf{V}}\right). \tag{6.20}$$

Written as a function of $\mathbf{V}_\perp$ (see Appendix A.3.1 for details),

$$p(\mathbf{V} \mid \boldsymbol{\eta}) \propto \mathcal{N}^{-1}\left(\mathrm{vec}(\mathbf{V}_\perp);\ -\mathrm{vec}(\mathbf{\Lambda}_{\mathbf{V}}\mathbf{V}_0\boldsymbol{S}),\ \boldsymbol{S} \otimes \mathbf{\Lambda}_{\mathbf{V}}\right). \tag{6.21}$$

The observation term is given by

$$q(\mathbf{Y} \mid \mathbf{U}, \mathbf{V}) = p(\mathbf{Y} \mid \mathbf{U}, \mathbf{V}) \propto \exp\left(-\frac{\lambda}{2} \|\mathbf{Y} - \mathbf{U}\mathbf{V}\|_F^2\right), \tag{6.22}$$

with the second order approximation

$$\begin{aligned}\log p_{\text{so}}(\mathbf{Y} \mid \mathbf{U}, \mathbf{V}) = \text{const} + \lambda \operatorname{tr} \mathbf{Y}\mathbf{V}_0^T\mathbf{U}^T \\ - \frac{\lambda}{2}\left(\operatorname{vec}(\mathbf{U})^T \operatorname{vec}(\mathbf{V}_\perp)^T\right) \begin{pmatrix} \mathbf{V}_0\mathbf{V}_0^T \otimes \mathbf{I} & \mathbf{F}(\mathbf{R} \otimes \mathbf{I}) \\ (\mathbf{R}^T \otimes \mathbf{I})\mathbf{F} & \mathbf{I} \otimes \mathbf{U}_0^T\mathbf{U}_0 \end{pmatrix} \begin{pmatrix} \operatorname{vec}(\mathbf{U}) \\ \operatorname{vec}(\mathbf{V}_\perp) \end{pmatrix}\end{aligned} \tag{6.23}$$

where $\mathbf{R} = \mathbf{U}_0\mathbf{V}_0 - \mathbf{Y}$ is the matrix of residuals and $\mathbf{F}$ is defined in (6.16). (See Appendix A.3.2.)

Finally, we combine all of these terms into second-order approximations to the log-posteriors with respect to $\mathbf{U}$ and $\mathbf{V}_\perp$. Take $\tilde{\mathbf{V}}_\perp = \mathbf{V}_\perp\mathbf{Q}^T$, where $\mathbf{Q}$ is a basis for the null space of $\mathbf{V}_0$. The second-order approximations are as follows:

$$\begin{aligned} q_{\text{post}}(\mathbf{U}, \mathbf{V}) &\propto p(\mathbf{U})\, q_{\text{so}}(\mathbf{V})\, p_{\text{so}}(\mathbf{Y} \mid \mathbf{U}, \mathbf{V}) \\ &\propto \mathcal{N}^{-1}\left(\begin{pmatrix} \operatorname{vec}(\mathbf{U}) \\ \operatorname{vec}(\tilde{\mathbf{V}}_\perp) \end{pmatrix}; \begin{pmatrix} \mathbf{h_u} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C}_q \end{pmatrix}\right) \end{aligned} \tag{6.24}$$

$$\begin{aligned} p_{\text{post}}(\mathbf{U}, \mathbf{V}) &\propto p(\mathbf{U})\, p(\mathbf{V} \mid \boldsymbol{\eta})\, p_{\text{so}}(\mathbf{Y} \mid \mathbf{U}, \mathbf{V}) \\ &\propto \mathcal{N}^{-1}\left(\begin{pmatrix} \operatorname{vec}(\mathbf{U}) \\ \operatorname{vec}(\tilde{\mathbf{V}}_\perp) \end{pmatrix}; \begin{pmatrix} \mathbf{h_u} \\ \mathbf{h_v} \end{pmatrix}, \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C}_p \end{pmatrix}\right), \end{aligned} \tag{6.25}$$

where

$$\begin{aligned} \mathbf{h_u} &\triangleq \lambda \operatorname{vec}(\mathbf{Y}\mathbf{V}_0^T) \\ \mathbf{h_v} &\triangleq -\operatorname{vec}(\boldsymbol{\Lambda}_\mathbf{V}\mathbf{V}_0\boldsymbol{S}\mathbf{Q}^T) \\ \mathbf{A} &\triangleq (\lambda\mathbf{V}_0\mathbf{V}_0^T + \mathbf{I}) \otimes \mathbf{I} \\ \mathbf{B} &\triangleq \lambda\mathbf{F}(\mathbf{R}_\perp \otimes \mathbf{I}) \\ \mathbf{C}_q &\triangleq \lambda\mathbf{I} \otimes \mathbf{U}_0^T\mathbf{U}_0 + \mathbf{I} \otimes D(\mathbf{V}_0\mathbf{V}_0^T)^{-1} \\ \mathbf{C}_p &\triangleq \lambda\mathbf{I} \otimes \mathbf{U}_0^T\mathbf{U}_0 + \boldsymbol{S}_\perp \otimes \boldsymbol{\Lambda}_\mathbf{V}. \end{aligned} \tag{6.26}$$

Here, $\mathbf{R}_\perp = \mathbf{R}\mathbf{Q}^T$ and $\boldsymbol{S}_\perp = \mathbf{Q}\boldsymbol{S}\mathbf{Q}^T$ are the projections of $\mathbf{R}$ and $\boldsymbol{S}$ onto the null

space of $\mathbf{V}_0$.

6.2.4.4 Analyzing the Gaussian case

The second-order Taylor approximation to the log-posterior defines a multivariate Gaussian distribution, as long as the Hessian is negative definite. This is verified in Appendix A.4.2. We now analyze the KL divergences between the two Gaussian approximations q_{post} and p_{post}. As shown in Appendix A.4.3, the KL divergence decomposes as $\mathrm{D}_{\mathrm{KL}}\left(q_{\text{post}}(\mathbf{U}, \tilde{\mathbf{V}}_\perp) \,\|\, p_{\text{post}}(\mathbf{U}, \tilde{\mathbf{V}}_\perp)\right) = \mathcal{T}_1 + \mathcal{T}_2 + \mathcal{T}_3$, where

$$\begin{aligned}
\mathcal{T}_1 &= \frac{1}{2}\log\left|\mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right| - \frac{1}{2}\log\left|\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right| \\
\mathcal{T}_2 &= \frac{1}{2}\operatorname{tr}\boldsymbol{\Sigma}_{\mathbf{v}}(\mathbf{C}_p - \mathbf{C}_q) \\
\mathcal{T}_3 &= \frac{1}{2}\boldsymbol{\Delta}_{\boldsymbol{\mu}}^T(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B})\boldsymbol{\Delta}_{\boldsymbol{\mu}} \\
\boldsymbol{\Sigma}_{\mathbf{v}} &= \left(\mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1} \\
\boldsymbol{\Delta}_{\boldsymbol{\mu}} &= \operatorname{vec}\left(\mathbb{E}_{p_{\text{post}}}[\tilde{\mathbf{V}}_\perp] - \mathbb{E}_{q_{\text{post}}}[\tilde{\mathbf{V}}_\perp]\right).
\end{aligned} \tag{6.27}$$

These terms each have intuitive interpretations. $\mathcal{T}_1$ corresponds to the ratio of the volumes of the two posteriors. (We would expect $\mathcal{T}_1$ to be negative, since p_{post} will ordinarily be more peaked than q_{post}. However, for the present discussion, it suffices to bound it.) $\mathcal{T}_2$ can be thought of as a variance term: some directions may have higher variance under q_{post} than under p_{post} – likely those directions corresponding to within-cluster variation. Finally, $\mathcal{T}_3$ can be thought of as a bias term: the two posteriors have different means.

These three terms are analyzed in Appendices A.4.4 through A.4.6. Each term is bounded above by

$$\frac{D}{2\lambda N}\operatorname{tr}\boldsymbol{\Lambda}_{\mathbf{V}}$$

plus smaller order terms, so the KL divergence is bounded by

$$\frac{3D}{2\lambda N}\operatorname{tr}\boldsymbol{\Lambda}_{\mathbf{V}} \tag{6.28}$$

plus smaller order terms. The meaning of this result is elaborated in Section 6.2.3.

6.2.5 Extension to other models

In this section, we have analyzed the accuracy of the CIS estimator, and the greedy initialization procedure of Chapter 4, for a particular clustering-within-low-rank model. This model corresponds to a slightly modified version of $\mathsf{G}(\mathsf{G}\mathsf{M}^T + \mathsf{G}) + \mathsf{G}$. However, we would ultimately like a way to determine, for a given model, whether these techniques are likely to give good results. While we don't have a formal statement of this, we informally discuss all of the modeling assumptions that were used in the analysis, and why they were required. Based on this, we speculate about what models CIS can be applied to.

- **Fixed latent dimension K and noise variance λ^{-1}.** We assumed both of these quantities were fixed, whereas in the work of Chapter 4, they were sampled from the posterior. When these parameters are sampled, there may be additional error if q systematically underestimates K. However, even with fixed K, the sampler implicitly fits the latent dimension by inferring the precision matrix $\mathbf{\Lambda_V}$. Therefore, systematic biases in the inferred dimension may already be accounted for by our analysis.

- **$q(\mathbf{V})$ and $p(\mathbf{V})$ are invariant to linear transormation.** (See Section 6.2.4.2). Instead of the diagonal covariance model from Chapter 3, we assumed a full covariance model for $\mathbf{V}$, and carefully chose a weakly informative prior for $\mathbf{\Lambda_V}$ so that $q(\mathbf{V})$ and $p(\mathbf{V})$ would be invariant to linear transformations. This was necessary for the analysis, because the low rank factorization has a rotational symmetry, and therefore the rotation of the latent space is not identifiable. Essentially, the priors on $\mathbf{V}$ were chosen to be insensitive to such rotations. We note that an invariance assumption such as this one is needed to rule out models such as $\mathsf{G}(\mathsf{M}\mathsf{G} + \mathsf{G}) + \mathsf{G}$ where the prior over $\mathbf{V}$ is sensitive to rotation.

- **Fully observed Y with spherical Gaussian noise.** We assumed that the model (but not necessarily the data) had spherical Gaussian noise, and that $\mathbf{Y}$ was fully observed. This was needed to express the posterior in terms of Kronecker products. We don't know if this assumption is required for CIS to perform well.

- **Within-cluster and between-cluster variance of V are scalar multiples.** As with the previous assumption, this one was needed in order to express the posterior in terms of Kronecker products.

- **Clustering model for V.** Our derivation used almost no structure specific to the clustering model. Section 6.2.4.1 reduced the problem to the case of

fixed assignments and variance parameters sampled from the posterior, so that $p(\mathbf{V} \mid \boldsymbol{\eta})$ was Gaussian. Therefore, the analysis works without modification for any prior over $\mathbf{V}$ which is Gaussian conditioned on its latent representation, such as $\mathsf{GB}^T + \mathsf{G}$ (if analogously modified to have full, rather than diagonal, covariance).

- q **is a low-rank factorization.** This assumption is somewhat more significant, in that we needed $\mathbf{U}$ and $\mathbf{V}$ to be continuous in order to approximate the posterior as Gaussian. Could we extend our analysis to models such as $\mathsf{M}(\mathsf{GG} + \mathsf{G}) + \mathsf{G}$? One step of our derivation (Appendix A.4.1) was to eliminate $\mathbf{U}$ by showing that $\mathbf{U}$ and $\mathbf{V}$ could be approximated as independent in the posterior; from this point on, the analysis involved only $\mathbf{V}$. Proving approximate independence could be a strategy for extending the analysis to other models.

Our analysis suggests the following thumb for designing model classes to be searched using the procedure of Chapter 4: the prior $p(\mathbf{V})$ must be insensitive to variation along any dimensions which are not identifiable under q.

6.3 Discussion

We have presented CIS, a fully compositional procedure for inference and marginal likelihood estimation in compositional models. We gave a theoretical analysis of the algorithm in the case of a clustering-within-low-rank model. Using a Gaussian approximation to the posterior, we derived a bound on the bias of the marginal likelihood estimates and on the KL divergence from the true posterior.

There is one obvious roadblock to proving that any algorithm can perform efficient inference in a compositional model: in most cases, even performing posterior inference in the simple models is NP-hard. While there are some recent results showing provably efficient algorithms for matrix factorization (Arora et al., 2012) and deep learning (Arora et al., 2013), these algorithms make strong assumptions about the process which generated the data and would not necessarily find the global optimum in practice. Assumptions about the data are especially problematic the context of compositional structure search, because the simpler models are believed to be wrong, and are fit only as a step towards fitting more complex models.

We adopted a different strategy for circumventing the NP-hardness barrier: we asked whether inference and model scoring for compositional models are any harder than inference and model scoring for the production rules. More precisely, we assumed *oracles* which compute exact posterior samples and marginal likelihoods for

each of the production rules. While there are no exact posterior inference algorithms for the production rules, there are approximate methods which work well in practice, and practical marginal likelihood estimators are plausibly within reach. Thus, CIS may turn out to be a practical algorithm.

While the analysis is limited to a small subset of the models in the grammar, the analysis should hopefully serve as a template for analyzing CIS applied to a wider variety of models. In general, this should inform a variety of broader questions, such as:

- When should we expect the greedy initialization approach of Chapter 4 to perform well?
- A common heuristic for model checking is to look at the latent variables and see if there is additional structure not captured by the model. When does this heuristic work?
- Why does layerwise training (Hinton et al., 2006) work well for learning deep representations?
- Which probabilistic inference problems are fundamental, and which can be performed compositionally in terms of simpler algorithms?
- How can one design a space of model structures which can be explored compositionally?

In the next chapter, we turn to algorithms for estimating marginal likelihood for the productions of the grammar.

Chapter 7

Evaluating marginal likelihood estimators

Chapter 6 introduced compositional importance sampling, a method for estimating marginal likelihood in compositional models which requires only marginal likelihood estimators corresponding to the productions in the grammar. Unfortunately, estimating marginal likelihood even for simple factorization models remains a difficult problem. This chapter and the next attempt to tackle the problem of marginal likelihood estimation in matrix factorizations and related models. This chapter introduces a novel technique for obtaining ground truth marginal likelihood values against which to evaluate estimators. Using this technique, we compare a wide variety of marginal likelihood estimators on three production rules from our grammar.

7.1 Motivation

A major obstacle to developing effective marginal likelihood (ML) estimators is that it is difficult even to know whether one's approximate ML estimates are accurate. The output of an ML estimator is a scalar value, and typically one has no independent way to judge the correctness of that value. Subtle implementation errors can lead to extremely inaccurate estimates with little indication that anything is amiss. Most estimators are based on sampling or optimization algorithms, and a failure to explore important modes of the posterior can also lead to highly inaccurate estimates. It is common for a particular algorithm to consistently under- or overestimate the true ML, even when the variance of the estimates appears to be small (e.g. Neal, 2008). Section 2.3.1 provides general background on the problems of ML estimation and partition function estimation more generally.

In this chapter, we present a framework for evaluating ML estimators for latent variable models. Our main contribution is a method for obtaining accurate ground truth estimates of ML for synthetic data generated from a model. This method can be used to test the correctness of implementations of ML estimators, measure their accuracy, and choose between different estimators for a given model. We use our method to rigorously compare ML estimators for several production rules of our grammar on realistic-sized datasets: a clustering model, a low rank approximation, and a binary attribute model. This highlights strengths and weaknesses of a variety of estimators, and should help inform the choice of ML estimators more generally. Since ML estimation is closely related to posterior inference, our approach also gives a novel quantitative criterion for comparing posterior inference algorithms.

7.1.1 Caveats

While the focus of this work is on the algorithmic issues involved in estimating ML, we should mention several caveats concerning the ML criterion itself. First, the very notion of a "correct" or "best" model may not be meaningful if none of the models under comparison accurately describe the data. In such cases, different models may better capture different aspects of the data, and the best choice is often application-dependent. This situation is especially relevant to the methods of this chapter, since the estimators are evaluated on synthetic data, and those results may not carry over if the model is a poor match to the data. For these reasons, ML should not be applied blindly, but should rather be used in conjunction with model checking methods such as posterior predictive checks (Gelman et al., 2014, chap. 6).

A common criticism of ML is that it is overly sensitive to the choice of hyperparameters, such as the prior variance of the model parameters (Kass, 1993; Kass and Raftery, 1995). Predictive criteria such as predictive likelihood and held-out error are insensitive to these hyperparameters for large datasets, because with enough data, the likelihood function will dominate the prior. However, a poor choice of hyperparameters can significantly affect the ML, even for arbitrarily large datasets. This can lead to a significant bias towards overly simple models, since the more parameters a model has, the stronger the effect of a poorly chosen prior. We note, however, that this is a problem with posterior inference over models, even if the ML is not computed explicitly. Techniques such as reversible jump MCMC (Green, 1995) and Bayesian nonparametrics (Ghahramani, 2012) suffer from precisely the same bias when the priors are misspecified.

There are several model selection criteria closely related to ML which alleviate the hyperparameter sensitivity. Berger and Pericchi (1996) proposed the *intrinsic Bayes*

factor, the probability of the data conditioned on a small number of data points. This can be equivalently viewed as computing the ratio of marginal likelihoods of different size datasets. Fractional Bayes factors (O'Hagan, 1995) have a similar form, but the denominator includes all of the data points, and each likelihood term is raised to a power less than 1. Another approach maximizes the ML with respect to the hyperparameters; this is known as empirical Bayes, the evidence approximation, or type-II maximum likelihood (MacKay, 1999). The motivation is that we can optimize over a small number of hyperparamters without overfitting too badly. Others suggest using ML, but designing the priors such that a poor choice of hyperparameters doesn't favor one model over another (Heckerman et al., 1995; Neal, 2001b). We note that all of these methods require computing high-dimensional integrals over model parameters and possibly latent variables much like the ones needed for ML. Therefore, they raise the same computational issues, and some of the algorithms discussed here can be directly used to estimate these other quantities as well.

For the remainder of the discussion, we will take as given that the goal is to estimate ML, and we will focus on the algorithmic issues.

7.2 Obtaining ground truth marginal likelihood

As mentioned above, evaluating ML estimators is difficult because the true value is not known. Sometimes, algorithms are tested on extremely small instances for which the ML can be computed exactly. This is useful for checking the mathematical correctness of an algorithm, but it gives little insight into how the algorithm performs on realistic dataset sizes. In particular, many failure modes of ML estimators, such as failure to mix, or large variance caused by importance sampling in high dimensions, simply would not arise for small instances.

As discussed in Section 2.3.1, unbiased estimators of the ML can be viewed as stochastic lower bounds in two senses:

1. By Jensen's inequality, they underestimate the true value on average: $\mathbb{E}[\log \hat{p}(\mathbf{y})] \leq \log p(\mathbf{y})$.

2. By Markov's inequality, they are unlikely to overestimate the true value by very much: $\Pr(\log \hat{p}(\mathbf{y}) > \log p(\mathbf{y}) + b) \leq e^{-b}$.

Many commonly used ML estimators, such as annealed importance sampling (AIS) and sequential Monte Carlo (SMC), are unbiased. One way to compare these estimators is to suppose that whichever one returns a larger value is closer to the truth. However, this doesn't answer the question of how close to the true value they are.

(A 1 nat difference is more significant if the estimates are close to the true value than if they are far from it.) Even if multiple estimators give similar estimates, this does not imply they are accurate, since multiple estimators can give the same wrong answer if they fail to explore the same modes.

Our method for obtaining ground truth is based on computing both stochastic upper bounds and stochastic lower bounds, such that the gap between the bounds is small. It is already understood how to achieve accurate lower bounds: AIS provably converges to the true value as the number of intermediate distributions is increased (Neal, 2001a). Finding good upper bounds is harder. (From a theoretical standpoint, we would expect it to be difficult to obtain good upper bounds, because these would act as certificates that there isn't any remaining unexplored mass in the posterior.)

As discussed in Section 2.3.4, the harmonic mean estimator is a stochastic upper bound when exact posterior samples are used. Unfortunately, there are two problems: first, if approximate samples are used, the estimator is not a stochastic upper bound, and in fact suffers from the same mixing issues as the other methods. Second, as we show in the experiments, even when exact samples are used, the bound is extremely poor.

For synthetic data, it is possible to work around both of these issues. For the issue of finding exact samples, observe that there are two different ways to sample from the joint distribution $p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{y})$ over parameters $\boldsymbol{\theta}$, latent variables $\mathbf{z}$, and observations $\mathbf{y}$. We can forward sample by first sampling $(\boldsymbol{\theta}, \mathbf{z})$ from $p(\boldsymbol{\theta}, \mathbf{z})$, and then sampling $\mathbf{y}$ from $p(\mathbf{y} \mid \boldsymbol{\theta}, \mathbf{z})$. Alternatively, we can first sample $\mathbf{y}$ from $p(\mathbf{y})$, and then sample $(\boldsymbol{\theta}, \mathbf{z})$ from the posterior $p(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{y})$. Since these two processes sample from the same joint distribution, the $(\boldsymbol{\theta}, \mathbf{z})$ generated during forward sampling is also an exact sample from the posterior $p(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{y})$. (The Geweke test (Geweke, 2004) is based on the same identity.) In other words, for a synthetic dataset, we have available a single exact sample from the posterior.[1]

The other problem with the harmonic mean estimator is that the bound is extremely poor even when one has exact samples. The reason is that it is an instance of simple importance sampling between dissimilar distributions in a high-dimensional space. Sections 2.3.5 and 2.3.6 discussed two ways to improve the performance of the likelihood weighting estimator by breaking it down into a series of smaller steps: annealed importance sampling (AIS) and sequential Monte Carlo (SMC). This suggests that a series of distributions bridging from the posterior to the prior could yield more accurate results. We now discuss two particular instantiations of this.

[1]More posterior samples can be obtained by running an MCMC algorithm starting from the original one. However, the statistics would likely be correlated with those of the original sample. We used a single exact sample for each of our experiments.

7.2.1 Reverse AIS

In Section 2.3.5, we discussed an interpretation of AIS as simple importance sampling over an extended state space, where the proposal and target distributions correspond to forward and backward annealing chains. We noted that that the reverse chain generally could not be sampled from explicitly because it required an exact sample from $p_T(\mathbf{x})$ – in this case, the posterior distribution. However, for synthetic data, the reverse chain can be run starting from an exact sample as described above. The importance weights for the forward chain using the backward chain as a proposal distribution are given by:

$$\frac{q_{for}(\mathbf{x}_1, \ldots, \mathbf{x}_T)}{q_{back}(\mathbf{x}_1, \ldots, \mathbf{x}_T)} = \frac{\mathcal{Z}_T}{\mathcal{Z}_1} w, \tag{7.1}$$

where

$$w \triangleq \frac{f_{T-1}(\mathbf{x}_{T-1})}{f_T(\mathbf{x}_{T-1})} \cdots \frac{f_1(\mathbf{x}_1)}{f_2(\mathbf{x}_1)}. \tag{7.2}$$

As in Section 2.3.5, because this ratio represents the importance weights between two normalized distributions, its expectation must be 1, and therefore $\mathbb{E}[w] = \mathcal{Z}_1/\mathcal{Z}_T$. Since p_1 is chosen to be the prior, $\mathcal{Z}_1 = 1$, and we obtain the following estimate of the ML:

$$\hat{p}_{back}(\mathbf{y}) = \frac{K}{\sum_{k=1}^{K} w^{(k)}}. \tag{7.3}$$

As discussed in Section 2.3.5, the standard AIS estimator (denoted here as $\hat{p}_{for}(\mathbf{y})$) is a stochastic lower bound, in that for any b, $\log \hat{p}_{for}(\mathbf{y}) \leq \log p(\mathbf{y}) + b$ with probability $1 - e^{-b}$. Similarly, because (7.3) is obtained from an unbiased estimate of $1/p(\mathbf{y})$, it is a stochastic upper bound on $p(\mathbf{y})$. Therefore, the true value is sandwiched between two bounds:

$$\log \hat{p}_{for}(\mathbf{y}) - b \leq \log p(\mathbf{y}) \leq \log \hat{p}_{back}(\mathbf{y}) + b \qquad \text{with probability } 1 - 2e^{-b}. \tag{7.4}$$

This is not an asymptotic result—it holds for *any* number of intermediate distributions. If we are fortunate enough to have forward and backward estimates which differ by a nats, then $\log p(\mathbf{y})$ is known to within $a + 2b$ nats with small probability of error. No additional work needs to be done to determine the variance of the estimators, or whether AIS has sufficiently explored the posterior.

In order for the method to be useful, it is also necessary that the gap between

the lower and upper estimates be small. As discussed in Section 2.3.5, the bias and variance of $\log \hat{p}(\mathbf{y})$ both approach zero as the number of intermediate distributions tends to infinity. The proof holds in the reverse direction as well. Therefore, if the forward and reverse estimators are run with large numbers of intermediate distributions, they will approach the same value from both sides, and the variance can be made arbitrarily small. There is no theoretical guarantee that a small gap will be achieved for any reasonable number of intermediate distributions, but our experiments demonstrate the practicality of the technique on several different model classes.

7.2.2 Sequential harmonic mean estimator

As discussed in Section 2.3.6.1, SMC with a single particle can be analyzed as a special case of AIS. Therefore, starting from an exact posterior sample, we can run the reverse chain for SMC as well. The resulting algorithm, which we call the sequential harmonic mean estimator (SHME), corresponds to starting with full observations and the exact posterior sample, and deleting one observation at a time. Each time an observation is deleted, the weights are updated with the likelihood of the observations, exactly as in SMC. The difference is in how the weights are used: when the resampling criterion is met, the particles are sampled proportionally to the *reciprocal* of their weights. Also, while Algorithm 2 computed arithmetic means of the weights in the resampling step and in the final partition function estimate, SHME uses the harmonic means of the weights.

As in SMC, we leave open the choice of proposal distribution $q(\mathbf{z}_i \mid \mathbf{y}_i, \boldsymbol{\theta}^{(k)})$. Possibilities include (2.27) and (2.29) of Section 2.3.6. Unlike in standard SMC, the proposal distribution does not affect the states sampled in the algorithm—rather, it is used to update the weights. Proposal distributions which better reflect the posterior are likely to result in lower variance weights.

The harmonic mean estimator has been criticized for its instability (Neal, 2008), so it is worth considering whether the same issues are relevant to SHME. One problem with the harmonic mean estimator is that it is equivalent to simple importance sampling where the target distribution is more spread out than the proposal distribution. Therefore, samples corresponding to the tails of the distribution have extremely large importance weights. The same problem is present here (though to a lesser degree) because the posterior $p(\mathbf{z}_{1:i}, \boldsymbol{\theta} \mid \mathbf{y}_{1:i})$ is slightly more peaked than the posterior given one less data point, $p(\mathbf{z}_{1:i}, \boldsymbol{\theta} \mid \mathbf{y}_{1:i-1})$. Therefore, we do not recommend using this procedure to compute a particle approximation to the posterior.

However, despite this problem, the algorithm can still yield accurate estimates

Algorithm 3 Sequential harmonic mean estimator

for $k = 1$ to K **do**
  $(\mathbf{z}^{(k)}, \boldsymbol{\theta}^{(k)}) \leftarrow$ exact sample from $p(\mathbf{z}, \boldsymbol{\theta} \,|\, \mathbf{y})$
  $w^{(k)} \leftarrow 1$
end for
for $i = T$ to 1 **do**
  for $k = 1$ to K **do**
    $(\mathbf{z}_{1:i}^{(k)}, \boldsymbol{\theta}^{(k)}) \leftarrow$ MCMC transition which leaves $p(\mathbf{z}_{1:i}, \boldsymbol{\theta} \,|\, \mathbf{y}_{1:i})$ invariant
    $w^{(k)} \leftarrow w^{(k)} p(\mathbf{z}_i^{(k)} \,|\, \boldsymbol{\theta}) p(\mathbf{y}_i \,|\, \mathbf{z}_i^{(k)}, \boldsymbol{\theta}^{(k)}) / q(\mathbf{z}_i \,|\, \mathbf{y}_i, \boldsymbol{\theta}^{(k)})$
  end for
  if resampling criterion met **then**
    Resample $(\mathbf{z}_{1:i}^{(k)}, \boldsymbol{\theta}^{(k)})$ proportionally to $1/w^{(k)}$
    $S \leftarrow \sum_{k=1}^{K} 1/w^{(k)}$
    for $k = 1$ to K **do**
      $w^{(k)} \leftarrow K/S$
    end for
  end if
end for
return $\hat{\mathcal{Z}} = \frac{K}{\sum_{k=1}^{K} 1/w^{(k)}}$

of the ML. To justify this theoretically, suppose we have a model for which $\boldsymbol{\theta}$ and $\mathbf{z}_i^{(k)}$ can both be integrated out analytically. (This is the case for the clustering model.) We analyze the gap between the SMC (lower bound) and SHME (upper bound) estimates. For simplicity, assume that only a single particle is used in each algorithm, and that the MCMC transition operator yields perfect samples. In this case, the expected SMC estimate of $\log \hat{p}(\mathbf{y})$ is given by:

$$\mathbb{E}[\log \hat{p}(\mathbf{y})] = \sum_{i=1}^{N} \mathbb{E}_{p(\mathbf{z}_{1:i-1} \,|\, \mathbf{y}_{1:i-1})} \left[\log p(\mathbf{y}_i \,|\, \mathbf{z}_{1:i-1}, \mathbf{y}_{1:i-1}) \right]. \tag{7.5}$$

On the other hand, the SHME estimate is given by:

$$\mathbb{E}[\log \hat{p}(\mathbf{y})] = \sum_{i=1}^{N} \mathbb{E}_{p(\mathbf{z}_{1:i-1} \,|\, \mathbf{y}_{1:i})} \left[\log p(\mathbf{y}_i \,|\, \mathbf{z}_{1:i-1}, \mathbf{y}_{1:i-1}) \right]. \tag{7.6}$$

The only difference between these two expressions is that the posterior expectation in (7.6) conditions on one more data point than that of (7.5). Intuitively, in the context of the clustering model, seeing an additional data point may influence how

the remaining data points are clustered. If the degree of influence is small, then the upper and lower bounds are close to each other, and therefore close to $\log p(\mathbf{y})$.

7.2.3 Relationship with inference

As discussed in Section 2.3.7, the problems of inference and ML estimation are equivalent for variational Bayes: the KL divergence from the posterior equals the gap between the variational lower bound and the true ML. While we are not aware of any general statement of this form for sampling-based methods, similar relationships hold for particular estimators. In particular, the same identity holds in expectation for the SIS estimator of Section 2.3.3. If $q(\mathbf{z}, \boldsymbol{\theta})$ is the proposal distribution, then the expectation of (2.15) with a single sample is given by:

$$\begin{aligned}\mathbb{E}_{q(\mathbf{z},\boldsymbol{\theta})}\left[\log p(\mathbf{z}, \boldsymbol{\theta}, \mathbf{y}) - \log q(\mathbf{z}, \boldsymbol{\theta})\right] &= \log p(\mathbf{y}) + \mathbb{E}_{q(\mathbf{z},\boldsymbol{\theta})}\left[\log p(\mathbf{z}, \boldsymbol{\theta} \,|\, \mathbf{y}) - \log q(\mathbf{z}, \boldsymbol{\theta})\right] \\ &= \log p(\mathbf{y}) - \mathrm{D}_{\mathrm{KL}}(q(\mathbf{z}, \boldsymbol{\theta}) \,\|\, p(\mathbf{z}, \boldsymbol{\theta} \,|\, \mathbf{y})). \end{aligned} \tag{7.7}$$

Observe that this expectation is exactly the same as the variational lower bound (2.39) when q is used as the variational approximation. This highlights a surprising equivalence between variational and sampling-based ML estimators which makes it possible to compare both on equal footing.

Furthermore, we have discussed two sampling-based estimators which can be seen as importance sampling on an extended state space: AIS (Section 2.3.5) and SMC with a single particle (Section 2.3.6.1). Let $\mathbf{v}$ denote all of the variables sampled in one of these algorithms other than $\mathbf{z}$ and $\boldsymbol{\theta}$. (For instance, in AIS, it denotes all of the states other than the final one.) In this case, the above derivation can be modified:

$$\begin{aligned}\mathbb{E}[\log \hat{p}(\mathbf{y})] &= \mathbb{E}_{q(\mathbf{z},\boldsymbol{\theta},\mathbf{v})}\left[\log p(\mathbf{z}, \boldsymbol{\theta}, \mathbf{v}, \mathbf{y}) - \log q(\mathbf{z}, \boldsymbol{\theta}, \mathbf{v})\right] \\ &= \log p(\mathbf{y}) - \mathrm{D}_{\mathrm{KL}}(q(\mathbf{z}, \boldsymbol{\theta}, \mathbf{v}) \,\|\, p(\mathbf{z}, \boldsymbol{\theta}, \mathbf{v} \,|\, \mathbf{y})) \\ &\leq \log p(\mathbf{y}) - \mathrm{D}_{\mathrm{KL}}(q(\mathbf{z}, \boldsymbol{\theta}) \,\|\, p(\mathbf{z}, \boldsymbol{\theta} \,|\, \mathbf{y})). \end{aligned} \tag{7.8}$$

This implies that the KL divergence from the true posterior is bounded by the bias of the estimator. If either of these algorithms returns an accurate estimate of the ML, it must also yield a good posterior sample. While the ML bounds can be used to determine which of two inference algorithms is more accurate, it is hard to measure the significance of the gap unless $p(\mathbf{y})$ is known. (If algorithm A has a KL divergence of 11 from the true posterior and algorithm B has a KL divergence of 1, this is a large difference. If A has a KL divergence of 1010 and B has a KL divergence of

1000, this has less practical relevance.)

However, in conjunction with the algorithms of this section, the above inequalities can be used to measure the accuracy of posterior samples on synthetic datasets. In particular, if we use an importance sampling based estimator to lower bound $p(\mathbf{y})$ and a (possibly different) algorithm to upper bound it, the gap between the estimates serves as a stochastic bound on the KL divergence from the true posterior. Therefore, for those algorithms which are based on importance sampling, our experiments comparing ML estimates can also be seen as evaluations of posterior inference.

7.3 Experiments

In this section, we compare several ML estimation algorithms on simple matrix factorization models: clustering ($\mathsf{MG}+\mathsf{G}$), low rank approximation ($\mathsf{GG}+\mathsf{G}$), and binary attributes ($\mathsf{BG}+\mathsf{G}$). The following estimators are compared:

- the Bayesian information criterion (BIC)
- likelihood weighting (Section 2.3.3)
- the harmonic mean estimator (HME) (Section 2.3.4), using a Markov chain starting from the exact sample
- annealed importance sampling (AIS) (Section 2.3.5)
- sequential Monte Carlo (SMC), using a single particle (Section 2.3.6)
- variational Bayes (Section 2.3.7). We report results both with and without the symmetry coorection, where the ML lower bound is multiplied by the number of equivalent relabelings ($K!$ for all models we consider).
- the Chib-Murray-Salakhutdinov (CMS) estimator (Section 2.3.8)
- nested sampling (Skilling, 2006)

7.3.1 Implementation

In order to be fair to all algorithms, the implementations share the same MCMC transition operators wherever possible. The implementations are not especially optimized, and therefore all of the algorithms could likely be sped up considerably with careful tuning. However, because the MCMC transition operators make up most

of the running time and are shared between most of the algorithms, the running times are directly comparable. The only exceptions are variational Bayes and nested sampling, whose update rules are not shared with the other algorithms.

All of the estimators except for BIC, likelihood weighting, and variational Bayes require an MCMC transition operator which preserves the posterior distribution. In addition, some of the algorithms required implementing some additional computations:

- AIS requires MCMC operators for each of the intermediate distributions.
- SMC requires the ability to compute or approximate the likelihood of a data point under the predictive distribution.
- The CMS estimator requires implementing the reverse transition operators and requires the ability to compute the probability of transitioning between any given pair of states. The latter is a nontrivial requirement, in that it disallows operators which compute auxiliary variables.
- Nested sampling requires an MCMC operator for the prior subject to the constraint on the likelihood.
- Variational Bayes is an optimization, rather than sampling, algorithm. In our implementation, the updates all involved optimizing one of the component distributions given the others.

The operators implemented for the specific models are as follows:

- **Clustering.** The transition operator was collapsed Gibbs sampling (see Section 2.2.2). The cluster centers were collapsed out wherever possible in all computations. The predictive likelihood can be computed exactly given the cluster assignments and variance parameters, with the cluster centers collapsed out.
- **Low rank.** The transition operator was block Gibbs sampling (see Section 2.2.1). For computing predictive likelihood, the right factor was sampled from the posterior, and the left factor was marginalized out analytically.
- **Binary attributes.** The transition operator was collapsed Gibbs sampling. For all computations, the feature matrix was collapsed out analytically wherever possible, and the tricks of Doshi-Velez and Ghahramani (2009) (discussed in Section 2.2.2) were used to update the posterior efficiently.

In general, we face a tradeoff between performance and difficulty of implementation. Therefore, it is worth discussing the relative difficulty of implementation of different estimators. In general, BIC, likelihood weighting, and the harmonic mean estimator required almost no work to implement beyond the MCMC sampler. Of the sampling based estimators, AIS and nested sampling required the most work to implement, because they each required implementing a full set of MCMC transition operators specific to those algorithms.[2] SMC and the CMS estimator were in between: they required only a handful of additional functions beyond the basic MCMC operators.

Compared with the sampling methods, variational Bayes typically requires somewhat more math to derive the update rules. However, it is considerably simpler to test (see Section 7.3.2). For the low rank and clustering models, implementing variational Bayes required a comparable amount of effort to implementing the MCMC transitions. For the binary attribute model, variational Bayes was considerably easier to implement than the efficient collapsed sampler.

7.3.2 Testing

ML estimators are notoriously difficult to implement correctly. The problem is that an ML estimator returns a scalar value, and there is little obvious indication if the value is wrong. Furthermore, buggy MCMC transition operators often yield seemingly plausible posterior samples, yet lead to bogus ML estimates when used in an algorithm such as AIS. For these reasons, it is worth discussing methods for testing mathematical correctness of ML estimator implementations. We used several strategies, which we recommend following in any work involving ML estimation:

1. Most of the MCMC operators were implemented in terms of functions which returned conditional probability distributions. (The distributions were classes which knew how to sample from themeslves and evaluate their density functions.) The conditional probability distributions can be "unit tested" by checking that they are consistent with the joint probability distribution. In particular,

$$\frac{p(x \mid u)}{p(x' \mid u)} = \frac{p(x, u)}{p(x', u)}$$

[2] AIS is most often used in the undirected setting, where the transition operators for the model itself are easily converted to transition operators for the intermediate distributions. In the directed setting, however, raising the likelihood to a power can destroy the directed structure, and therefore implementing collapsed samplers can be more involved.

must hold for *any* triple (x, x', u). This form of unit testing is preferable to simulation-based tests, because the identity must hold exactly, and fails with high probability if the functions computing conditional probabilities are incorrect.

Analogously, the updates for variational Bayes were tested by checking that they returned local maxima of the variational lower bound.

2. To test the MCMC algorithms themselves, we used the Geweke test (Geweke, 2004). This can be thought of as an "integration test," since it checks that all the components of the sampler are working together correctly. This test is based on the fact that there are two different ways of sampling from the joint distribution over parameters $\boldsymbol{\theta}$, latent variables $\mathbf{z}$, and data $\mathbf{y}$. First, one can sample forwards from the model. Second, one can begin with a forwards sample and alternate between (a) applying the MCMC transition operator, which preserves the posterior $p(\boldsymbol{\theta}, \mathbf{z} \,|\, \mathbf{y})$, and (b) resampling $\mathbf{y}$ from $p(\mathbf{y} \,|\, \boldsymbol{\theta}, \mathbf{z})$. If the implementation is correct, these two procedures should yield samples from *exactly* the same distribution. One can check this by checking P-P plots of various statistics of the data.

   The Geweke test is considered the gold standard for testing MCMC algorithms. It can detect surprisingly subtle bugs, because the process of resampling the data tends to amplify small biases in the sampler. (E.g., if the MCMC operator slightly overestimates the noise, the data will be regenerated with a larger noise, and the bias will be amplified over many iterations.) The drawback of the Geweke test is that it gives no indication of where the bug is. Therefore, it is recommended that one run it only after all of the unit tests pass.

3. The ML estimators were tested on toy distributions, where the ML could be computed analytically, and on very small instances of the clustering and binary models, where it could be computed through brute force enumeration of all latent variable configurations.

4. Because we had implemented a variety of ML estimators, we could check that they agreed with each other on at least some easy problem instances, such as those with extremely small or large single-to-noise ratios (SNR), or with small numbers of data points. (Because steps 1 and 2 strenuously test the parts of the implementation that are shared between different estimators, any further bugs will hopefully cause the estimates returned by different estimators to differ.)

The vast majority of bugs that we caught were caught in step 1, only a handful in steps 2 and 3, and none in step 4. We would recommend using 1, 2, and 3 for any work

which depends on ML estimation. (Steps 1 and 3 are applicable to partition function estimation more generally, while the Geweke test is specific to directed models.) Step 4 may be overkill for most applications because it requires implementing multiple estimators, but it provides an additional degree of reassurance in the correctness of the implementation.

The techniques of this section test only the *mathematical correctness* of the implementation, and do not guarantee that the algorithm returns an accurate answer. The algorithm may still return inaccurate results because the MCMC sampler fails to mix or because of statistical variability in the estimator. These are the factors that the experiments of this chapter are intended to measure.

7.3.3 Algorithm parameters

Each of the ML estimators provides one or more knobs which control the tradeoff between accuracy and computation time. In order to investigate the accuracy as a function of running time, we varied one knob for each algorithm and set the rest to reasonable defaults. The following parameters were varied for each algorithm:

- **Likelihood weighting** and **harmonic mean:** The independent variable was the number of proposals.
- **Annealed importance sampling:** The annealing path consisted of geometric averages of the initial and target distributions. Because AIS is sometimes unstable near the endpoints of a linear path, we used the following sigmoidal schedule which allocates more intermediate distributions near the endpoints:

$$\tilde{\beta}_t = \sigma\left(\delta\left(\frac{2t}{T} - 1\right)\right)$$
$$\beta_t = \frac{\tilde{\beta}_t - \tilde{\beta}_1}{\tilde{\beta}_T - \tilde{\beta}_1},$$

  where σ denotes the logistic sigmoid function and δ is a free parameter. (We used $\delta = 4$.) In our experiments, the independent variable was T, the number of intermediate distributions.
- **Sequential Monte Carlo:** We used only a single particle in all experiments, and the independent variable was the number of MCMC transitions per data point.

- **Chib-Murray-Salakhutdinov:** We used a single sample $(\boldsymbol{\theta}^*, \mathbf{z}^*)$ and varied the number of MCMC transitions starting from that sample.

- **Variational Bayes:** The independent variable was the number of random restarts in the optimization procedure. Specifically, in each attempt, optimization was continued until the objective function improved by less than 0.01 nats over 50 iterations, at which point another random restart was done. The highest value obtained over all random restarts was reported.

- **Nested sampling:** The algorithm has three parameters: the number of steps, the number of particles, and the number of MCMC transitions per step. The number of steps was chosen automatically by stopping when the (multiplicative) likelihood updates dropped below $1 + e^{-10}$. We found that using only 2 particles (the smallest number for which the algorithm is defined) consistently gave the most accurate results for modest computation time. Therefore, the independent variable was the number of MCMC transitions per step.

When applying algorithms such as AIS or SMC, it is common to average the estimates over multiple samples, rather than using a single sample. For this set of experiments, we ran 25 independent trials of each estimator. We report two sets of results: the average estimates using only a single sample, and the estimates which combine all of the samples.[3] As discussed in Section 7.3.5, there was little qualitative difference between the two conditions.

7.3.4 How much accuracy is required?

What level of accuracy do we require from an ML estimator? At the very least, we would like the errors in the estimates to be small enough to detect "substantial" log-ML differences between alternative models. Kass and Raftery (1995) offered the following table to summarize significance levels of ML ratios:

[3]For algorithms which are unbiased estimators of the marginal likelihood (AIS, SMC, CMS, and likelihood weighting), the arithmetic mean of the individual estimates was taken. For algorithms which are unbiased estimators of the reciprocal (harmonic mean, reverse AIS, SHME), the harmonic mean was used. For variational inference, the max over all trials was used. For nested sampling, the average of the log-ML estimates was used.

$\log_{10} p_1(\mathbf{y}) - \log_{10} p_2(\mathbf{y})$	$p_1(\mathbf{y})/p_2(\mathbf{y})$	Strength of evidence against p_2
0 to 1/2	1 to 3.2	Not worth more than a bare mention
1/2 to 1	3.2 to 10	Substantial
1 to 2	10 to 100	Strong
> 2	> 100	Decisive

This table serves as a reference point if one believes one of the models is precisely correct. However, in most cases, all models under consideration are merely simplifications of reality.

Concretely, suppose we have a dataset consisting of $N = 1000$ data points, and we are considering two models, $\mathcal{M}_1$ and $\mathcal{M}_2$. If $\mathcal{M}_1$ achieves an average predictive likelihood score which is 0.1 nats per data point higher than that of $\mathcal{M}_2$, this translates into a log-ML difference of 100 nats. Interpreted as a log-odds ratio, this would be considered overwhelming evidence. However, the difference in predictive likelihood is rather small, and in practice may be outweighed by other factors such as computation time and interpretability. Roughly speaking, 1 nat is considered a large difference in predictive likelihood, while 0.1 nats is considered small. Therefore, we may stipulate that an ML estimation error of $0.1N$ nats is acceptable, while one of N nats is not.

Alternatively, we can judge the errors against the resolution necessary to compare different model classes. For instance, if one is comparing the flat clustering model $\mathsf{MG} + \mathsf{G}$ against the structureless model G, the former should yield better predictions about every entry of the $N \times D$ observation matrix. Therefore, its log-ML score should be higher by $O(ND)$ nats. Similarly, if one is comparing $\mathsf{MG} + \mathsf{G}$ and the co-clustering model $\mathsf{M}(\mathsf{GM}^T + \mathsf{G}) + \mathsf{G}$, the latter would give a better prior for the $K \times D$ matrix of cluster centers, so it should improve the log-ML by $O(KD)$ nats. A model which finds additional structure in the parameters for each block would improve the log-ML by a further $O(KK')$ nats, where K and K' are the number of row and column clusters. Thus, in a sense, the accuracy of an ML estimator determines the depth of the structures it is able to distinguish.

Finally, there is an empirical yardstick we can use, namely comparing the ML scores of different models fit to the same datasets. Table 7.1 shows the ML estimates for all three models under consideration, on all three of the synthetic datasets. The estimates were obtained from AIS with 30,000 intermediate distributions.[4] These

[4]We are guaranteed accurate results only for the diagonal entries, for which upper bounds were available. However, AIS with 30,000 intermediate distributions yielded accurate estimates in all comparisons against ground truth (see Section 7.3.5).

	Clustering		Low rank		Binary	
Clustering	-2377.5		-2390.6	(13.1)	-2383.2	(5.7)
Low rank	-2214.2	(69.1)	-2145.1		-2171.4	(26.3)
Binary	-2268.6	(49.2)	-2241.7	(22.3)	-2219.4	

Table 7.1: Marginal likelihood scores for all three models evaluated on synthetic data drawn from all three models. **Rows:** the model used to generate the synthetic data. **Columns:** the model fit to the data. Each entry gives the marginal likelihood score estimated using AIS, and (in parentheses) the log-ML difference from the correct model.

numbers suggest that, for a dataset with 50 data points and 25 dimensions, the ML estimators need to be accurate to within tens of nats to distinguish different Level 1 factorization models.

7.3.5 Results

All of the ML estimation algorithms were run on all three of the models. A synthetic dataset was generated for each model with 50 data points and 25 input dimensions. There were 10 latent components for the clustering and binary models and 5 for the low rank model. In all cases, the "ground truth" estimate was obtained by averaging the log-ML estimates of the forward and reverse AIS chains with the largest number of intermediate distributions. In all cases, the two estimates agreed to within 1 nat. Therefore, by the analysis of Section 2.3.1, the ground truth value is accurate to within a few nats with high probability.

As mentioned in Section 7.3.3, each algorithm was run independently 25 times, and the results are reported both for the individual trials and for the combined estimates using all 25 trials. We plot the average log-ML estimates as a function of running time in order to visualize the bias of each estimator. In addition, we plot the root mean squared error (RMSE) values as a function of running time. We do not report MSE values for the AIS runs with the largest number of intermediate distributions because the estimates were used to compute the ground truth value.

Section 7.3.4 argued, from various perspectives, that the log-ML estimates need to be accurate on the order of 10 nats to distinguish different model classes. Therefore, for all models, we report which algorithms achieved RMSE of less than 10 nats, and how much time they required to do so.

Clustering. The results for the clustering model $\mathsf{MG}+\mathsf{G}$ are shown in Figures 7-1 and 7-2. Figure 7-1 shows the log-ML estimates for all estimators, while Figure 7-2

shows the RMSE of the log-ML estimates compared to the ground truth. Of the algorithms which do not require an exact posterior sample, only three achieved this level of accuracy: AIS, SMC, and nested sampling (NS). SMC gave accurate results the fastest, achieving an RMSE of 4.6 nats in only 9.7 seconds. By comparison, AIS took 37.8 seconds for an RMSE of 7.0 nats, and NS took 51.2 seconds for an RMSE of 5.7 nats.

We are not aware of any mathematical results concerning whether NS is an upper or lower bound on the log-ML. Our results suggest that it tends to underestimate the log-ML, similarly to the other algorithms. However, it significantly overestimated the log-ML on many individual runs, suggesting that it is not truly a stochastic lower bound.

The most naive ML estimator, likelihood weighting (LW), vastly underestimated the true value. Its mirror image, the harmonic mean estimator (HME), vastly overestimated it. The Bayesian information criterion (BIC) gave by far the least accurate estimate, with MSE dwarfing even that of LW. This is remarkable, since the BIC requires fitting the model, while LW is simply a form of random guessing. This suggests that the BIC should be treated cautiously on small datasets, despite its asymptotic guarantees. The CMS estimator was more accurate than LW and HME, but still far from the true value. The MSE values for LW, HME, and CMS were nearly constant over at least 2 orders of magnitude in running time, suggesting that they cannot be made more accurate simply by running them longer.

A single run of the variational Bayes optimization took only 0.1 seconds, after which it returned a log-ML lower bound which was better than LW achieved after many samples. However, even after many random restarts, the best lower bound it achieved was still 15 nats below the true value, even with the symmetry correction. According to our earlier analysis, this suggests that it would not be accurate enough to distinguish different models. In order to determine if the gap was due to local optima, we ran the optimization starting from a point estimate on the sample which generated the data. In this experiment (and for the other two models as well), VB with random initializations was able to find the same optimum, or a slightly better one, suggesting that the gap is due to an inherent limit in the approximation rather than to local optima.

For parameter settings where individual samples of AIS and SMC gave results accurate to within 10 nats, combining the 25 samples gave quantitatively more accurate estimates. However, for all of the other algorithms and parameter settings, combining 25 trials made little difference to the overall accuracy, suggesting that an inaccurate estimator cannot be made into an accurate one simply by using more samples. (The same effect was observed for the other two models.) Roughly speak-

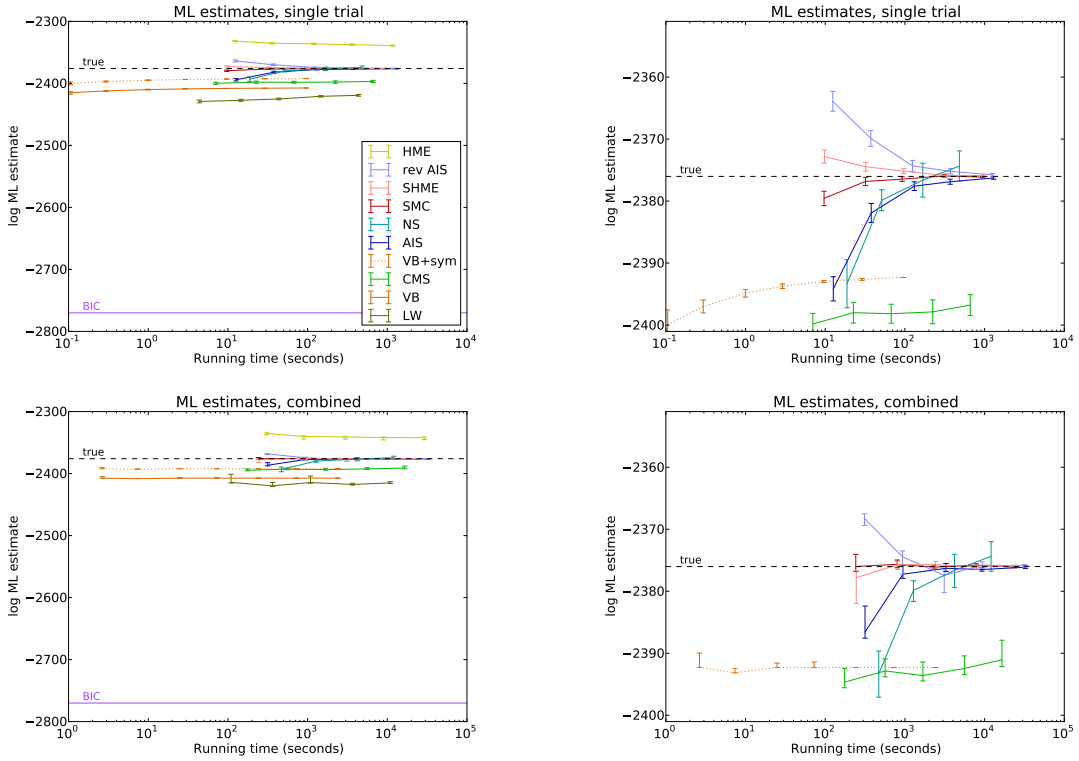


Figure 7-1: Comparison of marginal likelihood estimators on the clustering model $\mathsf{MG}+\mathsf{G}$. **Top:** average log-ML estimates for each of the 25 individual trials. (The right-hand figure is zoomed in.) **Bottom:** average log-ML estimates combined between the 25 trials. (The right-hand figure is zoomed in.) Note that there is little qualitative difference from the individual trials. **HME** = harmonic mean estimator. **rev AIS** = reverse AIS. **SHME** = sequential harmonic mean estimator. **SMC** = sequential Monte Carlo. **NS** = nested sampling. **AIS** = annealed importance sampling. **VB+sym** = variational Bayes with symmetry correction. **CMS** = Chib-Murray-Salakhutdinov estimator. **VB** = variational Bayes. **LW** = likelihood weighting.

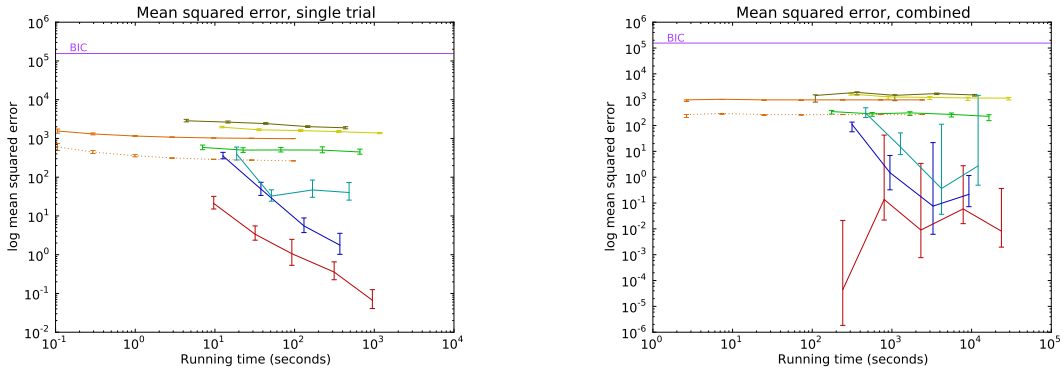


Figure 7-2: Mean squared error relative to ground truth for individual trials (left) and combined estimates (right) for the clustering model MG+G. See Figure 7-1 for the abbreviation key.

ing, it appears that one would rather spend a given amount of computation time to compute a handful of accurate estimates rather than a large number of sloppy ones. (Note that this is not true for all partition function estimation problems; for instance, in some of the experiments of Chapter 8, high-accuracy results were often obtained by averaging over many AIS runs, while some individual runs were inaccurate.)

Low rank. The results on the low rank factorization GG + G overwhelmingly favor AIS: its accuracy after only 1.6 seconds (RMSE = 8.6) matched or surpassed all other algorithms with up to 20 minutes of running time. In fact, AIS was the only algorithm to achieve an RMSE of less than 10.

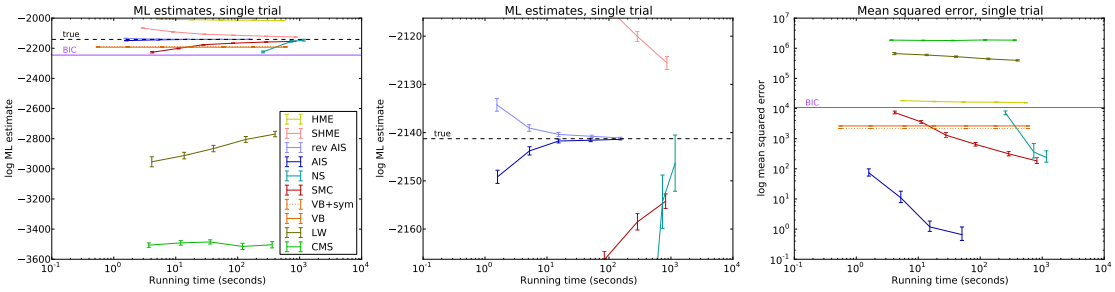


Figure 7-3: Comparison of marginal likelihood estimators on the low rank model GG + G. **Left:** average log-ML estimates across the 25 trials. **Middle:** same as left, but zoomed in. **Right:** average MSE of individual samples. See Figure 7-1 for the abbreviation key.

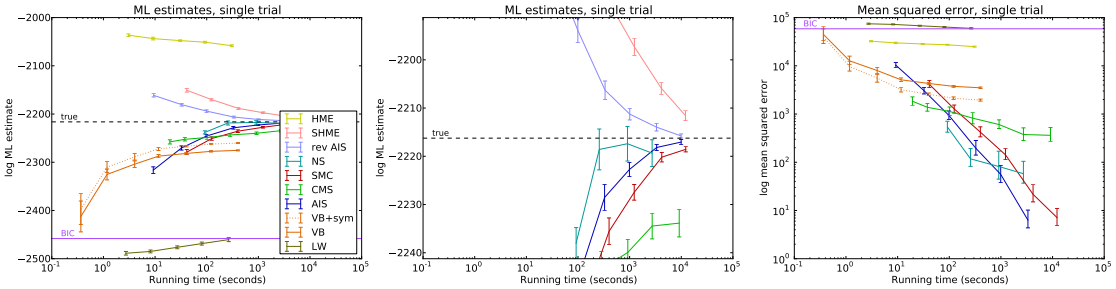


Figure 7-4: Comparison of marginal likelihood estimators on the binary attribute model BG + G. **Left:** average log-ML estimates across the 25 trials. **Middle:** same as left, but zoomed in. **Right:** average MSE of individual samples. See Figure 7-1 for the abbreviation key.

One reason that NS did not perform as well on this model as it did on the clustering model is that it took more steps to reach the region with high posterior mass. *E.g.*, with 5 MCMC transitions per step, it required 904 steps, as compared with 208 for clustering and 404 for binary. Another reason is that the MCMC implementation could not take advantage of the same structure which allowed block Gibbs sampling for the remaining algorithms; instead, one variable was resampled at a time from its conditional distribution. (For the clustering and binary models, the NS transition operators were similar to the ones used by the other algorithms.)

The CMS estimator underestimated the true value by over 1000 nats. The reason is that $p(\mathbf{U}^\star, \mathbf{V}^\star \mid \mathbf{Y})$ is estimated using an MCMC chain starting close to the point estimate $p(\mathbf{U}^\star, \mathbf{V}^\star)$. The model has a large space of symmetries which the Markov chain explores slowly, because $\mathbf{U}$ and $\mathbf{V}$ are tightly coupled. Therefore, the first few samples dramatically overestimate the probability of transitioning to $(\mathbf{U}^\star, \mathbf{V}^\star)$, and it is impossible for later samples to cancel out this bias because the transition probabilities are averaged arithmetically. In general, variational Bayes is also known to have difficulty modeling spaces with symmetries when tightly coupled variables are restricted to be independent in the variational approximation. Interestingly, it was able to attenuate the effect by making $\mathbf{U}$ and $\mathbf{V}$ small in magnitude, thereby reducing the coupling between them.

Binary attributes. Finally, the results for the binary attribute model BG + G are shown in Figure 7-4. Similarly to the clustering model, three algorithms came within 10 nats of the true value: NS, AIS, and SMC. NS and AIS each crossed the 10 nat threshold in similar amounts of time: AIS achieved an RMSE of 7.5 nats in 16.5 minutes, while NS achieved an RMSE of 9.0 nats in 15.1 minutes. By contrast, SMC

achieved RMSE values of 11.9 and 4.7 in 20.5 minutes and 69 minutes, respectively. AIS and SMC continued to give more accurate results with increased computation time, while the accuracy of NS was hindered by the variance of the estimator. Overall, this experiment suggests that estimating the ML of a binary attribute model remains a difficult problem. 15 minutes is a very long time for a dataset with only 50 data points and 25 input dimensions.

7.4 Discussion

Based on our experiments, we observe that the relative performance of different ML estimators varied tremendously depending on the model. More work is required to understand which algorithms perform well on which models and why. Our overall recommendation is that AIS should be the first algorithm to try for a given model, because in all of our experiments, it achieved accurate results given enough intermediate distributions. If AIS is too slow, then SMC and NS are also worth considering.

Interestingly, of the three strongest performing algorithms in our experiments—AIS, SMC, and NS—both AIS and SMC are instances of bridging between a tractable distribution and an intractable one using a sequence of intermediate distributions (see Section 2.3.6.1). Any algorithm which shares this structure can be reversed using the technique of Section 7.2 to obtain a stochastic upper bound on the log-ML of synthetic data. Therefore, if better algorithms are devised which build upon AIS and SMC (either separately or in combination), they will automatically lead to more precise ground truth ML values for synthetic data. We believe the rigorous quantitative evaluation framework we presented will accelerate progress in developing ML estimation algorithms for latent variable models.

While NS does not share the same mathematical structure as AIS and SMC, it also involves a sequence of distributions, albeit ones which are chosen adaptively. In fact, Skilling (2006) considered it similar enough to AIS that he proposed using NS to find better AIS schedules. In short, all three of the algorithms which performed well in our experiments are based on some form of annealing. This suggests that a better understanding of annealing in general is crucial to improving ML estimators. The next chapter presents a framework for analyzing and designing annealing paths.

Chapter 8

Alternative annealing paths

The previous chapter compared a variety of marginal likelihood estimators and found that annealed importance sampling (AIS) gives accurate estimates across a variety of model classes. In those experiments, the intermediate distributions consisted of geometric averages of the prior and posterior, which had a natural interpretation in terms of fractional data points. In many cases, however, the choice of intermediate distributions is less obvious. In this chapter, we present a framework for designing and analyzing annealing paths. We focus on the problem of estimating partition functions of Markov random fields (MRFs), because this case leads to a particularly insightful mathematical analysis, and because it is useful in its own right. While the algorithms are specific to MRFs and exponential families, we believe many of the insights uncovered by the analysis should extend to the directed case.

MRFs are defined in terms of an unnormalized probability distribution, and computing the probability of a state requires computing the (usually intractable) partition function. This is problematic for model selection, since one often wishes to compute the probability assigned to held-out test data. AIS is especially widely used because given enough computing time, it can provide high-accuracy estimates. AIS has enabled precise quantitative comparisons of powerful generative models in image statistics (Sohl-Dickstein and Culpepper, 2012; Theis et al., 2011) and deep learning (Salakhutdinov and Murray, 2008; Desjardins et al., 2011; Taylor and Hinton, 2009). Unfortunately, applying AIS in practice can be computationally expensive and require laborious hand-tuning of annealing schedules. Because of this, many generative models still have not been quantitatively compared in terms of held-out likelihood.

While there has been much work on automatically tuning AIS schedules (Neal, 1996; Behrens et al., 2012; Calderhead and Girolami, 2009) and on alternative algorithms based on similar principles (Frenkel and Smit, 2002; Gelman and Meng,

1998; Neal, 1996; Iba, 2001), very little attention has been devoted to the choice of path itself. The nearly universal practice is to use geometric averages of the initial and target distributons. Tantalizingly, Gelman and Meng (1998) derived the optimal paths for some toy models in the context of path sampling and showed that they vastly outperformed geometric averages. However, as choosing an optimal path is generally intractable, geometric averages still predominate.

In this chapter, we present a theoretical framework for evaluating alternative paths in terms of the asymptotic bias and variance of the estimator for large numbers of intermediate distributons. Based on this analysis, we propose a novel path defined by averaging moments of the initial and target distributions. We show that geometric averages and moment averages optimize different variational objectives, derive an asymptotically optimal piecewise linear schedule, and analyze the asymptotic performance of both paths. The proposed path often outperforms geometric averages at estimating partition functions of restricted Boltzmann machines (RBMs).

8.1 Estimating Partition Functions

In this section, we review the properties of AIS which are needed in this chapter. More details on AIS can be found in Section 2.3.5.

Suppose we have a probability distribution $p_b(\mathbf{x}) = f_b(\mathbf{x})/\mathcal{Z}_b$ defined on a space $\mathcal{X}$, where $f_b(\mathbf{x})$ can be computed efficiently for a given $\mathbf{x} \in \mathcal{X}$, and we are interested in estimating the partition function $\mathcal{Z}_b$. Annealed importance sampling (AIS) is an algorithm which estimates $\mathcal{Z}_b$ by gradually changing, or "annealing," a distribution. In particular, one must specify a sequence of $T+1$ intermediate distributions $p_t(\mathbf{x}) = f_t(\mathbf{x})/\mathcal{Z}_t$ for $t = 0, \dots T$, where $p_a(\mathbf{x}) = p_0(\mathbf{x})$ is a tractable initial distribution, and $p_b(\mathbf{x}) = p_T(\mathbf{x})$ is the intractable target distribution. For simplicity, assume all distributions are strictly positive on $\mathcal{X}$. For each p_t, one must also specify an MCMC transition operator $\mathcal{T}_t$ (e.g. Gibbs sampling) which leaves p_t invariant. AIS alternates between MCMC transitions and importance sampling updates, as shown in Alg 4.

The output of AIS is an unbiased estimate $\hat{\mathcal{Z}}_b$ of $\mathcal{Z}_b$. Remarkably, unbiasedness holds even in the context of *non-equilibrium* samples along the chain (Neal, 2001a; Jarzynski, 1997). However, unless the intermediate distributions and transition operators are carefully chosen, $\hat{\mathcal{Z}}_b$ may have high variance and be far from $\mathcal{Z}_b$ with high probability.

The mathematical formulation of AIS leaves much flexibility for choosing intermediate distributions. However, one typically defines a path $\gamma : [0, 1] \to \mathcal{P}$ through some family $\mathcal{P}$ of distributions. The intermediate distributions p_t are chosen to be points along this path corresponding to a *schedule* $0 = \beta_0 < \beta_1 < \ldots < \beta_T = 1$. One

Algorithm 4 Annealed Importance Sampling

for $k = 1$ to K **do**
  $\mathbf{x}_0 \leftarrow$ sample from $p_0(\mathbf{x})$
  $w^{(k)} \leftarrow \mathcal{Z}_a$
  for $t = 1$ to T **do**
    $w^{(k)} \leftarrow w^{(k)} \frac{f_t(\mathbf{x}_{t-1})}{f_{t-1}(\mathbf{x}_{t-1})}$
    $\mathbf{x}_t \leftarrow$ sample from $\mathcal{T}_t(\mathbf{x} \mid \mathbf{x}_{t-1})$
  end for
end for
return $\hat{\mathcal{Z}}_b = \sum_{k=1}^{K} w^{(k)}/K$

typically uses the geometric path γ_{GA}, defined in terms of geometric averages of p_a and p_b:

$$p_\beta(\mathbf{x}) = f_\beta(\mathbf{x})/\mathcal{Z}(\beta) = f_a(\mathbf{x})^{1-\beta} f_b(\mathbf{x})^\beta / \mathcal{Z}(\beta). \tag{8.1}$$

Commonly, f_a is the uniform distribution, and (8.1) reduces to $p_\beta(\mathbf{x}) = f_b(\mathbf{x})^\beta/\mathcal{Z}(\beta)$. This motivates the term "annealing," and β resembles an inverse temperature parameter. As in simulated annealing, the "hotter" distributions often allow faster mixing between modes which are isolated in p_b.

AIS is closely related to a broader family of techniques for posterior inference and partition function estimation, all based on the following identity from statistical physics:

$$\log \mathcal{Z}_b - \log \mathcal{Z}_a = \int_0^1 \mathbb{E}_{\mathbf{x}\sim p_\beta} \left[\frac{\mathrm{d}}{\mathrm{d}\beta} \log f_\beta(\mathbf{x}) \right] \mathrm{d}\beta. \tag{8.2}$$

Thermodynamic integration (Frenkel and Smit, 2002) estimates (8.2) using numerical quadrature, and path sampling (Gelman and Meng, 1998) does so with Monte Carlo integration. The weight update in AIS can be seen as a finite difference approximation. Tempered transitions (Neal, 1996) is a Metropolis-Hastings proposal operator which heats up and cools down the distribution, and computes an acceptance ratio by approximating (8.2).

The choices of a path and a schedule are central to all of these methods. Most work on adapting paths has focused on tuning schedules along a geometric path (Neal, 1996; Behrens et al., 2012; Calderhead and Girolami, 2009). Neal (1996) showed that the geometric *schedule* was optimal for annealing the scale parameter of a Gaussian, and Behrens et al. (2012) extended this result more broadly. The aim

of this paper is to propose, analyze, and evaluate a novel alternative to γ_{GA} based on averaging moments of the initial and target distributions.

8.2 Surprising behavior of geometric averages

While the geometric averages formula (8.1) has a compelling motivation in terms of annealing, geometric averages of distributions can sometimes exhibit surprising behavior. This section discusses two examples. While the examples are chosen to illustrate particular pathologies, they are distillations of problems which we have actually encountered when applying AIS, and which only became apparent after considerable time spent debugging.

For the first example, consider a mixture of two spherical Gaussians with a binary threshold observation model:

$$\begin{aligned} z &\sim \text{Bernoulli}(0.5) \\ \boldsymbol{\mu} &= \begin{cases} (0,0,0,0,0)^T & \text{if } z = 0 \\ (5,5,5,5,5)^T & \text{if } z = 1 \end{cases} \\ \mathbf{x} \,|\, \boldsymbol{\mu} &\sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{I}) \\ y_i &= \begin{cases} 0 & \text{if } x_i \leq 0 \\ 1 & \text{if } x_i > 0 \end{cases} \end{aligned}$$

This example is representative of issues that arise in evaluating predictive likelihood of discrete models such as BG+G. For the initial distribution, let us use $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ independent of z. Suppose we observe the vector $(1,1,1,1,1)^T$. Taking the geometric average of the two distributions, the intermediate distributions are given by:

$$p_\beta(z, \mathbf{x}) \propto p(z)\, \mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{I})^{1-\beta}\, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{I})^{\beta},$$

subject to the constraint of $x_i > 0$.

By inspection, we see that $z = 1$ with probability 0.5 in the initial distribution and roughly $1 - 2^{-5}$ in the target distribution. We may expect, therefore, that the probability gradually increases. However, as shown in Figure 8-1, the probability first dips close to zero before finally becoming large around $\beta = 0.9$. Figure 8-1 plots one term in the geometric average, $\mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{I})^{1-\beta}\, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{I})^{\beta}$, for different values of β. Not only does the distribution change, but the normalizing constant changes as well. (This example shows that the geometric average of two directed models is not

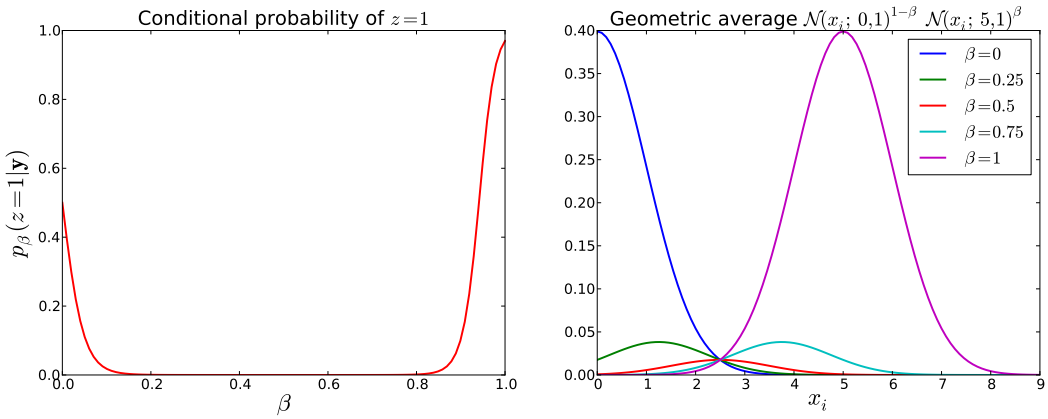


Figure 8-1: **(left)** the conditional probability of $z = 1$ for the intermediate distributions in the mixture example, as a function of β. The probability is close to zero for most values of β. **(right)** The geometric averages of two non-overlapping gaussians, for different values of β. Notice that the geometric averages are small in magnitude when the distributions have little overlap. This explains why the intermediate distributions are biased towards $z = 0$, since that explanation agrees with the initial distribution.

necessarily a directed model.) In Section 8.4.1, we will see that the geometric average favors explanations which are likely under *both* the intial and target distributions; in this case, $z = 0$ is favored because it overlaps more with the initial distribution.

From this example, one may conclude that the problem with geometric averages is that the intermediate distributions are unnormalized. What if we modify the definition to separately normalize each explanation under the prior? In other words, define

$$p_\beta(\mathbf{x} \,|\, z) = \frac{p_a(\mathbf{x})^{1-\beta}\, p_b(\mathbf{x} \,|\, z)^\beta}{\int p_a(\mathbf{x})^{1-\beta}\, p_b(\mathbf{x} \,|\, z)^\beta \,\mathrm{d}\mathbf{x}}.$$

Unlike traditional geometric averages, this method preserves the directed structure. It yields sensible results in the mixture example. However, consider another model where the latent variable is continuous:

$$\begin{aligned} u &\sim \mathcal{N}(0, 3) \\ \mathbf{x} &\sim \mathcal{N}\left(u\mathbf{1}, \mathbf{I}\right) \\ y_i &= \begin{cases} 0 & \text{if } x_i \leq 0 \\ 1 & \text{if } x_i > 0 \end{cases} \end{aligned}$$

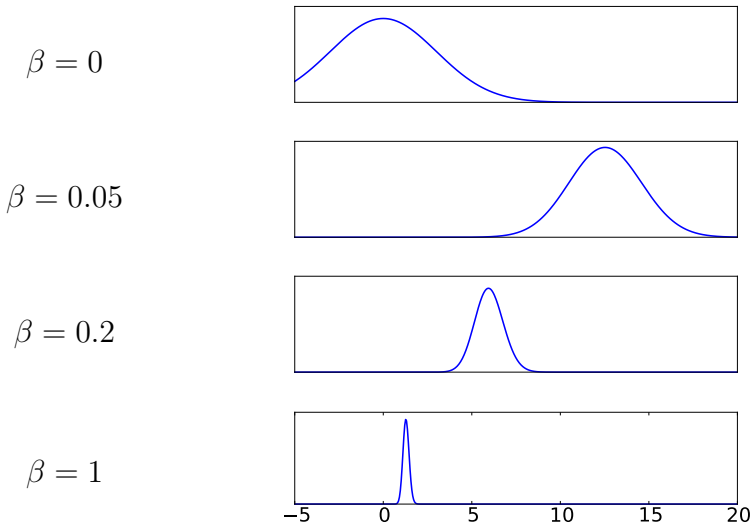


Figure 8-2: Posterior over u for intermediate distributions in the continuous example. The intermediate distributions are centered on values which are more extreme than either the initial or target distributions.

This model corresponds to evaluating the predictive likelihood of a continuous model such as GG+G. As before, let $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ independent of u in the initial distribution. Then the CPDs for the intermediate distributions are given by:

$$\begin{aligned} p_\beta(\mathbf{x} \,|\, u) &= \frac{\mathcal{N}(\mathbf{x} \,|\, \mathbf{0}, \mathbf{I})^{1-\beta} \mathcal{N}(\mathbf{x} \,|\, u\mathbf{1}, \mathbf{I})^{\beta}}{\int \mathcal{N}(\mathbf{x} \,|\, \mathbf{0}, \mathbf{I})^{1-\beta} \mathcal{N}(\mathbf{x} \,|\, u\mathbf{1}, \mathbf{I})^{\beta} \, \mathrm{d}\mathbf{x}} \\ &= \mathcal{N}(\mathbf{x} \,|\, \beta u \mathbf{1}, \mathbf{I}). \end{aligned}$$

Suppose we are given the observation vector

$$\mathbf{y} = (\underbrace{1, 1, \ldots, 1}_{\text{90 1's}}, \underbrace{0, \ldots, 0}_{\text{10 0's}})^T.$$

The posteriors over u for various values of β are shown in Figure 8-2. Effectively, the model is able to compensate for the shrinking effect of β by making u large. The intermediate distributions, therefore, take values more extreme than either the initial or target distributions.

These examples show that it is not always easy to come up with an annealing path where the intermediate distributions are sensible. The rest of this chapter focuses on

a novel annealing path defined by averaging the moments of the initial and target distributions.

8.3 Analyzing AIS Paths

When analyzing AIS, it is common to assume *perfect transitions*, i.e. that each transition operator $\mathcal{T}_t$ returns an independent and exact sample from the distribution p_t (Neal, 2001a). This corresponds to the (somewhat idealized) situation where the Markov chains mix quickly. As Neal (2001a) pointed out, assuming perfect transitions, the Central Limit Theorem shows that the samples $w^{(k)}$ are approximately log-normally distributed. In this case, the variances $\mathrm{var}(w^{(k)})$ and $\mathrm{var}(\log w^{(k)})$ are both monotonically related to $\mathbb{E}[\log w^{(k)}]$. Therefore, our analysis focuses on $\mathbb{E}[\log w^{(k)}]$.

Assuming perfect transitions, the expected log weights are given by:

$$\begin{aligned}\mathbb{E}[\log w^{(k)}] &= \log \mathcal{Z}_a + \sum_{t=0}^{T-1} \mathbb{E}_{p_t}[\log f_{t+1}(\mathbf{x}) - \log f_t(\mathbf{x})] \\ &= \log \mathcal{Z}_b - \sum_{t=0}^{T-1} \mathrm{D}_{\mathrm{KL}}(p_t \| p_{t+1}). \end{aligned} \tag{8.3}$$

In other words, each $\log w^{(k)}$ can be seen as a biased estimator of $\log \mathcal{Z}_b$, where the bias $\delta = \log \mathcal{Z}_b - \mathbb{E}[\log w^{(k)}]$ is given by the sum of KL divergences $\sum_{t=0}^{T-1} \mathrm{D}_{\mathrm{KL}}(p_t \| p_{t+1})$.

Suppose $\mathcal{P}$ is a family of probability distributions parameterized by $\boldsymbol{\theta} \in \Theta$, and the $T+1$ distributions $p_0, \ldots, p_T$ are chosen to be linearly spaced along a path $\gamma : [0,1] \to \mathcal{P}$. Let $\boldsymbol{\theta}(\beta)$ represent the parameters of the distribution $\gamma(\beta)$. As T is increased, the bias δ decays like $1/T$, and the asymptotic behavior is determined by a functional $\mathcal{F}(\gamma)$.

Theorem 1. *Suppose $T+1$ distributions p_t are linearly spaced along a path γ. Assuming perfect transitions, if $\boldsymbol{\theta}(\beta)$ and the Fisher information matrix $\mathbf{G}_{\boldsymbol{\theta}}(\beta) = \mathrm{cov}_{\mathbf{x} \sim p_{\boldsymbol{\theta}}}(\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}))$ are continuous and piecewise smooth, then as $T \to \infty$ the bias δ behaves as follows:*

$$T\delta = T \sum_{t=0}^{T-1} \mathrm{D}_{\mathrm{KL}}(p_t \| p_{t+1}) \to \mathcal{F}(\gamma) \equiv \frac{1}{2} \int_0^1 \dot{\boldsymbol{\theta}}(\beta)^T \mathbf{G}_{\boldsymbol{\theta}}(\beta) \dot{\boldsymbol{\theta}}(\beta) d\beta, \tag{8.4}$$

where $\dot{\boldsymbol{\theta}}(\beta)$ represents the derivative of $\boldsymbol{\theta}$ with respect to β. [See Appendix B.1 for proof.]

This result reveals a relationship with path sampling, as Gelman and Meng (1998) showed that the variance of the path sampling estimator is proportional to the same functional. One useful result from their analysis is a derivation of the optimal schedule along a given path. In particular, the value of $\mathcal{F}(\gamma)$ using the optimal schedule is given by $\ell(\gamma)^2/2$, where ℓ is the Riemannian path length defined by

$$\ell(\gamma) = \int_0^1 \sqrt{\dot{\boldsymbol{\theta}}(\beta)^T \mathbf{G}_{\boldsymbol{\theta}}(\beta)\dot{\boldsymbol{\theta}}(\beta)}d\beta. \tag{8.5}$$

Intuitively, the optimal schedule allocates more distributions to regions where p_β changes quickly. While Gelman and Meng (1998) derived the optimal paths and schedules for some simple examples, they observed that this is intractable in most cases and recommended using geometric paths in practice.

The above analysis assumes perfect transitions, which can be unrealistic in practice because many distributions have separated modes between which mixing is difficult. As Neal (2001a) observed, in such cases, AIS can be viewed as having two sources of variance: that caused by variability within a mode, and that caused by misallocation of samples between modes. The former source of variance is well modeled by the perfect transitions analysis and can be made small by adding more intermediate distributions. The latter, however, can persist even with large numbers of intermediate distributions. While our theoretical analysis assumes perfect transitions, our proposed method often gave substantial improvement empirically in situations with poor mixing.

8.4 Moment Averaging

As discussed in Section 8.1, the typical choice of intermediate distributions for AIS is the geometric averages path γ_{GA} given by (8.1). In this section, we propose an alternative path for exponential family models. An exponential family model is defined as

$$p(\mathbf{x}) = \frac{1}{\mathcal{Z}(\boldsymbol{\eta})} h(\mathbf{x}) \exp\left(\boldsymbol{\eta}^T \mathbf{g}(\mathbf{x})\right), \tag{8.6}$$

where $\boldsymbol{\eta}$ are the natural parameters and $\mathbf{g}$ are the sufficient statistics. Exponential families include a wide variety of statistical models, including Markov random fields.

In exponential families, geometric averages correspond to averaging the natural

parameters:

$$\boldsymbol{\eta}(\beta) = (1-\beta)\boldsymbol{\eta}(0) + \beta\boldsymbol{\eta}(1). \tag{8.7}$$

Exponential families can also be parameterized in terms of their moments $\mathbf{s} = \mathbb{E}[\mathbf{g}(\mathbf{x})]$. For any minimal exponential family (*i.e.* one whose sufficient statistics are linearly independent), there is a one-to-one mapping between moments and natural parameters (Wainwright and Jordan, 2008, p. 64). We propose an alternative to γ_{GA} called the *moment averages* path, denoted γ_{MA}, and defined by averaging the moments of the initial and target distributions:

$$\mathbf{s}(\beta) = (1-\beta)\mathbf{s}(0) + \beta\mathbf{s}(1). \tag{8.8}$$

This path exists for any exponential family model, since the set of realizable moments is convex (Wainwright and Jordan, 2008). It is unique, since $\mathbf{g}$ is unique up to affine transformation.

As an illustrative example, consider a multivariate Gaussian distribution parameterized by the mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. The moments are $\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu}$ and $-\frac{1}{2}\mathbb{E}[\mathbf{x}\mathbf{x}^T] = -\frac{1}{2}(\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T)$. By plugging these into (8.8), we find that γ_{MA} is given by:

$$\boldsymbol{\mu}(\beta) = (1-\beta)\boldsymbol{\mu}(0) + \beta\boldsymbol{\mu}(1) \tag{8.9}$$

$$\boldsymbol{\Sigma}(\beta) = (1-\beta)\boldsymbol{\Sigma}(0) + \beta\boldsymbol{\Sigma}(1) + \beta(1-\beta)(\boldsymbol{\mu}(1) - \boldsymbol{\mu}(0))(\boldsymbol{\mu}(1) - \boldsymbol{\mu}(0))^T. \tag{8.10}$$

In other words, the means are linearly interpolated, and the covariances are linearly interpolated and stretched in the direction connecting the two means. Intuitively, this stretching is a useful property, because it increases the overlap between successive distributions with different means. A comparison of the two paths is shown in Figure 8-3.

Next consider the example of a restricted Boltzmann machine (RBM), a widely used model in deep learning. A binary RBM is a Markov random field over binary vectors $\mathbf{v}$ (the visible units) and $\mathbf{h}$ (the hidden units), and which has the distribution

$$p(\mathbf{v}, \mathbf{h}) \propto \exp\left(\mathbf{a}^T\mathbf{v} + \mathbf{b}^T\mathbf{h} + \mathbf{v}^T\mathbf{W}\mathbf{h}\right). \tag{8.11}$$

The parameters of the model are the visible biases $\mathbf{a}$, the hidden biases $\mathbf{b}$, and the weights $\mathbf{W}$. Since these parameters are also the natural parameters in the exponential family representation, γ_{GA} reduces to linearly averaging the biases and the weights. The sufficient statistics of the model are the visible activations $\mathbf{v}$, the hidden

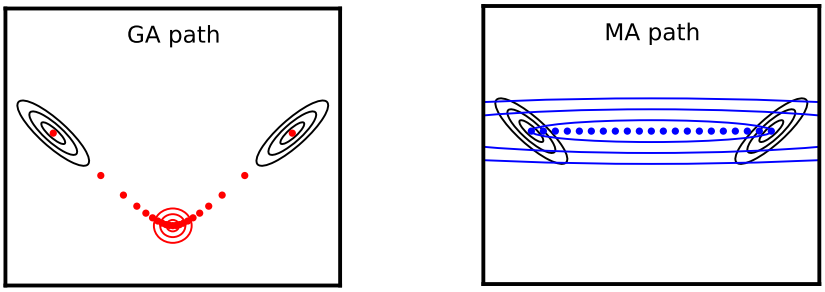


Figure 8-3: Comparison of γ_{GA} and γ_{MA} for multivariate Gaussians: intermediate distribution for $\beta = 0.5$, and $\boldsymbol{\mu}(\beta)$ for β evenly spaced from 0 to 1.

activations $\mathbf{h}$, and the products $\mathbf{v}\mathbf{h}^T$. Therefore, γ_{MA} is defined by:

$$\mathbb{E}[\mathbf{v}]_\beta = (1-\beta)\mathbb{E}[\mathbf{v}]_0 + \beta\mathbb{E}[\mathbf{v}]_1 \tag{8.12}$$

$$\mathbb{E}[\mathbf{h}]_\beta = (1-\beta)\mathbb{E}[\mathbf{h}]_0 + \beta\mathbb{E}[\mathbf{h}]_1 \tag{8.13}$$

$$\mathbb{E}[\mathbf{v}\mathbf{h}^T]_\beta = (1-\beta)\mathbb{E}[\mathbf{v}\mathbf{h}^T]_0 + \beta\mathbb{E}[\mathbf{v}\mathbf{h}^T]_1 \tag{8.14}$$

For many models of interest, including RBMs, it is infeasible to determine γ_{MA} exactly, as it requires solving two often intractable problems: (1) computing the moments of p_b, and (2) solving for model parameters which match the averaged moments $\mathbf{s}(\beta)$. However, much work has been devoted to practical approximations (Hinton, 2002; Tieleman, 2008), some of which we use in our experiments with intractable models. Since it would be infeasible to moment match every β_t even approximately, we introduce the moment averages spline (MAS) path, denoted γ_{MAS}. We choose a set of R values $\beta_1, \ldots, \beta_R$ called *knots*, and solve for the natural parameters $\boldsymbol{\eta}(\beta_j)$ to match the moments $\mathbf{s}(\beta_j)$ for each knot. We then interpolate between the knots using geometric averages. The analysis of Section 8.4.2 shows that, under the assumption of perfect transitions, using γ_{MAS} in place of γ_{MA} does not affect the cost functional $\mathcal{F}$ defined in Theorem 4.

8.4.1 Variational Interpretation

By interpreting γ_{GA} and γ_{MA} as optimizing different variational objectives, we gain additional insight into their behavior. For geometric averages, the intermediate distribution $\gamma_{GA}(\beta)$ minimizes a weighted sum of KL divergences to the initial and

target distributions:

$$p_\beta^{(GA)} = \arg\min_q \ (1-\beta)\mathrm{D}_{\mathrm{KL}}(q\|p_a) + \beta\mathrm{D}_{\mathrm{KL}}(q\|p_b). \tag{8.15}$$

On the other hand, γ_{MA} minimizes the weighted sum of KL divergences in the reverse direction:

$$p_\beta^{(MA)} = \arg\min_q \ (1-\beta)\mathrm{D}_{\mathrm{KL}}(p_a\|q) + \beta\mathrm{D}_{\mathrm{KL}}(p_b\|q). \tag{8.16}$$

See Appendix B.2 for the derivations. The objective function (8.15) is minimized by a distribution which puts significant mass only in the "intersection" of p_a and p_b, i.e. those regions which are likely under both distributions. By contrast, (8.16) encourages the distribution to be spread out in order to capture all high probability regions of both p_a and p_b. This interpretation helps explain why the intermediate distributions in the Gaussian example of Figure 8-3 take the shape that they do. In our experiments, we found that γ_{MA} often gave more accurate results than γ_{GA} because the intermediate distributions captured regions of the target distribution which were missed by γ_{GA}.

8.4.2 Asymptotics under Perfect Transitions

In general, we found that γ_{GA} and γ_{MA} can look very different. Intriguingly, both paths always result in the same value of the cost functional $\mathcal{F}(\gamma)$ of Theorem 4 for any exponential family model. Furthermore, nothing is lost by using the spline approximation γ_{MAS} in place of γ_{MA}:

Theorem 2. *For any exponential family model with natural parameters $\boldsymbol{\eta}$ and moments $\mathbf{s}$, all three paths share the same value of the cost functional:*

$$\mathcal{F}(\gamma_{GA}) = \mathcal{F}(\gamma_{MA}) = \mathcal{F}(\gamma_{MAS}) = \frac{1}{2}(\boldsymbol{\eta}(1) - \boldsymbol{\eta}(0))^T(\mathbf{s}(1) - \mathbf{s}(0)). \tag{8.17}$$

Proof. The two parameterizations of exponential families satisfy the relationship $\mathbf{G}_{\boldsymbol{\eta}}\dot{\boldsymbol{\eta}} = \dot{\mathbf{s}}$ (Amari and Nagaoka, 2000, sec. 3.3). Therefore, $\mathcal{F}(\gamma)$ can be rewritten as $\frac{1}{2}\int_0^1 \dot{\boldsymbol{\eta}}(\beta)^T\dot{\mathbf{s}}(\beta)\,\mathrm{d}\beta$. Because γ_{GA} and γ_{MA} linearly interpolate the natural parameters

and moments respectively,

$$\mathcal{F}(\gamma_{GA}) = \frac{1}{2}(\boldsymbol{\eta}(1) - \boldsymbol{\eta}(0))^T \int_0^1 \dot{\mathbf{s}}(\beta)\,\mathrm{d}\beta = \frac{1}{2}(\boldsymbol{\eta}(1) - \boldsymbol{\eta}(0))^T(\mathbf{s}(1) - \mathbf{s}(0)) \tag{8.18}$$

$$\mathcal{F}(\gamma_{MA}) = \frac{1}{2}(\mathbf{s}(1) - \mathbf{s}(0))^T \int_0^1 \dot{\boldsymbol{\eta}}(\beta)\,\mathrm{d}\beta = \frac{1}{2}(\mathbf{s}(1) - \mathbf{s}(0))^T(\boldsymbol{\eta}(1) - \boldsymbol{\eta}(0)). \tag{8.19}$$

Finally, to show that $\mathcal{F}(\gamma_{MAS}) = \mathcal{F}(\gamma_{MA})$, observe that γ_{MAS} uses the geometric path between each pair of knots $\gamma(\beta_j)$ and $\gamma(\beta_{j+1})$, while γ_{MA} uses the moments path. The above analysis shows the costs must be equal for each segment, and therefore equal for the entire path. □

This analysis shows that all three paths result in the same expected log weights asymptotically, assuming perfect transitions. There are several caveats, however. First, we have noticed experimentally that γ_{MA} often yields substantially more accurate estimates of $\mathcal{Z}_b$ than γ_{GA} even when the average log weights are comparable. Second, the two paths can have very different mixing properties, which can strongly affect the results. Third, Theorem 2 assumes *linear* schedules, and in principle there is room for improvement if one is allowed to tune the schedule.

For instance, consider annealing between two Gaussians $p_a = \mathcal{N}(\mu_a, \sigma)$ and $p_b = \mathcal{N}(\mu_b, \sigma)$. The optimal schedule for γ_{GA} is a linear schedule with cost $\mathcal{F}(\gamma_{GA}) = O(d^2)$, where $d = |\mu_b - \mu_a|/\sigma$. Using a linear schedule, the moment path also has cost $O(d^2)$, consistent with Theorem 2. However, most of the cost of the path results from instability near the endpoints, where the variance changes suddenly. Using an optimal schedule, which allocates more distributions near the endpoints, the cost functional falls to $O((\log d)^2)$, which is within a constant factor of the optimal path derived by Gelman and Meng (1998). (See Appendix B.3 for the derivations.) In other words, while $\mathcal{F}(\gamma_{GA}) = \mathcal{F}(\gamma_{MA})$, they achieve this value for different reasons: γ_{GA} follows an optimal schedule along a bad path, while γ_{MA} follows a bad schedule along a near-optimal path. We speculate that, combined with the procedure of Section 8.4.3 for choosing a schedule, moment averages may result in large reductions in the cost functional for some models.

Figure 8-4 shows a visualization of both annealing paths, as well as the optimal path given analytically by Gelman and Meng (1998). The space of univariate Gaussian distributions can be viewed as a Riemannian manifold where the Riemannian metric is Fisher information (Amari and Nagaoka, 2000). As pointed out by Gelman and Meng (1998), the cost functional for the optimal schedule depends on the path length (8.5) on the manifold. The Riemannian metric is visualized in terms of balls of the metric, and the path length can be thought of as the number of balls the path

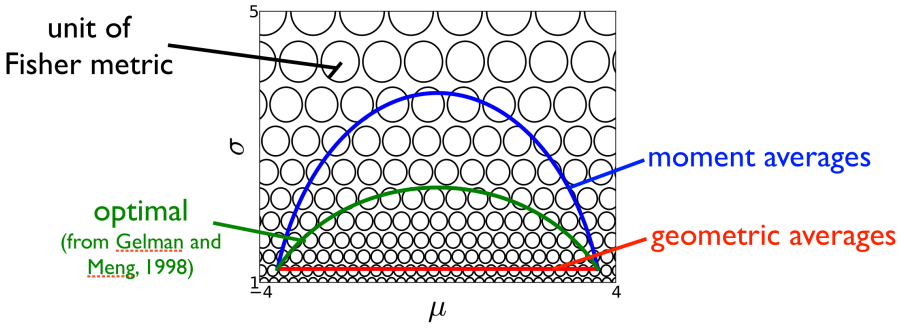


Figure 8-4: A visualization of the Riemannian manifold of univariate Gaussian distributions and several choices of annealing path. The balls correspond to the Riemannian metric (Fisher information). The cost of a path under the optimal schedule depends on the path length, which can be visualized as the number of balls it crosses.

crosses. It is interesting that the optimal path and the moment averages path, while not identical, have a qualitatively similar shape.

8.4.3 Optimal Binned Schedules

In general, it is hard to choose a good schedule for a given path. However, consider the set of *binned schedules*, where the path is divided into segments, some number T_j of intermediate distributions are allocated to each segment, and the distributions are spaced linearly within each segment. Under the assumption of perfect transitions, there is a simple formula for an asymptotically optimal binned schedule which requires only the parameters obtained through moment averaging:

Theorem 3. *Let γ be any path for an exponential family model defined by a set of knots β_j, each with natural parameters $\boldsymbol{\eta}_j$ and moments $\mathbf{s}_j$, connected by segments of either γ_{GA} or γ_{MA} paths. Then, under the assumption of perfect transitions, an asymptotically optimal allocation of intermediate distributions to segments is given by:*

$$T_j \propto \sqrt{(\boldsymbol{\eta}_{j+1} - \boldsymbol{\eta}_j)^T (\mathbf{s}_{j+1} - \mathbf{s}_j)}. \tag{8.20}$$

Proof. By Theorem 2, the cost functional for segment j is $F_j = \frac{1}{2}(\boldsymbol{\eta}_{j+1} - \boldsymbol{\eta}_j)^T(\mathbf{s}_{j+1} - \mathbf{s}_j)$. Hence, with T_j distributions allocated to it, it contributes F_j/T_j to the total cost. The values of T_j which minimize $\sum_j F_j/T_j$ subject to $\sum_j T_j = T$ and $T_j \geq 0$

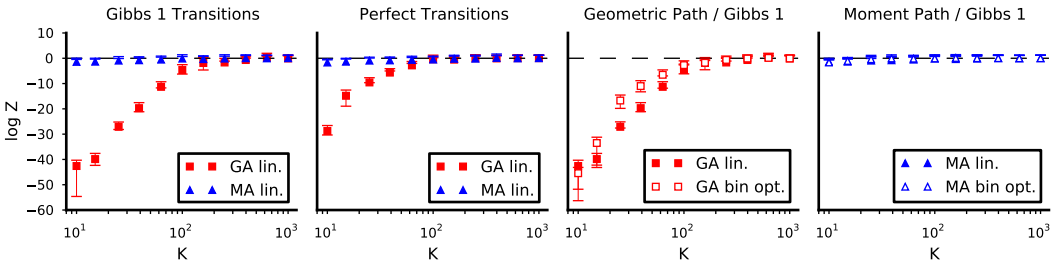


Figure 8-5: Estimates of $\log \mathcal{Z}_b$ for a normalized Gaussian as T, the number of intermediate distributions, is varied. True value: $\log \mathcal{Z}_b = 0$. Error bars show bootstrap 95% confidence intervals. (Best viewed in color.)

are given by $T_j \propto \sqrt{F_j}$. □

8.5 Experimental Results

In order to compare our proposed path with geometric averages, we ran AIS using each path to estimate partition functions of several probability distributions. For all of our experiments, we report two sets of results. First, we show the estimates of $\log \mathcal{Z}$ as a function of T, the number of intermediate distributions, in order to visualize the amount of computation necessary to obtain reasonable accuracy. Second, as recommended by Neal (2001a), we report the effective sample size (ESS) of the weights for a large T. This statistic roughly measures how many independent samples one obtains using AIS.[1] All results are based on 5,000 independent AIS runs, so the maximum possible ESS is 5,000.

8.5.1 Annealing Between Two Distant Gaussians

In our first experiment, the initial and target distributions were the two Gaussians shown in Fig. 8-3, whose parameters are $\mathcal{N}\left(\binom{-10}{0}, \begin{pmatrix} 1 & -0.85 \\ -0.85 & 1 \end{pmatrix}\right)$ and $\mathcal{N}\left(\binom{10}{0}, \begin{pmatrix} 1 & 0.85 \\ 0.85 & 1 \end{pmatrix}\right)$. As both distributions are normalized, $\mathcal{Z}_a = \mathcal{Z}_b = 1$. We compared γ_{GA} and γ_{MA} both

[1]The ESS is defined as $K/(1 + s^2(w_*^{(k)}))$ where $s^2(w_*^{(k)})$ is the sample variance of the normalized weights (Neal, 2001a). In general, one should regard ESS estimates cautiously, as they can give misleading results in cases where an algorithm completely misses an important mode of the distribution. However, as we report the ESS in cases where the estimated partition functions are close to the true value (when known) or agree closely with each other, we believe the statistic is meaningful in our comparisons.

under perfect transitions, and using the Gibbs transition operator. We also compared linear schedules with the optimal binned schedules of Section 8.4.3, using 10 segments evenly spaced from 0 to 1.

Figure 8-5 shows the estimates of $\log \mathcal{Z}_b$ for T ranging from 10 to 1,000. Observe that with 1,000 intermediate distributions, all paths yielded accurate estimates of $\log \mathcal{Z}_b$. However, γ_{MA} needed fewer intermediate distributions to achieve accurate estimates. For example, with $T = 25$, γ_{MA} resulted in an estimate within one nat of $\log \mathcal{Z}_b$, while the estimate based on γ_{GA} was off by 27 nats.

This result may seem surprising in light of Theorem 2, which implies that $\mathcal{F}(\gamma_{GA}) = \mathcal{F}(\gamma_{MA})$ for linear schedules. In fact, the average log weights for γ_{GA} and γ_{MA} were similar for all values of T, as the theorem would suggest; e.g., with $T = 25$, the average was -27.15 for γ_{MA} and -28.04 for γ_{GA}. However, because the γ_{MA} intermediate distributions were broader, enough samples landed in high probability regions to yield reasonable estimates of $\log \mathcal{Z}_b$.

8.5.2 Partition Function Estimation for RBMs

Our next set of experiments focused on restricted Boltzmann machines (RBMs), a building block of many deep learning models (see Section 8.4). We considered RBMs trained with three different methods: contrastive divergence (CD) (Hinton, 2002) with one step (CD1), CD with 25 steps (CD25), and persistent contrastive divergence (PCD) (Tieleman, 2008). All of the RBMs were trained on the MNIST handwritten digits dataset (LeCun et al., 1998), which has long served as a benchmark for deep learning algorithms. We experimented both with small, tractable RBMs and with full-size, intractable RBMs.

Since it is hard to compute γ_{MA} exactly for RBMs, we used the moments spline path γ_{MAS} of Section 8.4 with the 9 knot locations $0.1, 0.2, \ldots, 0.9$. We considered the two initial distributions discussed by Salakhutdinov and Murray (2008): (1) the uniform distribution, equivalent to an RBM where all the weights and biases are set to 0, and (2) the *base rate RBM*, where the weights and hidden biases are set to 0, and the visible biases are set to match the average pixel values over the MNIST training set.

Small, Tractable RBMs: To better understand the behavior of γ_{GA} and γ_{MAS}, we first evaluated the paths on RBMs with only 20 hidden units. In this setting, it is feasible to exactly compute the partition function and moments and to generate exact samples by exhaustively summing over all 2^{20} hidden configurations. The moments of the target RBMs were computed exactly, and moment matching was performed with conjugate gradient using the exact gradients.

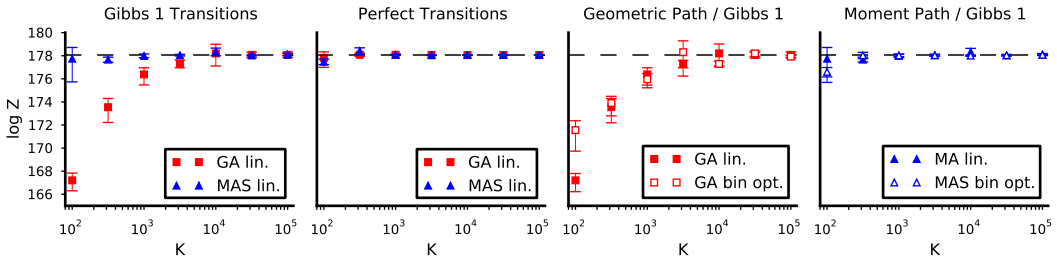


Figure 8-6: Estimates of $\log \mathcal{Z}_b$ for the tractable PCD(20) RBM as T, the number of intermediate distributions, is varied. Error bars indicate bootstrap 95% confidence intervals. (Best viewed in color.)

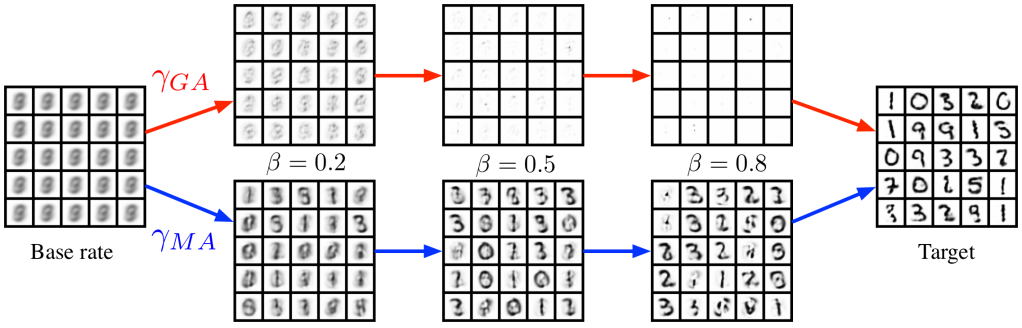


Figure 8-7: Visible activations for samples from the PCD(500) RBM. **(left)** base rate RBM, $\beta = 0$ **(top)** geometric path **(bottom)** MAS path **(right)** target RBM, $\beta = 1$.

The results are shown in Figure 8-6 and Table 8.1. Under perfect transitions, γ_{GA} and γ_{MAS} were both able to accurately estimate $\log \mathcal{Z}_b$ using as few as 100 intermediate distributions. However, using the Gibbs transition operator, γ_{MAS} gave accurate estimates using fewer intermediate distributions and achieved a higher ESS at $T = 100{,}000$. To check that the improved performance didn't rely on accurate moments of p_b, we repeated the experiment with highly biased moments.[2] The differences in $\log \hat{\mathcal{Z}}_b$ and ESS compared to the exact moments condition were not statistically significant.

Full-size, Intractable RBMs: For intractable RBMs, moment averaging re-

[2]In particular, we computed the biased moments from the conditional distributions of the hidden units given the MNIST training examples, where each example of digit class i was counted $i + 1$ times.

		CD1(20)			PCD(20)		
$p_a(\mathbf{v})$	path & schedule	$\log \mathcal{Z}_b$	$\log \hat{\mathcal{Z}}_b$	ESS	$\log \mathcal{Z}_b$	$\log \hat{\mathcal{Z}}_b$	ESS
uniform	GA linear	279.59	279.60	248	178.06	177.99	204
uniform	GA optimal binned		279.51	124		177.92	142
uniform	MAS linear		279.59	**2686**		178.09	**289**
uniform	MAS optimal binned		279.60	**2619**		178.08	**934**

Table 8.1: Comparing estimates of $\log \mathcal{Z}_b$ and effective sample size (ESS) for tractable RBMs. Results are shown for $T = 100{,}000$ intermediate distributions, with 5,000 chains and Gibbs transitions. Bolded values indicate ESS estimates that are not significantly different from the largest value (bootstrap hypothesis test with 1,000 samples at $\alpha = 0.05$ significance level). The maximum possible ESS is 5,000.

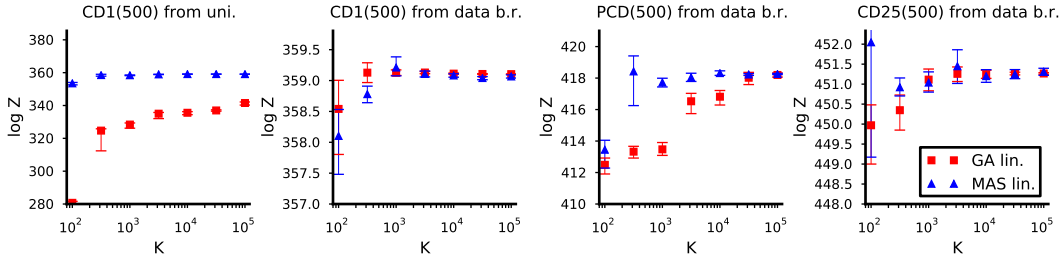


Figure 8-8: Estimates of $\log \mathcal{Z}_b$ for intractable RBMs. Error bars indicate bootstrap 95% confidence intervals. (Best viewed in color.)

quired approximately solving two intractable problems: moment estimation for the target RBM, and moment matching. We estimated the moments from 1,000 independent Gibbs chains, using 10,000 Gibbs steps with 1,000 steps of burn-in. The moment averaged RBMs were trained using PCD. (We used 50,000 updates with a fixed learning rate of 0.01 and no momentum.) In addition, we ran a cheap, inaccurate moment matching scheme (denoted MAS cheap) where visible moments were estimated from the empirical MNIST base rate and the hidden moments from the conditional distributions of the hidden units given the MNIST digits. Intermediate RBMs were fit using 1,000 PCD updates and 100 particles, for a total computational cost far smaller than that of AIS itself. Results of both methods are shown in Figure 8-8 and Table 8.2. Overall, the MAS results compare favorably with those of GA

		CD1(500)		PCD(500)		CD25(500)	
$p_a(\mathbf{v})$	path	$\log \hat{\mathcal{Z}}_b$	ESS	$\log \hat{\mathcal{Z}}_b$	ESS	$\log \hat{\mathcal{Z}}_b$	ESS
uniform	GA linear	341.53	4	417.91	169	451.34	13
uniform	MAS linear	359.09	3076	418.27	**620**	449.22	**12**
uniform	MAS cheap linear	359.09	**3773**	418.33	5	450.90	**30**
base rate	GA linear	359.10	**4924**	418.20	159	451.27	**2888**
base rate	MAS linear	359.07	2203	418.26	**1460**	451.31	304
base rate	MAS cheap linear	359.09	2465	418.25	359	451.14	244

Table 8.2: Comparing estimates of $\log \mathcal{Z}_b$ and effective sample size (ESS) for intractable RBMs. Results are shown for $T = 100{,}000$ intermediate distributions, with 5,000 chains and Gibbs transitions. Bolded values indicate ESS estimates that are not significantly different from the largest value (bootstrap hypothesis test with 1,000 samples at $\alpha = 0.05$ significance level). The maximum possible ESS is 5,000.

on both of our metrics. Performance was comparable under MAS cheap, suggesting that γ_{MAS} can be approximated cheaply and effectively. As with the tractable RBMs, we found that optimal binned schedules made little difference in performance, so we focus here on linear schedules.

The most serious failure was γ_{GA} for CD1(500) with uniform initialization, which underestimated our best estimates of the log partition function (and hence overestimated held-out likelihood) by nearly 20 nats. The geometric path from uniform to PCD(500) and the moments path from uniform to CD1(500) also resulted in underestimates, though less drastic. The rest of the paths agreed closely with each other on their partition function estimates, although some methods achieved substantially higher ESS values on some RBMs. One conclusion is that it's worth exploring multiple initializations and paths for a given RBM in order to ensure accurate results.

Figure 8-7 compares samples along γ_{GA} and γ_{MAS} for the PCD(500) RBM using the base rate initialization. For a wide range of β values, the γ_{GA} RBMs assigned most of their probability mass to blank images. As discussed in Section 8.4.1, γ_{GA} prefers configurations which are probable under both the initial and target distributions. In this case, the hidden activations were closer to uniform conditioned on a blank image than on a digit, so γ_{GA} preferred blank images. By contrast, γ_{MAS} yielded diverse, blurry digits which gradually coalesced into crisper ones.

8.6 Conclusion

We presented a theoretical analysis of the performance of AIS paths and proposed a novel path for exponential families based on averaging moments. We gave a variational interpretation of this path and derived an asymptotically optimal piecewise linear schedule. Moment averages performed well empirically at estimating partition functions of RBMs. We hope moment averaging can also improve other path-based sampling algorithms which typically use geometric averages, such as path sampling (Gelman and Meng, 1998), parallel tempering (Iba, 2001), and tempered transitions (Neal, 1996).

Chapter 9

Conclusions and future work

9.1 Contributions

In Chapter 3, we introduced a space of matrix decomposition models defined compositionally in terms of simple probabilistic modeling motifs and operations used to combine them. The models are organized into a grammar whose productions correspond to simple factorization models. This gives a compact notation for matrix decomposition models and highlights relationships between a variety of models from the literature. Chapter 4 proposed an inference procedure for this space of models based on recursively applying samplers specialized to the individual productions of the grammar. A greedy search through the space of models was able to recover the true structure for synthetic data and plausible structures for a variety of real-world datasets, all using the same code and no hand-tuned parameters.

Building upon this, Chapter 5 described an analogous compositional structure search for Gaussian process covariance kernels. On several time series datasets, this structure search yielded interpretable decompositions into phenomena occurring at different scales.

In Chapter 6, we introduced compositional importance sampling (CIS), a fully compositional algorithm for inference and marginal likelihood estimation. We analyzed the bias of the CIS estimator in the case of a clustering-within-low-rank model and showed that it is able to distinguish the first- and second-level models. This result also yields a bound on the error introduced by the greedy initialization procedure of Chapter 4.

A practical implementation of CIS will require advances in marginal likelihood estimators for the productions of the grammar. We believe progress has been hindered by a lack of quantitative frameworks for analyzing and evaluating partition function

estimators. Chapters 7 and 8 were both attempts at addressing this problem. In Chapter 7, we introduced a framework for evaluating marginal likelihood (ML) estimators against ground truth on synthetic data and compared a wide variety of ML estimation algorithms on three production rules from the grammar. We believe that having a rigorous evaluation method will spur accelerated progress on ML estimation algorithms.

Chapter 8 introduced a framework for improving annealed importance sampling (AIS), a state-of-the-art algorithm for estimating partition functions, and the one which performed the best in the experiments of Chapter 7. We introduced a framework for comparing different AIS paths—a widely overlooked factor which turns out to make a large difference in the performance of the estimator—and showed that averaging moments rather than natural parameters often yields substantial improvements in accuracy.

9.2 Future directions

We believe this work opens up more questions and avenues to explore than it closes off. There is much potential for extending the model spaces to more components, observation models, operators, and productions. Because of the combinatorial nature of modeling, the grammars of Chapters 3 and 5 capture only a small fraction of the models used in modern machine learning research. For instance, if there are 30 modeling motifs that machine learning researchers commonly apply, and our grammar captures 6 of them, then it still only captures $1/5^3$ of the models that are reachable within three productions. On the bright side, this implies that the benefit of adding more productions is superlinear. Twice as many productions yields 8 times as many third-level models.

Tensor decompositions are another model class where compositional structure search may prove fruitful. Some of the models presented in Section 3.4 were originally formulated as tensor models, and our grammar captured only the special case for matrices. In principle, tensor decompositions share the same compositional structure as matrix decompositions, so one could define a grammar analogous to that of Chapter 3. However, as the combinatorial explosion is even greater for tensors than for matrices, additional structure may need to be exploited in the search.

Expanding the space of models will require a better theoretical understanding of when compositional structure search is justified. The analysis of Section 6.2 gives a partial answer in the case of finding structure within a low rank approximation. Hopefully, that analysis will serve as a template for further work which determines, for instance, how to organize a space of tensor decomposition models so that it can

be searched efficiently.

Much work also remains to be done on estimating marginal likelihood for the productions themselves. Partition function estimation algorithms require choosing parameters which trade off time and accuracy, not to mention further design choices concerning the MCMC transition operators and the choice of annealing paths. Unless someone discovers magic values that perform well in all cases, an effective structure search system would need to make such choices automatically. The methods of Chapters 7 and 8 for quantitatively analyzing and evaluating partition function estimators open the door to using Bayesian optimization (Snoek et al., 2012) to tune the algorithms.

While most of this thesis has focused on algorithms for posterior inference and model scoring, there is also room for improvement in the search procedure itself. Bayesian optimization (Snoek et al., 2012) has recently emerged as a powerful technique for optimizing model hyperparameters. Conceivably, with an appropriate prior over the performances of different models, it could be used to search more efficiently through a compositional space of models compared to the current greedy search. Conversely, there are potential synergies in the opposite direction as well: Bayesian optimization currently relies on simple kernel structures such as the squared-exp kernel, so perhaps it could benefit from the Gaussian process structure search techniques of Chapter 5.

From the perspective of usability, one generally likes to know how much confidence to place in an algorithm's output. Rather than a point estimate of marginal likelihood, one would prefer confidence intervals. This could be done, at least heuristically, using the techniques of Chapter 7. While the technique for obtaining ML upper bounds is only mathematically justified for synthetic data generated from the model, one could obtain confidence intervals using the parametric bootstrap: fit the model to the data, generate synthetic data with the same hyperparameters, and evaluate the gap between the lower and upper ML bounds on the synthetic data. If the datasets are similar enough in terms of properties relied on by the estimator, the gap between the bounds on the synthetic data should translate to a confidence interval for the real data. Having confidence intervals would add a degree of reliability to a structure search system.

In mathematics, much progress was driven by the classification of simple groups. Having a complete enumeration of simple groups enables mathematicians to prove properties of groups in general by proving the statements for the simple groups and the operations used to combine them. In machine learning, we face an analogous question. Which models are fundamental, in terms of requiring their own special-purpose inference algorithms, and which models can be fit compositionally in terms of

simple component algorithms? A better understanding of the range of applicability of CIS and related techniques should help to focus algorithmic effort on the models where it is most needed.

Learning high-level representations of abstract concepts has long been a holy grail of AI resesarch. While it is difficult to pin down a formal definition of representation, we informally posit that a good representation is one which enables humans and machines to discover further patterns on top of it. But learning representations which support higher-level representations is precisely the basis of the compositional structure search techniques presented in this thesis. We believe the techniques and results presented here constitute a step towards the goal of learning high-level representations.

Appendix A

CIS derivations

A.1 Priors

We now motivate the choice of prior p_{wi}. For reasons discussed in Section 6.2.4.2, we would like the prior over $\mathbf{V}$ to be invariant to linear transformations of the latent space, *i.e.* transformations of the form $\mathbf{V} \mapsto \mathbf{T}\mathbf{V}$. This can be achieved with the uninformative prior $q(\mathbf{\Lambda_V}) \propto |\mathbf{\Lambda_V}|^{-(K+1)/2}$:

$$\begin{aligned} q(\mathbf{V}) &\propto \int q(\mathbf{\Lambda_V})\, q(\mathbf{V} \mid \mathbf{\Lambda_V})\, \mathrm{d}\mathbf{\Lambda_V} \\ &= \int |\mathbf{\Lambda_V}|^{-(K+1)/2}\, \mathcal{N}^{-1}(\mathrm{vec}(\mathbf{V}); \mathbf{0}, \mathbf{I} \otimes \mathbf{\Lambda_V})\, \mathrm{d}\mathbf{\Lambda_V} \\ &= \int |\mathbf{\Lambda_V}|^{(D-K-1)/2} \exp\left(-\frac{1}{2} \operatorname{tr} \mathbf{V}\mathbf{V}^T \mathbf{\Lambda_V}\right) \mathrm{d}\mathbf{\Lambda_V} \\ &\propto \left|\mathbf{V}\mathbf{V}^T\right|^{-D/2}. \end{aligned} \tag{A.1}$$

The integral in the last step is based on the normalizing constant for the Wishart distribution (Bishop, 2006, app. B). Similarly,

$$p(\mathbf{V} \mid r, \mathbf{z}) \propto \left|\mathbf{V}\boldsymbol{S}\mathbf{V}^T\right|^{-D/2}, \tag{A.2}$$

where $\boldsymbol{S}$ is defined in (6.20).

Unfortunately, under this prior, the *posterior* is improper, with infinite mass on

large values of $\mathbf{\Lambda_V}$. Instead, we use the proper distribution

$$p_{wi}(\mathbf{\Lambda_V}) \propto \begin{cases} |\mathbf{\Lambda_V}|^{-(K+1)/2} & \text{if } \epsilon\mathbf{I} \preceq \mathbf{\Lambda_V} \preceq \epsilon^{-1}\mathbf{I} \\ 0 & \text{otherwise,} \end{cases} \tag{A.3}$$

for small ϵ, which agrees with the uninformative prior for non-extreme values of $\mathbf{\Lambda_V}$, but does not have this degeneracy. For the remainder of the discussion, we use the approximations (A.1) and (A.2).

A.2 Model symmetries

First, note that $p(\mathbf{U}) = q(\mathbf{U})$ and $p(\mathbf{Y} \mid \mathbf{U}, \mathbf{V}) = q(\mathbf{Y} \mid \mathbf{U}, \mathbf{V})$. Also, using (A.1) and (A.2),

$$\begin{aligned} \frac{p(\mathbf{T}_2\mathbf{V} \mid r, \mathbf{z})}{p(\mathbf{V} \mid r, \mathbf{z})} &= \frac{\left|\mathbf{T}_2\mathbf{V}\boldsymbol{S}\mathbf{V}^T\mathbf{T}_2^T\right|^{-D/2}}{|\mathbf{V}\boldsymbol{S}\mathbf{V}^T|^{-D/2}} \\ &= |\mathbf{T}_2|^{-D} \\ &= \frac{\left|\mathbf{T}_2\mathbf{V}\mathbf{V}^T\mathbf{T}_2^T\right|^{-D/2}}{|\mathbf{V}\mathbf{V}^T|^{-D/2}} \\ &= \frac{q(\mathbf{T}_2\mathbf{V})}{q(\mathbf{V})} \end{aligned} \tag{A.4}$$

Then,

$$\begin{aligned} p(\mathbf{T}_2\mathbf{V}) &= \sum_{\mathbf{z}} \int p(r)\, p(\mathbf{z})\, p(\mathbf{T}_2\mathbf{V} \mid r, \mathbf{z})\, \mathrm{d}r \\ &= \frac{q(\mathbf{T}_2\mathbf{V})}{q(\mathbf{V})} \sum_{\mathbf{z}} \int p(r)\, p(\mathbf{z})\, p(\mathbf{V} \mid r, \mathbf{z})\, \mathrm{d}r \\ &= \frac{q(\mathbf{T}_2\mathbf{V})}{q(\mathbf{V})}\, p(\mathbf{V}). \end{aligned} \tag{A.5}$$

Therefore,

$$\begin{aligned}\frac{p(\mathbf{U}\mathbf{T}_1, \mathbf{T}_2\mathbf{V} \,|\, \mathbf{Y})}{p(\mathbf{U}, \mathbf{V} \,|\, \mathbf{Y})} &= \frac{p(\mathbf{U}\mathbf{T}_1)\, p(\mathbf{T}_2\mathbf{V})\, p(\mathbf{Y} \,|\, \mathbf{U}\mathbf{T}_1, \mathbf{T}_2\mathbf{V})}{p(\mathbf{U})\, p(\mathbf{V})\, p(\mathbf{Y} \,|\, \mathbf{U}, \mathbf{V})} \\ &= \frac{q(\mathbf{U}\mathbf{T}_1)\, q(\mathbf{T}_2\mathbf{V})\, q(\mathbf{Y} \,|\, \mathbf{U}\mathbf{T}_1, \mathbf{T}_2\mathbf{V})}{q(\mathbf{U})\, q(\mathbf{V})\, q(\mathbf{Y} \,|\, \mathbf{U}, \mathbf{V})} \\ &= \frac{q(\mathbf{U}\mathbf{T}_1, \mathbf{T}_2\mathbf{V} \,|\, \mathbf{Y})}{q(\mathbf{U}, \mathbf{V} \,|\, \mathbf{Y})}. \end{aligned} \tag{A.6}$$

A.3 Second-order Taylor approximation

A.3.1 Priors

We first compute the second-order approximation to $\log q(\mathbf{V})$.

$$\begin{aligned}\mathrm{d} \log \left|\mathbf{V}\mathbf{V}^T\right| &= \operatorname{tr}\left(\mathbf{V}\mathbf{V}^T\right)^{-1} \mathrm{d}\left(\mathbf{V}\mathbf{V}^T\right) \\ &= 2 \operatorname{tr}\left(\mathbf{V}\mathbf{V}^T\right)^{-1} \left(\mathbf{V}\mathrm{d}\mathbf{V}^T\right) \\ &= \mathbf{0} \qquad \text{(A.7)} \\ \mathrm{d}^2 \log \left|\mathbf{V}\mathbf{V}^T\right| &= 2 \operatorname{tr} \mathrm{d}(\mathbf{V}\mathbf{V}^T)^{-1}(\mathbf{V}\mathrm{d}\mathbf{V}^T) + 2 \operatorname{tr}(\mathbf{V}\mathbf{V}^T)^{-1}\mathrm{d}(\mathbf{V}\mathrm{d}\mathbf{V}^T) \\ &= -2 \operatorname{tr}(\mathbf{V}\mathbf{V}^T)^{-1}(\mathbf{V}\mathrm{d}\mathbf{V}^T + \mathrm{d}\mathbf{V}\mathbf{V}^T)(\mathbf{V}\mathbf{V}^T)^{-1}(\mathbf{V}\mathrm{d}\mathbf{V}^T) + 2 \operatorname{tr}(\mathbf{V}\mathbf{V}^T)^{-1}\mathrm{d}\mathbf{V}\mathrm{d}\mathbf{V}^T \\ &= 2 \operatorname{tr}(\mathbf{V}\mathbf{V}^T)^{-1}\mathrm{d}\mathbf{V}\mathrm{d}\mathbf{V}^T, \qquad \text{(A.8)}\end{aligned}$$

where steps (A.7) and (A.8) use the assumption that $\mathbf{V}\mathrm{d}\mathbf{V}^T = \mathbf{0}$ (Section 6.2.4.2). In Kronecker product notation, this can be written:

$$\mathrm{d}^2 \log \left|\mathbf{V}\mathbf{V}^T\right| = 2 \operatorname{vec}(\mathrm{d}\mathbf{V})^T \left(\mathbf{I} \otimes (\mathbf{V}\mathbf{V}^T)^{-1}\right) \operatorname{vec}(\mathrm{d}\mathbf{V}).$$

For values of $\mathbf{V}$ such that $\mathbf{\Lambda}_\mathbf{V}$ is far from the constraint boundary, we have $\log q(\mathbf{V}) \approx -\frac{D}{2} \log \left|\mathbf{V}\mathbf{V}^T\right|$ (see (A.1)). Therefore,

$$\log q(\mathbf{V}) \approx \mathcal{N}^{-1}\left(\operatorname{vec}(\mathbf{V});\, \mathbf{0},\, \mathbf{I} \otimes D\left(\mathbf{V}\mathbf{V}^T\right)^{-1}\right). \tag{A.9}$$

The prior over $\mathbf{V}$ is given by:

$$\begin{aligned} p(\mathbf{V} \,|\, \boldsymbol{\eta}) &= \mathcal{N}^{-1}(\mathrm{vec}(\mathbf{V}); \mathbf{0}, \boldsymbol{S} \otimes \boldsymbol{\Lambda}_{\mathbf{V}}) \\ &\propto \exp\left(-\frac{1}{2} \operatorname{tr} \mathbf{V}\boldsymbol{S}\mathbf{V}^T\boldsymbol{\Lambda}_{\mathbf{V}}\right) \\ &= \exp\left(-\frac{1}{2} \operatorname{tr} \mathbf{V}_0\boldsymbol{S}\mathbf{V}_0^T\boldsymbol{\Lambda}_{\mathbf{V}} - \operatorname{tr} \mathbf{V}_0\boldsymbol{S}\mathbf{V}_\perp^T\boldsymbol{\Lambda}_{\mathbf{V}} - \frac{1}{2}\mathbf{V}_\perp\boldsymbol{S}\mathbf{V}_\perp^T\boldsymbol{\Lambda}_{\mathbf{V}}\right) \\ &\propto \exp\left(-\operatorname{vec}\left(\boldsymbol{\Lambda}_{\mathbf{V}}\mathbf{V}_0\boldsymbol{S}\right)^T \operatorname{vec}(\mathbf{V}_\perp) - \frac{1}{2}\operatorname{vec}(\mathbf{V}_\perp)^T\left(\boldsymbol{S} \otimes \boldsymbol{\Lambda}_{\mathbf{V}}\right)\operatorname{vec}(\mathbf{V}_\perp)\right). \end{aligned} \tag{A.10}$$

A.3.2 Observation term

We compute the first and second derivatives of $f(\mathbf{U}, \mathbf{V}) = \frac{1}{2}\left\|\mathbf{UV} - \mathbf{Y}\right\|^2$:

$$\begin{aligned} \mathrm{d}f &= \operatorname{tr}\left(\mathbf{UV} - \mathbf{Y}\right)^T \mathrm{d}\left(\mathbf{UV} - \mathbf{Y}\right) \\ &= \operatorname{tr}\left(\mathbf{UV} - \mathbf{Y}\right)^T \left(\mathbf{U}\mathrm{d}\mathbf{V} + \mathrm{d}\mathbf{U}\mathbf{V}\right) \\ &= \operatorname{tr} \mathbf{R}^T\mathbf{U}\mathrm{d}\mathbf{V} + \mathbf{R}^T\mathrm{d}\mathbf{U}\mathbf{V} \end{aligned} \tag{A.11}$$

$$\begin{aligned} \mathrm{d}^2 f &= \operatorname{tr} \mathrm{d}(\mathbf{UV} - \mathbf{Y})^T \mathrm{d}(\mathbf{UV} - \mathbf{Y}) + \operatorname{tr}\left(\mathbf{UV} - \mathbf{Y}\right)^T \mathrm{d}^2(\mathbf{UV} - \mathbf{Y}) \\ &= \operatorname{tr}\left(\mathrm{d}\mathbf{U}\mathbf{V} + \mathbf{U}\mathrm{d}\mathbf{V}\right)^T \left(\mathrm{d}\mathbf{U}\mathbf{V} + \mathbf{U}\mathrm{d}\mathbf{V}\right) + \operatorname{tr}\left(\mathbf{UV} - \mathbf{Y}\right)^T \mathrm{d}\mathbf{U}\mathrm{d}\mathbf{V} \\ &= \operatorname{tr}\left[\mathbf{V}^T\mathrm{d}\mathbf{U}^T\mathrm{d}\mathbf{U}\mathbf{V} + 2\mathrm{d}\mathbf{V}^T\mathbf{U}^T\mathrm{d}\mathbf{U}\mathbf{V} + \mathrm{d}\mathbf{V}^T\mathbf{U}^T\mathbf{U}\mathrm{d}\mathbf{V}\right] + \operatorname{tr}\mathbf{R}^T\mathrm{d}\mathbf{U}\mathrm{d}\mathbf{V} \\ &= \operatorname{tr}\left[\mathbf{V}^T\mathrm{d}\mathbf{U}^T\mathrm{d}\mathbf{U}\mathbf{V} + \mathrm{d}\mathbf{V}^T\mathbf{U}^T\mathbf{U}\mathrm{d}\mathbf{V}\right] + \operatorname{tr}\mathbf{R}^T\mathrm{d}\mathbf{U}\mathrm{d}\mathbf{V} \\ &= \left(\operatorname{vec}(\mathrm{d}\mathbf{U})^T \;\; \operatorname{vec}(\mathrm{d}\mathbf{V})^T\right) \begin{pmatrix} \mathbf{V}\mathbf{V}^T \otimes \mathbf{I} & \mathbf{F}(\mathbf{R} \otimes \mathbf{I}) \\ (\mathbf{R}^T \otimes \mathbf{I})\mathbf{F} & \mathbf{I} \otimes \mathbf{U}^T\mathbf{U} \end{pmatrix} \begin{pmatrix} \operatorname{vec}(\mathrm{d}\mathbf{U}) \\ \operatorname{vec}(\mathrm{d}\mathbf{V}) \end{pmatrix}. \end{aligned} \tag{A.12}$$

The second to last step uses the assumption that $\mathbf{V}\mathrm{d}\mathbf{V}^T = \mathbf{0}$. Therefore, the Taylor approximation is given by

$$\begin{aligned}
\log p_{\mathrm{so}}(\mathbf{Y} \mid \mathbf{U}, \mathbf{V}) &= \log p(\mathbf{Y} \mid \mathbf{U}_0, \mathbf{V}_0) - \lambda \operatorname{tr} \mathbf{R}^T \left(\mathbf{U}_0(\mathbf{V} - \mathbf{V}_0) + (\mathbf{U} - \mathbf{U}_0)\mathbf{V}_0\right) \\
&\quad - \frac{\lambda}{2} \left(\operatorname{vec}(\mathbf{U} - \mathbf{U}_0)^T \ \operatorname{vec}(\mathbf{V} - \mathbf{V}_0)^T\right) \begin{pmatrix} \mathbf{V}_0\mathbf{V}_0^T \otimes \mathbf{I} & \mathbf{F}(\mathbf{R} \otimes \mathbf{I}) \\ (\mathbf{R}^T \otimes \mathbf{I})\mathbf{F} & \mathbf{I} \otimes \mathbf{U}_0^T\mathbf{U}_0 \end{pmatrix} \begin{pmatrix} \operatorname{vec}(\mathbf{U} - \mathbf{U}_0) \\ \operatorname{vec}(\mathbf{V} - \mathbf{V}_0) \end{pmatrix} \\
&= \text{const} + \mathcal{Q} - \lambda \operatorname{tr} \mathbf{R}^T(\mathbf{U}_0\mathbf{V} + \mathbf{U}\mathbf{V}_0) \\
&\quad + \lambda \left(\operatorname{vec}(\mathbf{U})^T \operatorname{vec}(\mathbf{V})^T\right) \begin{pmatrix} \mathbf{V}_0\mathbf{V}_0^T \otimes \mathbf{I} & \mathbf{F}(\mathbf{R} \otimes \mathbf{I}) \\ (\mathbf{R}^T \otimes \mathbf{I})\mathbf{F} & \mathbf{I} \otimes \mathbf{U}_0^T\mathbf{U}_0 \end{pmatrix} \begin{pmatrix} \operatorname{vec}(\mathbf{U}_0) \\ \operatorname{vec}(\mathbf{V}_0) \end{pmatrix} \\
&= \text{const} + \mathcal{Q} + \lambda \operatorname{tr} \left(\mathbf{U}_0^T\mathbf{U}\mathbf{V}_0\mathbf{V}_0^T + \mathbf{U}_0^T\mathbf{U}_0\mathbf{V}\mathbf{V}_0^T\right) \\
&= \text{const} + \mathcal{Q} + \lambda \operatorname{tr} \left(\mathbf{U}_0^T\mathbf{U}\mathbf{V}_0\mathbf{V}_0^T\right) \qquad \text{(A.13)}
\end{aligned}$$

where

$$\begin{aligned}
\mathcal{Q} &= -\frac{\lambda}{2} \left(\operatorname{vec}(\mathbf{U})^T \operatorname{vec}(\mathbf{V})^T\right) \begin{pmatrix} \mathbf{V}_0\mathbf{V}_0^T \otimes \mathbf{I} & \mathbf{F}(\mathbf{R} \otimes \mathbf{I}) \\ (\mathbf{R}^T \otimes \mathbf{I})\mathbf{F} & \mathbf{I} \otimes \mathbf{U}_0^T\mathbf{U}_0 \end{pmatrix} \begin{pmatrix} \operatorname{vec}(\mathbf{U}) \\ \operatorname{vec}(\mathbf{V}) \end{pmatrix} \\
&= -\lambda \operatorname{tr} \mathbf{R}\mathbf{V}_0^T\mathbf{U}^T - \frac{\lambda}{2} \left(\operatorname{vec}(\mathbf{U})^T \operatorname{vec}(\mathbf{V}_\perp)^T\right) \begin{pmatrix} \mathbf{V}_0\mathbf{V}_0^T \otimes \mathbf{I} & \mathbf{F}(\mathbf{R} \otimes \mathbf{I}) \\ (\mathbf{R}^T \otimes \mathbf{I})\mathbf{F} & \mathbf{I} \otimes \mathbf{U}_0^T\mathbf{U}_0 \end{pmatrix} \begin{pmatrix} \operatorname{vec}(\mathbf{U}) \\ \operatorname{vec}(\mathbf{V}_\perp) \end{pmatrix}
\end{aligned}$$

A.4 Analyzing the Gaussian approximation

In this section, we compute the KL divergence between Gaussian approximations to the two posteriors. We begin with a few observations which are used repeatedly throughout this section. First, observe that

$$\mathbf{U}_0^T\mathbf{U}_0 \approx N\mathbf{I}. \qquad \text{(A.14)}$$

This is true because covariance can be "shifted" between $\mathbf{U}$ and $\mathbf{V}$ by taking $\mathbf{U} \mapsto \mathbf{U}\mathbf{T}$ and $\mathbf{V} \mapsto \mathbf{T}^{-1}\mathbf{V}$, where $\mathbf{T}$ is an invertible matrix. This transformation does not affect the observation term $p(\mathbf{Y} \mid \mathbf{U}, \mathbf{V})$. The priors $q(\mathbf{V})$ and $p(\mathbf{V})$ are invariant to such linear transformations, while $p(\mathbf{U}) = q(\mathbf{U})$ assumes a unit normal distribution. Therefore, the posterior should favor explanations where $\mathbf{U}^T\mathbf{U} \approx N\mathbf{I}$.

Next, $\mathbf{\Lambda}_\mathbf{V}$ should be close to its maximum likelihood solution, namely

$$\mathbf{\Lambda}_\mathbf{V}^{-1} \approx \frac{1}{D}\mathbf{V}_0\boldsymbol{S}\mathbf{V}_0^T. \qquad \text{(A.15)}$$

Finally, $\boldsymbol{S}^{-1} = r\mathbf{Z}\mathbf{Z}^T + \mathbf{I} \succeq \mathbf{I}$, so

$$\boldsymbol{S} \preceq \mathbf{I}. \tag{A.16}$$

Since $\boldsymbol{S}_\perp$ is the projection of $\boldsymbol{S}$ onto a subspace, we have

$$\boldsymbol{S}_\perp \preceq \mathbf{I} \tag{A.17}$$

as well.

Throughout this section, we use the shorthand

$$\begin{aligned} \mathbf{u} &= \mathrm{vec}(\mathbf{U}) \\ \mathbf{v} &= \mathrm{vec}(\tilde{\mathbf{V}}_\perp). \end{aligned} \tag{A.18}$$

A.4.1 Marginal covariance is approximately conditional covariance

In the Gaussian approximation to the posterior, $\mathbf{C}_p$ represents the conditional precision of $\mathbf{v}$ given $\mathbf{u}$, and $\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}$ represents the marginal precision. We now show that these two quantities are approximately equal. The derivation, and the result, are identical if $\mathbf{C}_q$ is substituted for $\mathbf{C}_p$. Let $\|\cdot\|_2$ denote the matrix 2-norm, *i.e.* the largest signular value.

$$\begin{aligned} \left\|\mathbf{I} - \mathbf{C}_p^{-1}(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B})\right\|_2 &= \left\|\mathbf{C}_p^{-1}\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right\|_2 \\ &\leq \left\|\mathbf{C}_p^{-1}\right\|_2 \left\|\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right\|_2 \\ &= \left\|\mathbf{C}_p^{-1}\right\|_2 \left\|\lambda^2(\mathbf{R}_\perp^T \otimes \mathbf{I})\mathbf{F}(\lambda\mathbf{V}_0{\mathbf{V}_0}^T \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{I})^{-1}\mathbf{F}(\mathbf{R}_\perp \otimes \mathbf{I})\right\| \\ &= \left\|\mathbf{C}_p^{-1}\right\|_2 \left\|\lambda^2\mathbf{R}_\perp^T\mathbf{R}_\perp \otimes (\lambda\mathbf{V}_0\mathbf{V}_0^T + \mathbf{I})^{-1}\right\|_2 \\ &= \lambda^2 \left\|\mathbf{C}_p^{-1}\right\|_2 \left\|\mathbf{R}_\perp^T\mathbf{R}_\perp\right\|_2 \left\|(\lambda\mathbf{V}_0\mathbf{V}_0^T + \mathbf{I})^{-1}\right\|_2 \end{aligned} \tag{A.19}$$

The first step uses the general fact that $\|\mathbf{X}\mathbf{Y}\| \leq \|\mathbf{X}\|\|\mathbf{Y}\|$ for any matrix norm, and the last step uses the general fact that $\|\mathbf{X} \otimes \mathbf{Y}\|_2 = \|\mathbf{X}\|_2\|\mathbf{Y}\|_2$. Now,

$$\begin{aligned} \left\|\mathbf{C}_p^{-1}\right\|_2 &= \left\|(\lambda\mathbf{I} \otimes \mathbf{U}_0^T\mathbf{U}_0 + \boldsymbol{S} \otimes \boldsymbol{\Lambda}_\mathbf{V})^{-1}\right\|_2 \\ &\leq \left\|(\lambda\mathbf{I} \otimes \mathbf{U}_0^T\mathbf{U}_0)^{-1}\right\|_2 \\ &= 1/\Theta(N) \end{aligned} \tag{A.20}$$

Using our assumption of uncorrelated residuals from Section 6.2.2,

$$\left\|\mathbf{R}_\perp^T \mathbf{R}_\perp\right\|_2 \leq \left\|\mathbf{R}^T \mathbf{R}\right\|_2 = \|\mathbf{R}\|_2^2 = O(N+D). \tag{A.21}$$

Finally, $\left\|(\lambda \mathbf{V}_0 \mathbf{V}_0^T + \mathbf{I})^{-1}\right\|_2 = 1/\Theta(D)$. Putting this together,

$$\left\|\mathbf{I} - \mathbf{C}_p^{-1}(\mathbf{C}_p - \mathbf{B}^T \mathbf{A}^{-1} \mathbf{B})\right\|_2 = O\left(\frac{1}{N} + \frac{1}{D}\right). \tag{A.22}$$

This implies that

$$(1-\epsilon)\mathbf{C}_p \preceq \mathbf{C}_p - \mathbf{B}^T \mathbf{A}^{-1} \mathbf{B} \preceq \mathbf{C}_p \tag{A.23}$$

where $\epsilon = O(\frac{1}{N} + \frac{1}{D})$. (The right-hand inequality holds because $\mathbf{B}^T \mathbf{A}^{-1} \mathbf{B}$ is positive semidefinite.) Hence, whenever $\mathbf{C}_p - \mathbf{B}^T \mathbf{A}^{-1} \mathbf{B}$ appears multiplicatively in a formula, we make the approximation

$$\mathbf{C}_p - \mathbf{B}^T \mathbf{A}^{-1} \mathbf{B} \approx \mathbf{C}_p. \tag{A.24}$$

A.4.2 Taylor approximation is Gaussian

In order for the second-order Taylor approximation to define a Gaussian, the matrices

$$\begin{aligned} \mathbf{\Lambda}_{q_{\text{post}}(\mathbf{u},\mathbf{v})} &= \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C}_q \end{pmatrix} \\ \mathbf{\Lambda}_{p_{\text{post}}(\mathbf{u},\mathbf{v})} &= \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C}_p \end{pmatrix} \end{aligned} \tag{A.25}$$

must be positive definite (PD). One characterization of PD matrices is that $\mathbf{\Lambda}_{p_{\text{post}}(\mathbf{u},\mathbf{v})}$ is PD if and only if $\mathbf{A}$ and $\mathbf{C}_p - \mathbf{B}^T \mathbf{A}^{-1} \mathbf{B}$ are both PD (Boyd and Vandenberghe, 2004, app. A.5.5). First, note that $\mathbf{A}$ and $\mathbf{C}_p$ are both PD, because they consist of sums and Kronecker products of PD matrices. Since $\mathbf{C}_p$ is PD, the left-hand inequality of (A.23) implies that $\mathbf{C}_p - \mathbf{B}^T \mathbf{A}^{-1} \mathbf{B}$ is PD for large D. The proof also applies for $\mathbf{\Lambda}_{p_{\text{post}}(\mathbf{u},\mathbf{v})}$, with $\mathbf{C}_q$ substituted for $\mathbf{C}_p$.

A.4.3 KL divergence between two Gaussians

The approximate posteriors q_{post} and p_{post} are Gaussians with precision matrices defined in (A.25). First, note that because $q_{\text{post}}(\mathbf{u} \,|\, \mathbf{v}) = p_{\text{post}}(\mathbf{u} \,|\, \mathbf{v})$,

$$\mathrm{D}_{\mathrm{KL}}(q_{\text{post}}(\mathbf{u}, \mathbf{v}) \,\|\, p_{\text{post}}(\mathbf{u}, \mathbf{v})) = \mathrm{D}_{\mathrm{KL}}(q_{\text{post}}(\mathbf{v}) \,\|\, p_{\text{post}}(\mathbf{v})). \tag{A.26}$$

(The vectors $\mathbf{u}$ and $\mathbf{v}$ are defined in (A.18).) Therefore, we focus on the marginal distribution of $\mathbf{v}$.

The marginal precisions of $\mathbf{v}$ are obtained from the Schur complement formula:

$$\begin{aligned} \boldsymbol{\Lambda}_{q_{\text{post}}(\mathbf{v})} &= \mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B} \\ \boldsymbol{\Lambda}_{p_{\text{post}}(\mathbf{v})} &= \mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B} \end{aligned} \tag{A.27}$$

Plugging these into the PDF of a multivariate Gaussian,

$$\begin{aligned} \mathbb{E}_{q_{\text{post}}}[\log q(\mathbf{v})] &= -\frac{d}{2}\log 2\pi + \frac{1}{2}\log\left|\mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right| - \frac{1}{2}\mathbb{E}_{q_{\text{post}}}\left[(\mathbf{v}-\boldsymbol{\mu}_q)^T(\mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B})(\mathbf{v}-\boldsymbol{\mu}_q)\right] \\ &= -\frac{d}{2}\log 2\pi + \frac{1}{2}\log\left|\mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right| - \frac{1}{2}\operatorname{tr}\boldsymbol{\Sigma}_{\mathbf{v}}(\mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}) \\ \mathbb{E}_{q_{\text{post}}}[\log p(\mathbf{v})] &= -\frac{d}{2}\log 2\pi + \frac{1}{2}\log\left|\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right| - \frac{1}{2}\mathbb{E}_{q_{\text{post}}}\left[(\mathbf{v}-\boldsymbol{\mu}_p)^T(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B})(\mathbf{v}-\boldsymbol{\mu}_p)\right] \\ &= -\frac{d}{2}\log 2\pi + \frac{1}{2}\log\left|\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right| - \frac{1}{2}\operatorname{tr}\boldsymbol{\Sigma}_{\mathbf{v}}(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}) \\ &\quad - \frac{1}{2}\boldsymbol{\Delta}_{\boldsymbol{\mu}}^T(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B})\boldsymbol{\Delta}_{\boldsymbol{\mu}}, \end{aligned} \tag{A.28}$$

where d is the dimensionality of $\mathbf{v}$, $\boldsymbol{\Sigma}_{\mathbf{v}} = (\mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B})^{-1}$, and $\boldsymbol{\Delta}_{\boldsymbol{\mu}} = \mathbb{E}_{q_{\text{post}}}[\mathbf{v}] - \mathbb{E}_{p_{\text{post}}}[\mathbf{v}]$. Therefore,

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}\left(q_{\text{post}}(\mathbf{v}) \,\|\, p_{\text{post}}(\mathbf{v})\right) &= \mathbb{E}_{q_{\text{post}}}[\log q(\mathbf{v})] - \mathbb{E}_{q_{\text{post}}}[\log p(\mathbf{u})] \\ &= \mathcal{T}_1 + \mathcal{T}_2 + \mathcal{T}_3, \end{aligned} \tag{A.29}$$

where

$$\begin{aligned} \mathcal{T}_1 &= \frac{1}{2}\log\left|\mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right| - \frac{1}{2}\log\left|\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right| \\ \mathcal{T}_2 &= \frac{1}{2}\operatorname{tr}\boldsymbol{\Sigma}_{\mathbf{v}}(\mathbf{C}_p - \mathbf{C}_q) \\ \mathcal{T}_3 &= \frac{1}{2}\boldsymbol{\Delta}_{\boldsymbol{\mu}}^T(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B})\boldsymbol{\Delta}_{\boldsymbol{\mu}}. \end{aligned} \tag{A.30}$$

A.4.4 The volume term $\mathcal{T}_1$

$$
\begin{aligned}
\mathcal{T}_1 &= \frac{1}{2}\log\left|\mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right| - \frac{1}{2}\log\left|\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right| \\
&= \frac{1}{2}\log\left|\left(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\left(\mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)\right| \\
&= \frac{1}{2}\log\left|\mathbf{I} + \left(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\left(\mathbf{C}_q - \mathbf{C}_p\right)\right| \\
&\leq \frac{1}{2}\operatorname{tr}\left(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\left(\mathbf{C}_q - \mathbf{C}_p\right) && \text{(A.31)} \\
&\approx \frac{1}{2}\operatorname{tr}\mathbf{C}_p^{-1}(\mathbf{C}_q - \mathbf{C}_p) && \text{(A.32)} \\
&= \frac{1}{2}\operatorname{tr}\mathbf{C}_p^{-1}\left(\mathbf{I}\otimes D(\mathbf{V}_0\mathbf{V}_0^T)^{-1} - \boldsymbol{S}_\perp\otimes\boldsymbol{\Lambda}_\mathbf{V}\right) \\
&\leq \frac{1}{2}\operatorname{tr}\left(\lambda\mathbf{I}\otimes\mathbf{U}_0^T\mathbf{U}_0\right)^{-1}\left(\mathbf{I}\otimes D(\mathbf{V}_0\mathbf{V}_0^T)^{-1}\right) \\
&= \frac{1}{2\lambda}D\operatorname{tr}\mathbf{I}\ \operatorname{tr}(\mathbf{U}_0^T\mathbf{U}_0)^{-1}(\mathbf{V}_0\mathbf{V}_0^T)^{-1} \\
&\approx \frac{1}{2\lambda N}D\operatorname{tr}\mathbf{I}\ \operatorname{tr}(\mathbf{V}_0\mathbf{V}_0^T)^{-1} && \text{(A.33)} \\
&= \frac{1}{2\lambda N}D(D-K)\operatorname{tr}(\mathbf{V}_0\mathbf{V}_0^T)^{-1} \\
&\leq \frac{1}{2\lambda N}D(D-K)\operatorname{tr}(\mathbf{V}_0\boldsymbol{S}\mathbf{V}_0^T)^{-1} && \text{(A.34)} \\
&\approx \frac{1}{2\lambda N}(D-K)\operatorname{tr}\boldsymbol{\Lambda}_\mathbf{V} && \text{(A.35)}
\end{aligned}
$$

Step (A.31) holds because

$$\log\det\mathbf{I}+\mathbf{D} = \sum_i \log\nu_i + 1 \leq \sum_i \nu_i = \operatorname{tr}\mathbf{D}$$

for any square matrix $\mathbf{D}$, where the ν_i denote the eigenvalues of $\mathbf{D}$. (A.32) follows from (A.23), (A.33) follows from (A.14), (A.34) follows from (A.16), and (A.35) follows from (A.15). The remaining steps are simple substitutions and algebraic manipulations.

A.4.5 The variance term $\mathcal{T}_2$

$$
\begin{aligned}
\mathcal{T}_2 &= \frac{1}{2}\operatorname{tr}(\mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B})^{-1}(\mathbf{C}_p - \mathbf{C}_q) \\
&\approx \frac{1}{2}\operatorname{tr}\mathbf{C}_q{}^{-1}(\mathbf{C}_p - \mathbf{C}_q) && \text{(A.36)} \\
&= \frac{1}{2}\operatorname{tr}\mathbf{C}_q{}^{-1}\left(\boldsymbol{S}_\perp \otimes \boldsymbol{\Lambda}_\mathbf{V} - \mathbf{I} \otimes D(\mathbf{V}_0\mathbf{V}_0^T)^{-1}\right) \\
&\leq \frac{1}{2}\operatorname{tr}\left(\lambda\mathbf{I} \otimes (\mathbf{U}_0^T\mathbf{U}_0)\right)^{-1}(\boldsymbol{S}_\perp \otimes \boldsymbol{\Lambda}_\mathbf{V}) \\
&= \frac{1}{2\lambda}\operatorname{tr}\boldsymbol{S}_\perp \otimes (\mathbf{U}_0^T\mathbf{U}_0)^{-1}\boldsymbol{\Lambda}_\mathbf{V} \\
&= \frac{1}{2\lambda}\operatorname{tr}\boldsymbol{S}_\perp \ \operatorname{tr}(\mathbf{U}_0^T\mathbf{U}_0)^{-1}\boldsymbol{\Lambda}_\mathbf{V} \\
&\leq \frac{1}{2\lambda}(D-K)\operatorname{tr}(\mathbf{U}_0^T\mathbf{U}_0)^{-1}\boldsymbol{\Lambda}_\mathbf{V} && \text{(A.37)} \\
&\approx \frac{1}{2\lambda N}(D-K)\operatorname{tr}\boldsymbol{\Lambda}_\mathbf{V} && \text{(A.38)}
\end{aligned}
$$

(A.36) follows from (A.23), (A.37) follows from (A.17), and (A.38) follows from (A.14). The remaining steps are simple substitutions and algebraic manipulations.

A.4.6 The bias term $\mathcal{T}_3$

The difference in means of the two posteriors is given by:

$$
\begin{aligned}
\mathbb{E}_{p_{\text{post}}}[\mathbf{v}] - \mathbb{E}_{q_{\text{post}}}[\mathbf{v}] &= \left(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\left(\mathbf{h}_\mathbf{v} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{h}_\mathbf{u}\right) - \left(\mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\left(-\mathbf{B}^T\mathbf{A}^{-1}\mathbf{h}_\mathbf{u}\right) \\
&= \left(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{h}_\mathbf{v} + \left[\left(\mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1} - \left(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\right]\mathbf{B}^T\mathbf{A}^{-1}\mathbf{h}_\mathbf{u} \\
&= \left(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\left[\mathbf{h}_\mathbf{v} + \left(\mathbf{C}_p - \mathbf{C}_q\right)\left(\mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{A}^{-1}\mathbf{h}_\mathbf{u}\right] \\
&\triangleq \left(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\left[\mathbf{h}_\mathbf{v} + \mathbf{h}_{\mathbf{u}\to\mathbf{v}}\right].
\end{aligned}
$$

Now,

$$\begin{aligned}
\sqrt{\mathcal{T}_3} &= \sqrt{\frac{1}{2}(\mathbb{E}_{p_{\text{post}}}[\mathbf{v}] - \mathbb{E}_{q_{\text{post}}}[\mathbf{v}])^T \left(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)(\mathbb{E}_{p_{\text{post}}}[\mathbf{v}] - \mathbb{E}_{q_{\text{post}}}[\mathbf{v}])} \\
&= \sqrt{\frac{1}{2}(\mathbf{h_v} + \mathbf{h_{u\to v}})^T \left(\mathbf{C}_p - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1}(\mathbf{h_v} + \mathbf{h_{u\to v}})} \\
&\approx \sqrt{\frac{1}{2}(\mathbf{h_v} + \mathbf{h_{u\to v}})^T \mathbf{C}_p^{-1}(\mathbf{h_v} + \mathbf{h_{u\to v}})} \qquad \text{(A.39)} \\
&\leq \sqrt{\frac{1}{2}\mathbf{h_v}^T\mathbf{C}_p^{-1}\mathbf{h_v}} + \sqrt{\frac{1}{2}\mathbf{h}_{\mathbf{u}\to\mathbf{v}}^T\mathbf{C}_p^{-1}\mathbf{h}_{\mathbf{u}\to\mathbf{v}}}, \qquad \text{(A.40)}
\end{aligned}$$

where (A.39) follows from (A.23), and (A.40) holds because $\sqrt{\mathbf{x}^T\mathbf{D}\mathbf{x}}$ is a norm on $\mathbf{x}$ when $\mathbf{D}$ is positive definite.

We take these two terms in turn. For the first term,

$$\begin{aligned}
\mathbf{h_v}^T\mathbf{C}_p^{-1}\mathbf{h_v} &= \operatorname{vec}\left(\mathbf{\Lambda_V V}_0\boldsymbol{S}\mathbf{Q}^T\right)^T\left(\lambda\mathbf{I}\otimes\mathbf{U}_0^T\mathbf{U}_0 + \boldsymbol{S}_\perp\otimes\mathbf{\Lambda_V}\right)^{-1}\operatorname{vec}\left(\mathbf{\Lambda_V V}_0\boldsymbol{S}\mathbf{Q}^T\right) \\
&\leq \frac{1}{\lambda}\operatorname{vec}\left(\mathbf{\Lambda_V V}_0\boldsymbol{S}\mathbf{Q}^T\right)^T\left(\mathbf{I}\otimes\mathbf{U}_0^T\mathbf{U}_0\right)^{-1}\operatorname{vec}\left(\mathbf{\Lambda_V V}_0\boldsymbol{S}\mathbf{Q}^T\right) \\
&= \frac{1}{\lambda}\operatorname{tr}\left(\mathbf{U}_0^T\mathbf{U}_0\right)^{-1}\mathbf{\Lambda_V V}_0\boldsymbol{S}\mathbf{Q}^T\mathbf{Q}\boldsymbol{S}\mathbf{V}_0^T\mathbf{\Lambda_V} \\
&\leq \frac{1}{\lambda}\operatorname{tr}\left(\mathbf{U}_0^T\mathbf{U}_0\right)^{-1}\mathbf{\Lambda_V V}_0\boldsymbol{S}\mathbf{V}_0^T\mathbf{\Lambda_V} \qquad \text{(A.41)} \\
&\approx \frac{1}{\lambda N}D\operatorname{tr}\mathbf{\Lambda_V}. \qquad \text{(A.42)}
\end{aligned}$$

Here, (A.41) holds because $\mathbf{Q}^T\mathbf{Q} \preceq \mathbf{I}$ (the rows of $\mathbf{Q}$ are orthogonal) and because of (A.16). (A.42) follows from (A.14) and (A.15).

For the second term,

$$\begin{aligned}
\mathbf{h_{u\to v}} &= (\mathbf{C}_p - \mathbf{C}_q)\left(\mathbf{C}_q - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{A}^{-1}\mathbf{h_u} \\
&\approx (\mathbf{C}_p - \mathbf{C}_q)\mathbf{C}_q^{-1}\mathbf{B}^T\mathbf{A}^{-1}\mathbf{h_u} \\
&= \left(\boldsymbol{S}_\perp\otimes\mathbf{\Lambda_V} - \mathbf{I}\otimes D(\mathbf{V}_0\mathbf{V}_0^T)^{-1}\right)\left(\mathbf{I}\otimes\mathbf{U}_0^T\mathbf{U}_0\right)^{-1}\left(\mathbf{R}_\perp^T\otimes\mathbf{I}\right)\mathbf{F}\left(\mathbf{V}_0\mathbf{V}_0^T\otimes\mathbf{I}\right)^{-1}\operatorname{vec}(\mathbf{Y}\mathbf{V}_0{}^T) \\
&= \left(\boldsymbol{S}_\perp\otimes\mathbf{\Lambda_V} - \mathbf{I}\otimes D(\mathbf{V}_0\mathbf{V}_0^T)^{-1}\right)\left(\mathbf{I}\otimes\mathbf{U}_0^T\mathbf{U}_0\right)^{-1}\left(\mathbf{I}\otimes\mathbf{V}_0\mathbf{V}_0^T\right)^{-1}\operatorname{vec}(\mathbf{V}_0\mathbf{Y}^T\mathbf{R}_\perp)
\end{aligned}$$

Let $\|\cdot\|$ denote the Euclidean norm, $\|\cdot\|_2$ the matrix 2-norm (largest singular

value) and $\|\cdot\|_F$ the Frobenius norm.

$$\begin{aligned}\|\mathbf{h}_{\mathbf{u}\to\mathbf{v}}\| &\leq \left\|\boldsymbol{S}_\perp \otimes \boldsymbol{\Lambda}_\mathbf{V} - \mathbf{I}\otimes D(\mathbf{V}_0\mathbf{V}_0^T)^{-1}\right\|_2 \left\|\mathbf{I}\otimes(\mathbf{U}_0^T\mathbf{U}_0)^{-1}\right\|_2 \left\|\mathbf{I}\otimes(\mathbf{V}_0\mathbf{V}_0^T)^{-1}\right\|_2 \left\|\mathrm{vec}(\mathbf{V}_0\mathbf{Y}^T\mathbf{R}_\perp)\right\| \\ &= \left\|\boldsymbol{S}_\perp \otimes \boldsymbol{\Lambda}_\mathbf{V} - \mathbf{I}\otimes D(\mathbf{V}_0\mathbf{V}_0^T)^{-1}\right\|_2 \left\|(\mathbf{U}_0^T\mathbf{U}_0)^{-1}\right\|_2 \left\|(\mathbf{V}_0\mathbf{V}_0^T)^{-1}\right\|_2 \left\|\mathbf{V}_0\mathbf{Y}^T\mathbf{R}_\perp\right\|_F\end{aligned}$$

Taking these factors individually,

$$\begin{aligned}\left\|\boldsymbol{S}_\perp \otimes \boldsymbol{\Lambda}_\mathbf{V} - \mathbf{I}\otimes D(\mathbf{V}_0\mathbf{V}_0^T)^{-1}\right\|_2 &= O(1) \\ \left\|(\mathbf{U}_0^T\mathbf{U}_0)^{-1}\right\|_2 &= O(1/N) \\ \left\|(\mathbf{V}_0\mathbf{V}_0^T)^{-1}\right\|_2 &= O(1/D) \\ \left\|\mathbf{V}_0\mathbf{Y}^T\mathbf{R}_\perp\right\|_F &= \sqrt{\mathrm{tr}\,\mathbf{V}_0\mathbf{Y}^T\mathbf{R}_\perp\mathbf{R}_\perp^T\mathbf{Y}\mathbf{V}_0^T} \\ &\leq \nu(N+D)^{1/2}\sqrt{\mathrm{tr}\,\mathbf{V}_0\mathbf{Y}^T\mathbf{Y}\mathbf{V}_0^T} \qquad \text{(A.43)} \\ &= \nu(N+D)^{1/2}\left\|\mathbf{V}_0\mathbf{Y}^T\right\|_F \\ &\leq \nu(N+D)^{1/2}\left\|\mathbf{V}_0\right\|_F\left\|\mathbf{Y}^T\right\|_F \\ &= O(N^{1/2}D^{3/2}+ND)\end{aligned}$$

where ν is a constant that does not depend on N or D. (A.43) follows from the assumption of uncorrelated residuals in Section 6.2.2, the reasoning of (A.21), and the observation that $\mathbf{D}\preceq\|\mathbf{D}\|_2\mathbf{I}$ for a symmetric matrix $\mathbf{D}$. Combining these factors, $\|\mathbf{h}_{\mathbf{u}\to\mathbf{v}}\| = O(N^{-1/2}D^{1/2}+1)$. Therefore,

$$\begin{aligned}\mathbf{h}_{\mathbf{u}\to\mathbf{v}}^T\mathbf{C}_p^{-1}\mathbf{h}_{\mathbf{u}\to\mathbf{v}} &\leq \left\|\mathbf{C}_p^{-1}\right\|_2\left\|\mathbf{h}_{\mathbf{u}\to\mathbf{v}}\right\|^2 \\ &\leq \left\|\mathbf{I}\otimes(\mathbf{U}_0^T\mathbf{U}_0)^{-1}\right\|_2\left\|\mathbf{h}_{\mathbf{u}\to\mathbf{v}}\right\|^2 \\ &= O(N^{-2}D+N^{-1}).\end{aligned}$$

This term is smaller than the first term (which is $\Theta(N^{-1}D)$), so we drop it. We have, therefore, that

$$\mathcal{T}_3 \leq \frac{1}{2\lambda N}D\,\mathrm{tr}\,\boldsymbol{\Lambda}_\mathbf{V} + o(N^{-1}D).$$

Appendix B

Moment averaging derivations

B.1 Asymptotics of AIS

Theorem 4. *Suppose $T+1$ distributions p_t are linearly spaced along a path γ. Assuming perfect transitions, if $\boldsymbol{\theta}(\beta)$ and the Fisher information matrix $\mathbf{G}_{\boldsymbol{\theta}}(\beta) = \mathrm{cov}_{\mathbf{x}\sim p_{\boldsymbol{\theta}}}(\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}))$ are continuous and piecewise smooth, then as $T \to \infty$ the bias δ behaves as follows:*

$$T\delta = T \sum_{t=0}^{T-1} \mathrm{D}_{\mathrm{KL}}(p_t \| p_{t+1}) \to \mathcal{F}(\gamma) \equiv \frac{1}{2} \int_0^1 \dot{\boldsymbol{\theta}}(\beta)^T \mathbf{G}_{\boldsymbol{\theta}}(\beta) \dot{\boldsymbol{\theta}}(\beta) \, \mathrm{d}\beta, \tag{B.1}$$

where $\dot{\boldsymbol{\theta}}(\beta)$ represents the derivative of $\boldsymbol{\theta}$ with respect to β.

Proof. First, assume that $\boldsymbol{\theta}(\beta)$ and $\mathbf{G}_{\boldsymbol{\theta}}(\beta)$ are both smooth. Consider a second-order Taylor expansion of $\mathrm{D}_{\mathrm{KL}}(\boldsymbol{\theta}(\beta) \| \boldsymbol{\theta}(\beta + h))$ around $h = 0$. The constant and first order terms are zero. For the second order term,

$$\nabla^2_{\boldsymbol{\theta}} \mathrm{D}_{\mathrm{KL}}(\boldsymbol{\theta} \| \boldsymbol{\theta}_0)\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0} = \mathbf{G}_{\boldsymbol{\theta}},$$

so the second-order Taylor expansion is given by:

$$\mathrm{D}_{\mathrm{KL}}(\boldsymbol{\theta}(\beta) \| \boldsymbol{\theta}(\beta + h)) = \frac{1}{2} h^2 \dot{\boldsymbol{\theta}}^T(\beta) \mathbf{G}_{\boldsymbol{\theta}}(\beta) \dot{\boldsymbol{\theta}}(\beta) + \epsilon,$$

where

$$|\epsilon| \leq \frac{h^3}{6} \max_{\beta} \left| \frac{d^3}{dh^3} \mathrm{D}_{\mathrm{KL}}(\boldsymbol{\theta}(\beta) \| \boldsymbol{\theta}(\beta + h)) \right|.$$

(The maximum is finite because $\mathbf{G}_{\boldsymbol{\theta}}$ is smooth.)

Assuming a linear schedule, the bias is given by

$$\begin{aligned}\delta &= \sum_{t=0}^{T-1} \mathrm{D}_{\mathrm{KL}}(p_t \| p_{t+1}) \\ &= \sum_{t=0}^{T-1} \mathrm{D}_{\mathrm{KL}}(\boldsymbol{\theta}(t/T) \parallel \boldsymbol{\theta}((t+1)/T)) \\ &= \frac{1}{2T^2} \sum_{t=0}^{T-1} \dot{\boldsymbol{\theta}}(\beta_t)^T \mathbf{G}_{\boldsymbol{\theta}}(\beta_t) \dot{\boldsymbol{\theta}}(\beta_t) + \sum_{t=0}^{T-1} \epsilon_t\end{aligned}$$

The error term decays like $1/T^2$, so it approaches zero even when scaled by T. The asymptotic bias, therefore, is determined by the first term. When scaled by T, this approaches

$$\mathcal{F}(\gamma) \equiv \frac{1}{2} \int_0^1 \dot{\boldsymbol{\theta}}(\beta)^T \mathbf{G}_{\boldsymbol{\theta}}(\beta) \dot{\boldsymbol{\theta}}(\beta) \, \mathrm{d}\beta.$$

Therefore, $T\delta \to \mathcal{F}(\gamma)$.

In the above analysis, we assumed that $\boldsymbol{\theta}(\beta)$ and $\mathbf{G}_{\boldsymbol{\theta}}(\beta)$ were smooth. If they are merely piecewise smooth, the integral decomposes into sums over the smooth segments of γ. Similarly, the KL divergence terms corresponding to non-smooth points decay like $1/T^2$, so they approach zero when scaled by T. Ignoring these terms, the bias decomposes as a sum over the smooth segments of γ, so the theorem holds in the piecewise smooth case as well. □

B.2 Variational interpretations of geometric and moment averages

B.2.1 Geometric averages

For simplicity of notation, assume the state space $\mathcal{X}$ is discrete. Consider solving for a distribution q to minimize the weighted sum of KL divergences

$$(1-\beta)\mathrm{D}_{\mathrm{KL}}(q\|p_a) + \beta \mathrm{D}_{\mathrm{KL}}(q\|p_b) \tag{B.2}$$

with the constraint that $\sum_{\mathbf{x}} q(\mathbf{x}) = 1$. The Lagrangian is given by:

$$\begin{aligned}\mathcal{L}(q) &= \lambda\left(\sum_{\mathbf{x}} q(\mathbf{x}) - 1\right) + (1-\beta)\sum_{\mathbf{x}} q(\mathbf{x})\left(\log q(\mathbf{x}) - \log p_a(\mathbf{x})\right) + \\ &\quad + \beta\sum_{\mathbf{x}} q(\mathbf{x})\left(\log q(\mathbf{x}) - \log p_b(\mathbf{x})\right) \\ &= -\lambda + \sum_{\mathbf{x}} \lambda q(\mathbf{x}) + q(\mathbf{x})\log q(\mathbf{x}) - q(\mathbf{x})\left[(1-\beta)\log p_a(\mathbf{x}) - \beta\log p_b(\mathbf{x})\right]\end{aligned}$$

Differentiating with respect to $q(\mathbf{x})$,

$$\frac{\partial\mathcal{L}(q)}{\partial q(\mathbf{x})} = \lambda + 1 + \log q(\mathbf{x}) - (1-\beta)\log p_a(\mathbf{x}) - \beta\log p_b(\mathbf{x}).$$

Setting this to zero gives:

$$q(\mathbf{x}) \propto p_a(\mathbf{x})^{1-\beta} p_b(\mathbf{x})^{\beta}.$$

This is the optimum over the probability simplex. If p_a and p_b belong to an exponential family $\mathcal{P}$, with natural parameters $\boldsymbol{\eta}_{p_a}$ and $\boldsymbol{\eta}_{p_b}$, the optimum is achieved within $\mathcal{P}$ using $\boldsymbol{\eta}_\beta = (1-\beta)\boldsymbol{\eta}_{p_a} + \beta\boldsymbol{\eta}_{p_b}$.

B.2.2 Moment averages

Suppose we wish to find

$$p_\beta^{(MA)} = \arg\min_q \ (1-\beta)\mathrm{D}_{\mathrm{KL}}(p_a\|q) + \beta\mathrm{D}_{\mathrm{KL}}(p_b\|q). \tag{B.3}$$

We write the cost function in terms of the natural parameters $\boldsymbol{\eta}$:

$$\begin{aligned}J(\boldsymbol{\eta}) &= (1-\beta)\sum_{\mathbf{x}} p_a(\mathbf{x})(\log p_a(\mathbf{x}) - \log q(\mathbf{x})) + \beta\sum_{\mathbf{x}} p_b(\mathbf{x})(\log p_b(\mathbf{x}) - \log q(\mathbf{x})) \\ &= \text{const} - \sum_{\mathbf{x}} \left[(1-\beta)p_a(\mathbf{x}) + \beta p_b(\mathbf{x})\right]\log q(\mathbf{x}) \\ &= \text{const} + \log\mathcal{Z}(\boldsymbol{\eta}) - \sum_{\mathbf{x}} \left[(1-\beta)p_a(\mathbf{x}) + \beta p_b(\mathbf{x})\right]\boldsymbol{\eta}^T\mathbf{g}(\mathbf{x})\end{aligned}$$

The partial derivatives are given by:

$$\begin{aligned}\frac{\partial J}{\partial \eta_i} &= \sum_{\mathbf{x}} q(\mathbf{x}) g_i(\mathbf{x}) - \sum_{\mathbf{x}} \left[(1-\beta)p_a(\mathbf{x}) + \beta p_b(\mathbf{x})\right] g_i(\mathbf{x}) \\ &= \mathbb{E}_q[g_i(\mathbf{x})] - (1-\beta)\mathbb{E}_{p_a}(g_i(\mathbf{x})) - \beta\mathbb{E}_{p_b}(g_i(\mathbf{x}))\end{aligned}$$

Setting this to zero, we see that the optimal solution is given by averaging the moments of p_a and p_b:

$$\mathbb{E}_q[g_i(\mathbf{x})] = (1-\beta)\mathbb{E}_{p_a}(g_i(\mathbf{x})) + \beta\mathbb{E}_{p_b}(g_i(\mathbf{x}))$$

Intuitively, this can be thought of as a maximum likelihood estimate of $\boldsymbol{\eta}$ for a dataset with $(1-\beta)$ fraction of the points drawn from p_a and β fraction drawn from p_b.

B.3 AIS example: univariate Gaussians

Here we evaluate the cost functionals for the Gaussian example of Section 4.2 under γ_{GA} and γ_{MA} using both linear and optimal schedules. Recall that $p_a = \mathcal{N}(\mu_a, \sigma)$ and $p_b = \mathcal{N}(\mu_b, \sigma)$. The natural parameters of the Gaussian are the information form representation, with precision $\lambda = 1/\sigma^2$ and potential $\mathbf{h} = \lambda\mu$. The sufficient statistics are the first and (rescaled) second moments given by $\mathbb{E}[x] = \mu$ and $-\frac{1}{2}\mathbb{E}[x^2] = -\frac{1}{2}s \equiv -\frac{1}{2}(\sigma^2 + \mu^2)$.

Throughout this section, we use the relationship $\mathbf{G}_{\boldsymbol{\eta}}\dot{\boldsymbol{\eta}} = \dot{\mathbf{s}}$ (Amari (2000), sec. 3.3), so that $\mathcal{F}(\gamma)$ can be rewritten as $\frac{1}{2}\int_0^1 \dot{\boldsymbol{\eta}}(\beta)^T \dot{\mathbf{s}}(\beta)\, \mathrm{d}\beta$.

To simplify calculations, let β range from $-1/2$ to $1/2$ (rather than 0 to 1), and assume $\mu_a = -1/2$ and $\mu_b = 1/2$. The general case can be obtained by rescaling μ_a, μ_b, and σ.

B.3.1 Geometric averages

Geometric averages correspond to averaging the natural parameters:

$$\begin{aligned}\lambda(\beta) &= 1/\sigma^2 \\ \mathbf{h}(\beta) &= \beta/\sigma^2\end{aligned}$$

Solving for the moments,

$$\begin{aligned}\mu(\beta) &= \beta \\ s(\beta) &= \sigma^2 + \beta^2.\end{aligned}$$

The derivatives are given by:

$$\begin{aligned}\dot{\lambda}(\beta) &= 0 \\ \dot{h}(\beta) &= 1/\sigma^2 \\ \dot{\mu}(\beta) &= 1 \\ \dot{s}(\beta) &= 2\beta\end{aligned}$$

Ignoring the constant, the cost functional is given by:

$$\begin{aligned}\mathcal{F}(\gamma) &= \frac{1}{2}\int_{-1/2}^{1/2} \dot{h}(\beta)\dot{\mu}(\beta) - \frac{1}{2}\dot{\lambda}(\beta)\dot{s}(\beta)\,\mathrm{d}\beta \\ &= \frac{1}{2}\int_{-1/2}^{1/2} \frac{1}{\sigma^2}\,\mathrm{d}\beta \\ &= \frac{1}{2\sigma^2}.\end{aligned}$$

We can also compute the cost under the optimal schedule by computing the path length (see Section 3):

$$\begin{aligned}\ell(\gamma) &= \int_{-1/2}^{1/2} \sqrt{\dot{h}(\beta)\dot{\mu}(\beta) - \frac{1}{2}\dot{\lambda}(\beta)\dot{s}(\beta)}\,\mathrm{d}\beta \\ &= \int_{-1/2}^{1/2} \sqrt{1/\sigma^2}\,\mathrm{d}\beta \\ &= \frac{1}{\sigma}.\end{aligned}$$

Since the functional under the optimal schedule is given by $\ell^2/2$, these two answers agree with each other, i.e. the linear schedule is optimal.

We assumed for simplicity that $\mu_a = -1/2$ and $\mu_b = 1/2$. In general, we can rescale σ and $\mu_b - \mu_a$ by the same amount without changing the functional. Therefore, $\mathcal{F}(\gamma_{GA})$ is given by:

$$\frac{(\mu_b - \mu_a)^2}{2\sigma^2} \equiv \frac{d^2}{2}.$$

B.3.2 Moment averaging

Now let's look at moment averaging. The parameterizations are given by:

$$\begin{aligned}
\mu(\beta) &= \beta \\
s(\beta) &= \sigma^2 + \frac{1}{4} \\
\lambda(\beta) &= \left(\sigma^2 + \frac{1}{4} - \beta^2\right)^{-1} \\
\mathbf{h}(\beta) &= \left(\sigma^2 + \frac{1}{4} - \beta^2\right)^{-1} \beta
\end{aligned}$$

with derivatives

$$\begin{aligned}
\dot{\mu}(\beta) &= 1 \\
\dot{s}(\beta) &= 0 \\
\dot{\lambda}(\beta) &= 2\left(\sigma^2 + \frac{1}{4} - \beta^2\right)^{-2} \beta \\
\dot{h}(\beta) &= \lambda(\beta)\dot{\mu}(\beta) + \mu(\beta)\dot{\lambda}(\beta) \\
&= \left(\sigma^2 + \frac{1}{4} - \beta^2\right)^{-1} + 2\left(\sigma^2 + \frac{1}{4} - \beta^2\right)^{-2} \beta^2
\end{aligned}$$

The cost functional is given by:

$$\begin{aligned}
\mathcal{F}(\gamma_{MA}) &= \frac{1}{2}\int_{-1/2}^{1/2} \dot{\mu}(\beta)\dot{h}(\beta) - \frac{1}{2}\dot{s}(\beta)\dot{\lambda}(\beta) \\
&= \frac{1}{2}\int_{-1/2}^{1/2} \dot{h}(\beta)\,\mathrm{d}\beta \\
&= \frac{1}{2}[\mathbf{h}(1/2) - \mathbf{h}(-1/2)] \\
&= \frac{1}{2\sigma^2}.
\end{aligned}$$

This agrees exactly with $\mathcal{F}(\gamma_{GA})$, consistent with Theorem 2.

However, we can see by inspection that for small σ, most of the mass of this integral is concentrated near the endpoints, where the variance changes suddenly. This suggests that the optimal schedule would place more intermediate distributions

near the endpoints.

We can bound the cost under the optimal schedule by bounding the path length $\ell(\gamma_{MA})$:

$$\begin{aligned}
\ell(\gamma_{MA}) &= \int_{-1/2}^{1/2} \sqrt{\dot{\mu}(\beta)\dot{h}(\beta) - \frac{1}{2}\dot{s}(\beta)\dot{\lambda}(\beta)}\,\mathrm{d}\beta \\
&= \int_{-1/2}^{1/2} \sqrt{\dot{h}(\beta)}\,\mathrm{d}\beta \\
&= \int_{-1/2}^{1/2} \sqrt{\lambda(\beta)\dot{\mu}(\beta) + \mu(\beta)\dot{\lambda}(\beta)}\,\mathrm{d}\beta \\
&\leq \int_{-1/2}^{1/2} \sqrt{|\lambda(\beta)\dot{\mu}(\beta)|}\,\mathrm{d}\beta + \int_{-1/2}^{1/2} \sqrt{|\mu(\beta)\dot{\lambda}(\beta)|}\,\mathrm{d}\beta \\
&= \int_{-1/2}^{1/2} \frac{1}{\sqrt{\sigma^2 + \frac{1}{4} - \beta^2}}\,\mathrm{d}\beta + \sqrt{2}\int_{-1/2}^{1/2} \frac{|\beta|}{\sigma^2 + \frac{1}{4} - \beta^2}\,\mathrm{d}\beta \\
&= 2\sin^{-1}\left(\frac{1}{\sqrt{4\sigma^2+1}}\right) + \sqrt{2}\log\left(1 + \frac{1}{4\sigma^2}\right) \\
&\leq \pi + \sqrt{2}\log\left(1 + \frac{1}{4\sigma^2}\right)
\end{aligned}$$

The path length has dropped from linear to logarithmic! Since $\mathcal{F}$ grows like ℓ^2, the cost drops from quadratic to log squared.

This shows that even though Theorem 2 guarantees that both γ_{GA} and γ_{MA} have the same cost under a linear schedule, one path may do substantially better than the other if one is allowed to change the schedule.

Bibliography

D. J. Aldous. Exchangeability and related topics. In *Lecture Notes in Mathematics*, volume 1117, pages 1–198. Springer, 1985.

Shun-ichi Amari and Hiroshi Nagaoka. *Methods of Information Geometry*. Oxford University Press, 2000.

S. Arora, R. Ge, Y. Halpern, D. Mimno, A. Moitra, D. Sontag, Y. Wu, and M. Zhu. A practical algorithm for topic modeling with provable guarantees. arXiv:1212.4777, 2012.

S. Arora, A. Bhaskara, R. Ge, and T. Ma. Provable bounds for learning some deep representations. arXiv:1310.6343, 2013.

F. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Advances in Neural Information Processing Systems*, pages 105–112. 2009.

A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *Transactions on Information Theory*, 1998.

A. T. Basilevsky. *Statistical Factor Analysis and Related Methods: Theory and Applications*. Wiley-Interscience, 1994.

Gundula Behrens, Nial Friel, and Merrilee Hurn. Tuning tempered transitions. *Statistics and Computing*, 22:65–78, 2012.

R. M. Bell and Y. Koren. Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

J. O. Berger and L. R. Pericchi. The intrinsic bayes factor for model selection and prediction. *Journal of the American Statistical Association*, 91(433):109–122, 1996.

P. Berkes, R. Turner, and M. Sahani. On sparsity and overcompleteness in image models. In *Advances in Neural Information Processing Systems*, 2008.

W. Bing, Z. Wen-qiong, C. Ling, and L. Jia-hong. A GP-based kernel construction and optimization method for RVM. In *International Conference on Computer and Automation Engineering (ICCAE)*, volume 4, pages 419–423, 2010.

C.M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

D. M. Blei, M. I. Jordan, T. L. Griffiths, and J. B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *Neural Inf. Proc. Systems*, 2003.

J. Borgstrom, A. D. Gordon, M. Greenberg, J. Margetson, and J. van Gael. Measure transformer semantics for Bayesian machine learning. Technical Report MSR-TR-2011-18, Microsoft Research, 2011.

T. Bossomaier and A. W. Snyder. Why spatial frequency processing in the visual cortex? *Vision Research*, 26(8):1307–1309, 1986.

G.E.P. Box, G.M. Jenkins, and G.C. Reinsel. *Time series analysis: forecasting and control*. 1976.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Univeristy Press, 2004.

Ben Calderhead and Mark Girolami. Estimating Bayes factors via thermodynamic integration and population MCMC. *Computational Statistics and Data Analysis*, 53(12): 4028–4045, 2009.

C. M. Carvalho, M. S. Johannes, H. F. Lopes, and N. G. Polson. Particle learning and smoothing. *Statistical Science*, 25(1):88–106, 2010.

S. Chib. Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, 90(432):1313–1321, 1995.

S. Chib and I. Jeliazkov. Marginal likelihood from the Metropolis-Hastings output. *Journal of the American Statistical Association*, 96(453):270–281, 2001.

N. Chomsky. On cognitive structures and their development: a reply to Piaget. In M. Piattelli-Palmarini, editor, *Language and learning: the debate between Jean Piaget and Noam Chomsky*. Harvard University Press, 1980.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

M. Christoudias, R. Urtasun, and T. Darrell. Bayesian localized multiple kernel learning. *Technical report, EECS Department, University of California, Berkeley*, 2009.

P. del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–38, 1977.

Guillaume Desjardins, Aaron Courville, and Yoshua Bengio. On tracking the partition function. In *NIPS 24*. MIT Press, 2011.

L. Diosan, A. Rogozan, and J.P. Pecuchet. Evolving kernel functions for SVMs by genetic programming. In *Machine Learning and Applications, 2007*, pages 19–24. IEEE, 2007.

Finale Doshi-Velez and Zoubin Ghahramani. Accelerated sampling for the Indian buffet process. In *Int'l. Conf. on Machine Learning*, 2009.

D. Duvenaud, H. Nickisch, and C.E. Rasmussen. Additive Gaussian processes. In *Advances in Neural Information Processing Systems*, 2011.

D. Duvenaud, J. R. Lloyd, R. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, 2013.

G. Elidan, N. Lotner, N. Friedman, and D. Koller. Discovering hidden variables: a structure-based approach. In *Neural Inf. Proc. Systems*, 2000.

D. Erhan, P. Manzagol, Y. Bengio, S. Bengio, and P. Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *Artificial Intelligence and Statistics*, 2009.

Daan Frenkel and Berend Smit. *Understanding Molecular Simulation: From Algorithms to Applications*. Academic Press, 2 edition, 2002.

N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Intl. Conf. on Machine Learning*, 1997.

A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis*. CRC Press, 3 edition, 2014.

Andrew Gelman and Xiao-Li Meng. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical Science*, 13(2):163–186, 1998.

S. Geman. A limit theorem for the norm of random matrices. *Annals of Probability*, 8(2): 252–261, 1980.

J. Geweke. Getting it right: joint distribution tests of posterior simulators. *Journal of the American Statistical Association*, 99(467):799–804, 2004.

Z. Ghahramani. Bayesian nonparametrics and the probabilistic approach to modeling. *Philosophical Transactions of the Royal Society A*, 371(1984):1–27, 2012.

N. D. Goodman, V. K. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum. Church: a language for generative models. In *Neural Information Processing Systems*, 2008.

N. D. Goodman, T. Ullman, and J. B. Tenenbaum. Learning a theory of causality. In *Proceedings of the Conference of the Cognitive Science Society*, 2009.

P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 1995.

T. Griffiths and Z. Ghahramani. Infinite latent feature models and the indian buffet process. Technical report, Gatsby Computational Neuroscience Unit, 2005.

R. B. Grosse. Shift-invariant sparse coding for audio classification. In *Uncertainty in AI*, 2007.

Roger B. Grosse, Ruslan Salakhutdinov, William T. Freeman, and Joshua B. Tenenbaum. Exploiting compositionality to explore a large space of model structures. In *Uncertainty in AI*, 2012.

C. Gu. *Smoothing spline ANOVA models*. Springer Verlag, 2002. ISBN 0387953531.

S. Harmeling and C. K. I. Williams. Greedy learning of binary latent trees. *Pattern Analysis and Machine Intelligence*, 33(6):1087–1097, 2011.

T.J. Hastie and R.J. Tibshirani. *Generalized additive models*. Chapman & Hall/CRC, 1990.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 2006.

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

E. Hsu, K. Pulli, and J. Popovic. Style translations for human motion. In *ACM Transactions on Graphics*, 2005.

Y. Iba. Extended ensemble Monte Carlo. *International Journal of Modern Physics C*, 12 (5):623–656, 2001.

Christopher Jarzynski. Equilibrium free-energy differences from nonequilibrium measurements: A master-equation approach. *Physical Review E*, 56:5018–5035, 1997.

E. T. Jaynes. Highly informative priors. In *Proceedings of the Second International Meeting on Bayesian Statistics*, 1985.

Y. Karklin and M. S. Lewicki. A hierarchical Bayesian model for learning nonlinear statistical regularities in nonstationary natural signals. *Neural Computation*, 17(2):397–423, 2005.

Y. Karklin and M. S. Lewicki. Emergence of complex cell properties by learning to generalize in natural scenes. *Nature*, 457:83–86, January 2008.

R. E. Kass. Bayes factors in practice. *The Statistician*, 42(5):551–560, 1993.

R. E. Kass and A E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795, 1995.

Charles Kemp and Joshua B. Tenenbaum. The discovery of structural form. *PNAS*, 2008.

Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, pages 381–388, 2006.

B. W. Koehn and J. M. Zytkow. Experimenting and theorizing in theory formation. In *International Workshop on Machine Learning*, 1987.

T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 2007.

P. Langley and J. M. Zytkow. Data-driven approaches to empirical discovery. *Artificial Intelligence*, 40:283–312, 1989.

Pat Langley, Herbert A. Simon, and Gary L. Bradshaw. Heuristics for empirical discovery. In *Knowledge Based Learning Systems*. Springer-Verlag, London, UK, 1984.

N. Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816, 2005.

J. Lean, J. Beer, and R. Bradley. Reconstruction of solar irradiance since 1610: Implications for climate change. *Geophysical Research Letters*, 22(23):3195–3198, 1995.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

H. Lee, C. Ekanadham, and A. Y. Ng. Sparse deep belief net model for visual area V2. In *Neural Information Processing Systems*, 2008.

S. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using L1-regularization. In *NIPS*, 2006.

M. Li and P. Vitanyi. *An introduction to Kolmogorov complexity and its applications*. Springer, 1997.

S. Lyu and E. P. Simoncelli. Nonlinear extraction of independent components of natural images using radial Gaussianization. *Neural Computation*, 21(6):1485–1519, 2009.

D. J. C. MacKay. Comparison of approximate methods for handling hyperparameters. *Neural Computation*, 11(5):1035–1068, 1999.

E. Meeds, Z. Ghahramani, R. Neal, and S. T. Roweis. Modelling dyadic data with binary latent factors. In *NIPS*, volume 20, pages 1002–1009, 2006.

J. W. Miller and M. T. Harrison. A simple example of Dirichlet process mixture inconsistency for the number of components. In *Neural Information Processing Systems*, 2013.

K. T. Miller, T. L. Griffiths, and M. I. Jordan. Nonparametric latent feature models for link prediction. In *Advances in Neural Information Processing Systems*, 2009.

I. Murray and R. Salakhutdinov. Evaluating probabilities under high-dimensional latent variable models. In *Neural Information Processing Systems*, 2009.

R. M. Neal. Erroneous results in "Marginal likelihood from the Gibbs output". Available at `http://www.cs.toronto.edu/~radford/chib-lettter.html`, 1999.

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

R. M. Neal. Transferring prior information between models using imaginary data. Technical report, University of Toronto, 2001b.

R. M. Neal. Slice sampling. *Annals of Statistics*, 2003.

R. M. Neal. The harmonic mean of the likelihood: worst Monte Carlo method ever. Blog post, 2008. Available at http://radfordneal.wordpress.com/2008/08/17/the-harmonic-mean-of-the-likelihood-worst-monte-carlo-method-ever/.

Radford Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and Computing*, 6:353–366, 1996.

Radford M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 1992.

M. A. Newton and A. E. Raftery. Approximate Bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society: Series B (Methodology)*, 56(1):3–48, 1994.

A. O'Hagan. Fractional Bayes factors for model comparison. *Journal of the Royal Statistical Society: Series B (Methodology)*, 57(1):99–138, 1995.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–9, June 1996.

D. N. Osherson, J. Stern, O. Wilkie, M. Stob, and E. E. Smith. Default probability. *Cognitive Science*, 15:251–269, 1991.

A. Perfors, J. B. Tenenbaum, and T. Reiger. The learnability of abstract syntactic principles. *Cognition*, 118(3):306–338, 2011.

S. T. Piantadosi, J. B. Tenenbaum, and N. D. Goodman. Bootstrapping in a language of thought: a formal model of numerical concept learning. *Cognition*, 123:199–217, 2012.

S. Pinker. *The Blank Slate*. Penguin Books, 2003.

T.A. Plate. Accuracy versus interpretability in flexible modeling: Implementing a tradeoff using Gaussian process models. *Behaviormetrika*, 26:29–50, 1999. ISSN 0385-7417.

D. Pomerleau, G. E. Hinton, M. Palatucci, and T. M. Mitchell. Zero-shot learning with semantic output codes. In *NIPS*, 2009.

H. Poon and P. Domingos. Sum-product networks: a new deep architecture. In *Conference on Uncertainty in AI*, 2011.

J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–71, 2000.

J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions on Signal Processing*, 12(11):1338–1351, 2003.

C. E. Rasmussen and Z. Ghahramani. Occam's razor. In *Neural Information Processing Systems*, 2001.

C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

S. Roweis. EM algorithms for PCA and SPCA. In *Neural Information Processing Systems*, 1998.

D. Ruppert, M.P. Wand, and R.J. Carroll. *Semiparametric regression*, volume 12. Cambridge University Press, 2003.

R. Salakhutdinov and G. Hinton. Using deep belief nets to learn covariance kernels for Gaussian processes. *Advances in Neural information processing systems*, 20:1249–1256, 2008.

R. Salakhutdinov and G. E. Hinton. Deep Boltzmann machines. In *Neural Inf. Proc. Systems*, 2009.

R. Salakhutdinov, J. B. Tenenbaum, and A. Torralba. Learning with hierarchical-deep models. *Pattern Analysis and Machine Intelligence*, 35(8):1958–1971, 2013.

Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, 2008.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Int'l. Conf. on Machine Learning*, 2008.

Ruslan Salakhutdinov, Antonio Torralba, and Joshua B. Tenenbaum. Learning to share visual appearance for multiclass object detection. In *Comp. Vision and Pattern Recog.*, 2011.

M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

R. N. Shepard. Multidimensional scaling, tree-fitting, and clustering. *Science*, 210(4468): 390–398, 1980.

E. P. Simoncelli. Statistical models for images: compression, restoration, and synthesis. In *Asilomar Conference on Signals and Systems*, 1997.

John Skilling. Nested sampling for general Bayesian computation. *Bayesian Analysis*, 1 (4):833–859, 2006.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. arXiv:1206.2944, 2012.

Jascha Sohl-Dickstein and Benjamin J. Culpepper. Hamiltonian annealed importance sampling for partition function estimation. Technical report, Redwood Center, UC Berkeley, 2012.

R. J. Solomonoff. A formal theory of inductive inference, part I. *Information and Control*, 7(1):1–22, 1964.

M. Spain and P. Perona. Measuring and predicting importance of objects in our visual world. Technical report, California Institute of Technology, 2008.

I. Sutskever, R. Salakhutdinov, and J. B. Tenenbaum. Modelling relational data using Bayesian clustered tensor factorization. In *NIPS*, pages 1821–1828. 2009.

Graham Taylor and Geoffrey Hinton. Products of hidden Markov models: It takes $N > 1$ to tango. In *Uncertainty in Artificial Intelligence*, 2009.

Graham W. Taylor, Geoffrey E. Hinton, and Sam Roweis. Modeling human motion using binary latent variables. In *NIPS*, 2007.

J. B. Tenenbaum. *A Bayesian framework for concept learning*. PhD thesis, MIT, February 1999.

M. Teyssier and D. Koller. Ordering-based search: a simple and effective algorithm for learning Bayesian networks. In *UAI*, 2005.

Lucas Theis, Sebastian Gerwinn, Fabian Sinz, and Matthias Bethge. In all likelihood, deep belief is not enough. *Journal of Machine Learning Research*, 12:3071–3096, 2011.

Tijmen Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Intl. Conf. on Machine Learning*, 2008.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

Ljupco Todorovski and Saso Dzeroski. Declarative bias in equation discovery. In *Int'l. Conf. on Machine Learning*, 1997.

G. Wahba. *Spline models for observational data*. Society for Industrial Mathematics, 1990. ISBN 0898712440.

Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

Martin J. Wainwright, Eero P. Simoncelli, and Alan S. Willsky. Random cascades on wavelet trees and their use in analyzing and modeling natural images. In *Applied and Computational Harmonic Analysis*, 2001.

H. M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno. Evaluation methods for topic models. In *Int'l. Conf. on Machine Learning*, 2009.

T. Washio, H. Motoda, Y. Niwa, et al. Discovering admissible model equations from observed data based on scale-types and identity constraints. In *International Joint Conference On Artifical Intelligence*, volume 16, pages 772–779, 1999.

Andrew Gordon Wilson and Ryan Prescott Adams. Gaussian process covariance kernels for pattern discovery and extrapolation. Technical Report arXiv:1302.4245 [stat.ML], February 2013.

R. S. Zemel and G. E. Hinton. Developing population codes by minimizing description length. In *NIPS*, 1994.