

# Sandwiching the marginal likelihood using bidirectional Monte Carlo

Roger B. Grosse  
University of Toronto  
rgrosse@cs.toronto.edu

Zoubin Ghahramani  
University of Cambridge  
zoubin@eng.cam.ac.uk

Ryan P. Adams  
Twitter and Harvard University  
rpa@seas.harvard.edu

## Abstract

Computing the marginal likelihood (ML) of a model requires marginalizing out all of the parameters and latent variables, a difficult high-dimensional summation or integration problem. To make matters worse, it is often hard to measure the accuracy of one’s ML estimates. We present bidirectional Monte Carlo, a technique for obtaining accurate log-ML estimates on data simulated from a model. This method obtains stochastic lower bounds on the log-ML using annealed importance sampling or sequential Monte Carlo, and obtains stochastic upper bounds by running these same algorithms in reverse starting from an exact posterior sample. The true value can be sandwiched between these two stochastic bounds with high probability. Using the ground truth log-ML estimates obtained from our method, we quantitatively evaluate a wide variety of existing ML estimators on several latent variable models: clustering, a low rank approximation, and a binary attributes model. These experiments yield insights into how to accurately estimate marginal likelihoods.

## 1 Introduction

One commonly used model selection criterion is the marginal likelihood (ML) of the model, or  $p(\mathcal{D}|\mathcal{M}_i)$ , where  $\mathcal{D}$  denotes the observed data and  $\mathcal{M}_i$  denotes the model class (Kass and Raftery, 1995). Marginal likelihood is an appealing criterion for several reasons. First, it can be plugged into Bayes’ Rule to compute a posterior distribution over models, a practice known as Bayesian model comparison:

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

Second, the ML criterion 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 (MacKay, 1992; Rasmussen and Ghahramani, 2001). Third, it is closely related to description length (Barron et al., 1998), a compression-based criterion for model selection. Finally, since the ML 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 one is able to compute it efficiently. 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.

The main difficulty in applying marginal likelihood is that it is intractable to compute for most models of interest. Computing  $p(\mathcal{D}|\mathcal{M}_i)$  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 effort has been spent finding ways to compute or approximate partition functions for purposes of model selection (Kass and Raftery, 1995; Meng and Wong, 1996; Gelman and Meng, 1998; Neal, 2001a; Murray and Salakhutdinov, 2008; Attias, 2000; Chib, 1995; Murray and Salakhutdinov, 2009; Skilling, 2006; Wallach et al., 2009). Different partition function estimation algorithms vary greatly in their accuracy, computational efficiency, and ease of implementation. Unfortunately, it is often difficult to determine which method to use in a given situation.

ML estimators can give inaccurate results in several ways. Subtle implementation bugs 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).

A major obstacle to developing effective ML estimators, and partition function estimators more generally, 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 does not have independent access to that value (otherwise one would not need to run the estimator). In a handful of cases, such as Ising models (Jerrum and Sinclair, 1992), one can tractably approximate the partition function to arbitrary accuracy using specialized algorithms. These models can be used to benchmark partition function estimators. However, such polynomial-time approximation schemes may not be known for any models sufficiently similar to the one whose ML needs to be measured. Alternatively, one can test the estimator on small problem instances for which the partition function can be easily computed (e.g. Murray and Salakhutdinov, 2008), but this might not accurately reflect how it will perform on hard instances. It is also common to run multiple estimators and evaluate them based on their consistency with each other, under the implicit assumption that multiple algorithms are unlikely to fail in the same way. However, it has been observed that seemingly very different algorithms can report nearly the same value with high confidence, yet be very far from the true value (Iain Murray, personal communication).

In this paper, we present bidirectional Monte Carlo, a method for accurately estimating the ML for data *simulated* from a model by sandwiching the true value between stochastic upper and lower bounds. In the limit of infinite computation, the two stochastic bounds are guaranteed to converge, so one need only run the algorithms for enough steps to achieve the desired accuracy. The technique is applicable both in the setting of ML estimation (where one integrates out both the parameters and the latent variables for an entire dataset) and in the setting of held-out likelihood estimation (where the parameters are fixed and one wants to integrate out the latent variables for a single data case).

As of yet, we do not know of a way to obtain accurate ML upper bounds on real-world data. However, the ability to accurately estimate ML on simulated data has several implications. First, one can measure the accuracy of an ML estimator on simulated data with similar characteristics to the real-world data. This can give a rough indication of how accurate the results would be in practice, even though there is no rigorous guarantee. Second, one can use this technique to

construct a rigorous testbed for quantitatively evaluating ML estimators. This can be used to guide development of sampling algorithms, and perhaps even to optimize algorithm hyperparameters using Bayesian optimization (Snoek et al., 2012). Finally, this method also provides a rigorous way to quantitatively evaluate MCMC transition operators. In particular, the gap between the stochastic upper and lower bounds is itself a stochastic upper bound on the KL divergence of the distribution of approximate samples from the true posterior. By testing one’s approximate posterior sampler on simulated data, one can get a rough idea of how it is likely to perform in practice, at least if one trusts that the simulated data are sufficiently realistic.

The organization of this paper is as follows. Section 3 provides background on existing ML estimators which this work builds on. In Section 4, we describe bidirectional Monte Carlo, our method for sandwiching the marginal likelihood for simulated data. Section 5 describes some additional ML estimators which we compare in our experiments. Finally, in Section 6, we present experiments where we used our method to compute ML values on simulated data for three models: a clustering model, a low rank factorization, and a binary attribute model. We compare a wide variety of existing ML estimators on these models and conclude with recommendations for how the ML should be estimated in practice.

## 2 Preliminaries

Throughout this paper, in some cases it will be convenient to discuss the general partition function estimation problem, and in other cases it will be more convenient to discuss marginal likelihood estimation in particular. In the general partition function estimation setting, we have an unnormalized probability distribution  $f$  over states  $\mathbf{x} \in \mathcal{X}$ , and we wish to evaluate the partition function  $\mathcal{Z} = \sum_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$ .

When we discuss marginal likelihood in particular, we let  $\mathbf{y}$  denote the observations,  $\boldsymbol{\theta}$  denote the parameters, and  $\mathbf{z}$  denote the latent variables. (For instance, in a clustering model,  $\boldsymbol{\theta}$  might correspond to the locations of the cluster centers and  $\mathbf{z}$  might correspond to the assignments of data points to clusters.) For notational convenience, we will assume continuous parameters and discrete latent variables, but this is not required by any of the algorithms we discuss. One defines the model by way of a joint distribution which we assume obeys the factorization

$$p(\boldsymbol{\theta}, \mathbf{y}, \mathbf{z}) = p(\boldsymbol{\theta})p(\mathbf{z}|\boldsymbol{\theta})p(\mathbf{y}|\mathbf{z}, \boldsymbol{\theta}) \tag{1}$$

$$= p(\boldsymbol{\theta}) \prod_{i=1}^N p(\mathbf{z}_i|\boldsymbol{\theta}) p(\mathbf{y}_i|\mathbf{z}_i, \boldsymbol{\theta}) \tag{2}$$

We are interested in computing the marginal likelihood

$$p(\mathbf{y}) = \int p(\boldsymbol{\theta}) \prod_{i=1}^N \sum_{\mathbf{z}_i} p(\mathbf{z}_i|\boldsymbol{\theta}) p(\mathbf{y}_i|\mathbf{z}_i, \boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{3}$$

Marginal likelihood estimation can be seen as a special case of partition function estimation, where  $\mathbf{x} = (\boldsymbol{\theta}, \mathbf{z})$ , and  $f$  is the joint distribution  $p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{y})$  viewed as a function of  $\boldsymbol{\theta}$  and  $\mathbf{z}$ .

### 3 Background

Our method for sandwiching the marginal likelihood builds closely upon prior work on marginal likelihood estimation, in particular annealed importance sampling (AIS) and sequential Monte Carlo (SMC). This section reviews the techniques which we use in our procedure. Discussion of alternative ML estimators is deferred to Section 5.

#### 3.1 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  $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(K)}$  from a proposal distribution  $q$  whose support contains the support of  $p$ , and compute the estimate

$$\hat{\mathcal{Z}} = \frac{1}{K} \sum_{k=1}^K w^{(k)} \triangleq \frac{1}{K} \sum_{k=1}^K \frac{f(\mathbf{x}^{(k)})}{q(\mathbf{x}^{(k)})}. \quad (4)$$

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

$$\mathbb{E}_{\mathbf{x} \sim q} \left[ \frac{f(\mathbf{x})}{q(\mathbf{x})} \right] = \mathcal{Z}. \quad (5)$$

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  $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(K)}$  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.

#### 3.2 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 Eqn. 5, we obtain:

$$\mathbb{E}_{\boldsymbol{\theta}, \mathbf{z} \sim p(\boldsymbol{\theta}, \mathbf{z} | \mathbf{y})} \left[ \frac{p(\boldsymbol{\theta}, \mathbf{z})}{p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{y})} \right] = \frac{1}{p(\mathbf{y})}. \quad (6)$$

This suggests the following estimator: draw samples  $\{\boldsymbol{\theta}^{(k)}, \mathbf{z}^{(k)}\}_{k=1}^K$  from the posterior  $p(\boldsymbol{\theta}, \mathbf{z} | \mathbf{y})$ , and compute weights  $w^{(k)} = p(\boldsymbol{\theta}^{(k)}, \mathbf{z}^{(k)}) / p(\boldsymbol{\theta}^{(k)}, \mathbf{z}^{(k)}, \mathbf{y}) = 1 / p(\mathbf{y} | \boldsymbol{\theta}^{(k)}, \mathbf{z}^{(k)})$ . The weights  $w^{(k)}$  are unbiased estimators of the *reciprocal* of the marginal likelihood. The ML estimate, then, is computed from the harmonic mean of the likelihood values:

$$\hat{p}(\mathbf{y}) = \frac{K}{\sum_{k=1}^K w^{(k)}} = \frac{K}{\sum_{k=1}^K 1 / p(\mathbf{y} | \boldsymbol{\theta}^{(k)}, \mathbf{z}^{(k)})}. \quad (7)$$

While simple to implement, this estimator is unlikely to perform well in practice (Newton and Raftery, 1994; Neal, 2008).

---

**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  $\hat{\mathcal{Z}} = \sum_{k=1}^K w^{(k)} / K$ 
```

---

### 3.3 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 dissimilar 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 estimating partition functions. Mathematically, the algorithm takes as input a sequence of  $T$  distributions  $p_1, \dots, p_T$ , 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  $\beta_t$  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  $\mathcal{T}_1, \dots, \mathcal{T}_T$ , 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 an unnormalized distribution over  $\boldsymbol{\theta}$  and  $\mathbf{z}$ ). Because  $\mathbf{y}$  is fixed, the latter 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 to raising the likelihood term to a power less than 1:

$$f_t(\boldsymbol{\theta}, \mathbf{z}) = p(\boldsymbol{\theta}, \mathbf{z}) p(\mathbf{y} | \boldsymbol{\theta}, \mathbf{z})^{\beta_t}. \quad (8)$$

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 (Neal, 2001a). In particular, the full set of states  $\mathbf{x}_1, \dots, \mathbf{x}_{T-1}$  visited by the algorithm has a joint distribution (which we call the *forward distribution*) represented by:

$$q_{for}(\mathbf{x}_1, \dots, \mathbf{x}_{T-1}) = p_1(\mathbf{x}_1) \mathcal{T}_2(\mathbf{x}_2 | \mathbf{x}_1) \cdots \mathcal{T}_{T-1}(\mathbf{x}_{T-1} | \mathbf{x}_{T-2}). \quad (9)$$

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:

$$q_{back}(\mathbf{x}_1, \dots, \mathbf{x}_{T-1}) = p_T(\mathbf{x}_{T-1}) \mathcal{T}_{T-1}(\mathbf{x}_{T-2} | \mathbf{x}_{T-1}) \cdots \mathcal{T}_2(\mathbf{x}_1 | \mathbf{x}_2). \quad (10)$$

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

$$\frac{q_{back}(\mathbf{x}_1, \dots, \mathbf{x}_T)}{q_{for}(\mathbf{x}_1, \dots, \mathbf{x}_T)} = \frac{p_T(\mathbf{x}_{T-1})}{p_1(\mathbf{x}_1)} \frac{\mathcal{T}_2(\mathbf{x}_1 | \mathbf{x}_2)}{\mathcal{T}_2(\mathbf{x}_2 | \mathbf{x}_1)} \cdots \frac{\mathcal{T}_{T-1}(\mathbf{x}_{T-2} | \mathbf{x}_{T-1})}{\mathcal{T}_{T-1}(\mathbf{x}_{T-1} | \mathbf{x}_{T-2})} \quad (11)$$

$$= \frac{p_T(\mathbf{x}_{T-1})}{p_1(\mathbf{x}_1)} \frac{f_2(\mathbf{x}_1)}{f_2(\mathbf{x}_2)} \cdots \frac{f_{T-1}(\mathbf{x}_{T-2})}{f_{T-1}(\mathbf{x}_{T-1})} \quad (12)$$

$$= \frac{\mathcal{Z}_1}{\mathcal{Z}_T} w, \quad (13)$$

where  $w$  is the weight computed in Algorithm 1 and (12) 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  $p_T$ :

$$\mathbb{E}_{q_{for}}[wh(\mathbf{x}_{T-1})] = \frac{\mathcal{Z}_T}{\mathcal{Z}_1} \mathbb{E}_{p_T}[h(\mathbf{x})] \quad (14)$$

for any statistic  $h$ . (The partition function estimator corresponds to the special case where  $h(\mathbf{x}) = 1$ .) Ordinarily, one uses the normalized importance weights when estimating expectations.

### 3.4 Sequential Monte Carlo

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

$$p(\mathbf{y}_{1:N}) = p(\mathbf{y}_1) p(\mathbf{y}_2 | \mathbf{y}_1) \cdots p(\mathbf{y}_N | \mathbf{y}_{1:N-1}). \quad (15)$$

(In this section, we use Matlab-style slicing notation.) 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 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. (Some examples 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.*

$$q(\mathbf{z}_i | \mathbf{y}_i, \boldsymbol{\theta}^{(k)}) = p(\mathbf{z}_i | \boldsymbol{\theta}^{(k)}). \quad (16)$$

The particles are then weighted according to the observation likelihood:

$$w^{(k)} \leftarrow w^{(k)} p(\mathbf{y}_i | \mathbf{z}_i^{(k)}, \boldsymbol{\theta}^{(k)}). \quad (17)$$

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

$$q(\mathbf{z}_i | \mathbf{y}_i, \boldsymbol{\theta}^{(k)}) = p(\mathbf{z}_i | \mathbf{y}_i, \boldsymbol{\theta}^{(k)}). \quad (18)$$

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

$$w^{(k)} \leftarrow w^{(k)} p(\mathbf{y}_i | \boldsymbol{\theta}^{(k)}) \quad (19)$$

$$= w^{(k)} \sum_{\mathbf{z}_i} p(\mathbf{z}_i | \boldsymbol{\theta}^{(k)}) p(\mathbf{y}_i | \mathbf{z}_i, \boldsymbol{\theta}^{(k)}). \quad (20)$$

Posterior particle filtering can result in considerably lower variance of the weights compared to standard particle filtering, and therefore better marginal likelihood estimates. (We note that the posterior particle filter can only be applied to those models for which posterior inference of latent variables is tractable.)

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:

$$\mathbf{z}_i^{(k)} \leftarrow \text{sample from } q(\mathbf{z}_i^{(k)} | \mathbf{y}_{1:i}, \mathbf{z}_{1:i-1}^{(k)}) \quad (21)$$

$$w^{(k)} \leftarrow w^{(k)} \frac{p(\mathbf{z}_i^{(k)} | \mathbf{z}_{1:i-1}^{(k)}) p(\mathbf{y}_i | \mathbf{z}_{1:i}^{(k)}, \mathbf{y}_{1:i-1})}{q(\mathbf{z}_i^{(k)} | \mathbf{y}_{1:i}, \mathbf{z}_{1:i-1}^{(k)})} \quad (22)$$

### 3.4.1 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

---

**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  $\hat{\mathcal{Z}} = \frac{1}{K} \sum_{k=1}^K w^{(k)}$ 
```

---

representation one data point at a time, we 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 3.3.

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

$$f_t(\boldsymbol{\theta}, \mathbf{z}) = p(\boldsymbol{\theta}) \prod_{i=1}^N p(\mathbf{z}_i) \prod_{i=1}^t p(\mathbf{y}_i | \boldsymbol{\theta}, \mathbf{z}_i). \quad (23)$$

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

$$\frac{f_t(\boldsymbol{\theta}, \mathbf{z})}{f_{t-1}(\boldsymbol{\theta}, \mathbf{z})} = p(\mathbf{y}_t | \boldsymbol{\theta}, \mathbf{z}_t). \quad (24)$$

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  $p(\mathbf{z}_{t+1:N} | \boldsymbol{\theta})$ .

## 4 Sandwiching the marginal likelihood

In this section, we first define our notion of stochastic upper and lower bounds. We then describe our techniques for obtaining accurate stochastic upper and lower bounds on the log marginal likelihood for simulated data; in combination, these bounds give precise estimates of the true value. Stochastic lower bounds can be computed using various existing methods, as described in Section 4.1. Our



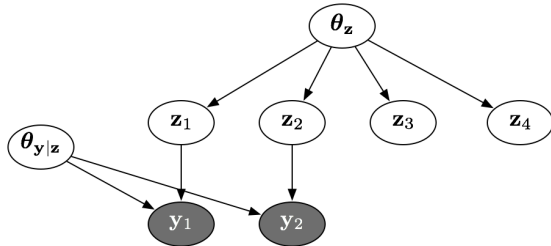


Figure 1: The intermediate distribution  $f_2(\mathbf{y}, \mathbf{z}) = p(\boldsymbol{\theta}) p(\mathbf{z}|\mathbf{y}) p(\mathbf{y}_{1:2}|\boldsymbol{\theta}, \mathbf{z})$ .

main technical contribution is a set of techniques for obtaining accurate stochastic upper bounds. These techniques require an exact sample from the posterior distribution, which one can obtain for simulated data. As described in Sections 4.2.1 and 4.2.2, the key idea is to run AIS or SMC in reverse, starting from the exact posterior sample. Since our final log-ML estimates are obtained by running AIS or SMC both forwards and backwards, we call our approach bidirectional Monte Carlo (BDMC).<sup>1</sup>

#### 4.1 Obtaining stochastic lower bounds

One can obtain stochastic lower bounds on the log-ML using a variety of existing algorithms, as we now outline. In this section, it is more convenient to discuss the general setting of partition function estimation.

Many partition function estimators, such as SIS (Section 3.1) and AIS (Section 3.3), are unbiased, *i.e.*  $\mathbb{E}[\hat{\mathcal{Z}}] = \mathcal{Z}$ .<sup>2</sup> Since  $\mathcal{Z}$  can vary over many orders of magnitude, it is often more meaningful to talk about estimating  $\log \mathcal{Z}$ , rather than  $\mathcal{Z}$ . Unfortunately, unbiased estimators of  $\mathcal{Z}$  may correspond to biased estimators of  $\log \mathcal{Z}$ . In particular, they are stochastic lower bounds, in two senses. 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

$$\Pr(\log \hat{\mathcal{Z}} > \log \mathcal{Z} + b) < e^{-b}. \quad (25)$$

In other words, the estimator is exceedingly unlikely to overestimate  $\log \mathcal{Z}$  by more than a few nats. One can improve this tail bound by combining multiple independent samples (Gogate et al., 2007), but in the context of log-ML estimation, an error of a few nats is insignificant.

The other sense in which  $\log \hat{\mathcal{Z}}$  is a stochastic lower bound on  $\log \mathcal{Z}$  follows from Jensen’s

<sup>1</sup>While we limit our discussion to exact samples obtained from simulated data, other techniques have also been proposed for obtaining exact samples. For instance, this can be done for Ising models using coupling from the past (Propp and Wilson, 1996). BDMC could be used in conjunction with such techniques, without the requirement of simulated data.

<sup>2</sup>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 3.1), 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.

inequality:

$$\mathbb{E}[\log \hat{\mathcal{Z}}] \leq \log \mathbb{E}[\hat{\mathcal{Z}}] = \log \mathcal{Z}. \quad (26)$$

In general, we will use the term *stochastic lower bound* to refer to an estimator which satisfies Eqns. 25 and 26.

Of course, it is not enough to have a stochastic lower bound; we would also like the estimates to be close to the true value. Fortunately, AIS and SMC are both consistent, in that they converge to the correct value in the limit of infinite computation. (For AIS, this means adding more intermediate distributions (Neal, 2001a); for SMC, it means adding more particles (del Moral et al., 2006).) We note that it is also possible to (deterministically) lower bound the log-ML using variational Bayes (discussed in more detail in Section 5.1). However, variational Bayes does not enjoy the same consistency guarantees as AIS and SMC.

## 4.2 Obtaining stochastic upper bounds

Heuristically speaking, we would expect good upper bounds to be harder to obtain than good lower bounds. For a lower bound on the log-ML, it suffices to exhibit regions of high posterior mass. For an upper bound, one would have to demonstrate the absence of any additional probability mass. Indeed, while variational upper bounds have been proposed (Wainwright et al., 2002), we aren't aware of any practical stochastic upper bounds which achieve comparable accuracy to AIS. But suppose we are given a hint in the form of an exact posterior sample. Here, we propose methods which make use of an exact posterior sample to give accurate stochastic upper bounds on the log-ML.

As discussed in Section 3.2, the harmonic mean estimator (HME) is derived from an unbiased estimate of the *reciprocal* of the ML. The arguments of Section 4.1 show that such unbiased estimates of the reciprocal correspond to stochastic upper bounds on the log-ML. Unfortunately, there are two problems with simply using the HME: first, if approximate posterior samples are used, the estimator is not a stochastic upper bound, and in fact can underestimate the log-ML if the sampler failed to find an important mode. Second, as pointed out by Neal (2008) and further confirmed in our experiments, even when exact samples are used, the bound can be extremely poor.

For simulated data, it is possible to work around both of these issues. For the issue of finding exact posterior 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  $\mathbf{y}$ : On one hand, we can simulate from the model by first sampling  $(\boldsymbol{\theta}, \mathbf{z})$  from  $p(\boldsymbol{\theta}, \mathbf{z})$ , and then sampling  $\mathbf{y}$  from  $p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{z})$ . Alternatively, we can first sample  $\mathbf{y}$  from  $p(\mathbf{y})$ , and then sample  $(\boldsymbol{\theta}, \mathbf{z})$  from the posterior  $p(\boldsymbol{\theta}, \mathbf{z}|\mathbf{y})$ . 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  $p(\boldsymbol{\theta}, \mathbf{z}|\mathbf{y})$ . (The Geweke test (Geweke, 2004) is based on the same identity.) In other words, for a simulated dataset, we have available a single exact sample from the posterior, namely the parameters and latent variables used to generate the data.<sup>3</sup>

The other problem with the HME is that the bound can be extremely poor even when computed with exact samples. This happens because the estimator is based on simple importance sampling from the posterior to the prior — two very dissimilar distributions in a high-dimensional space. As

---

<sup>3</sup>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.

discussed in Section 3.2, the HME is essentially the mirror image of likelihood weighting, which is simple importance sampling from the prior to the posterior. Sections 3.3 and 3.4 discussed two algorithms — annealed importance sampling (AIS) and sequential Monte Carlo (SMC) — which estimate marginal likelihoods through a series of much smaller importance sampling steps bridging from the prior to the posterior. The use of many small steps, rather than a single large step, typically results in far more accurate estimates of the log-ML. This suggests that, in place of the HME, one should use a series of small importance sampling steps bridging from the posterior to the prior. We now discuss two particular instantiations of this idea: reverse AIS and the sequential harmonic mean estimator.

#### 4.2.1 Reverse AIS

In Section 3.3, 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 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 simulated 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:

$$\frac{q_{for}(\mathbf{x}_1, \dots, \mathbf{x}_T)}{q_{back}(\mathbf{x}_1, \dots, \mathbf{x}_T)} = \frac{\mathcal{Z}_T}{\mathcal{Z}_1} w, \quad (27)$$

where

$$w \triangleq \frac{f_{T-1}(\mathbf{x}_{T-1}) \dots f_1(\mathbf{x}_1)}{f_T(\mathbf{x}_{T-1}) \dots f_2(\mathbf{x}_1)}. \quad (28)$$

As in Section 3.3, because Eqn. 27 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:

$$\hat{p}_{back}(\mathbf{y}) = \frac{K}{\sum_{k=1}^K w^{(k)}}. \quad (29)$$

This estimator corresponds to a stochastic upper bound on  $\log p(\mathbf{y})$ , for reasons which mirror those given in Section 4.1. By Markov’s inequality, since  $\mathbb{E}[1/\hat{p}_{back}(\mathbf{y})] = 1/p(\mathbf{y})$ ,

$$\Pr(\log \hat{p}_{back}(\mathbf{y}) < \log p(\mathbf{y}) - b) = \Pr\left(\frac{1}{\hat{p}_{back}(\mathbf{y})} > \frac{e^b}{p(\mathbf{y})}\right) < e^{-b}. \quad (30)$$

Also, by Jensen’s inequality,

$$\mathbb{E}[\log \hat{p}_{back}(\mathbf{y})] = -\mathbb{E}\left[\log \frac{1}{\hat{p}_{back}(\mathbf{y})}\right] \geq -\log \mathbb{E}\left[\frac{1}{\hat{p}_{back}(\mathbf{y})}\right] = -\log \frac{1}{p(\mathbf{y})} = \log p(\mathbf{y}) \quad (31)$$

Since this estimator is an instance of AIS, it inherits the consistency guarantees of AIS (Neal, 2001a).

---

**Algorithm 3** Sequential harmonic mean estimator (SHME)

---

```
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  $\hat{\mathcal{Z}} = \frac{K}{\sum_{k=1}^K 1/w^{(k)}}$ 
```

---

#### 4.2.2 Sequential harmonic mean estimator

It is also possible to run sequential Monte Carlo (SMC) in reverse to obtain a log-ML upper bound. We call the resulting algorithm the sequential harmonic mean estimator (SHME). Mathematically, this does not require any new ideas beyond those used in reverse AIS. As discussed in Section 3.4.1, 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 an 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, similarly to 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 SMC computes arithmetic means of the weights in the resampling step and in the final ML estimate, SHME uses the harmonic means of the weights.

As in SMC, we leave open the choice of proposal distribution  $q(\mathbf{z}_i | \mathbf{y}_i, \boldsymbol{\theta}^{(k)})$ . Possibilities include Eqns. 16 and 18 of Section 3.4. Unlike in standard SMC, the proposal distribution does not affect the sequence of states sampled in the algorithm—rather, it is used to update the weights. Proposal distributions which better match the posterior are likely to result in lower variance weights.

The naïve harmonic mean estimator has been criticized for its instability (Neal, 2008), so it is worth examining whether the same issues are relevant to SHME. One problem with the naïve HME is that it is an instance of simple importance sampling where the target distribution is more spread out than the proposal distribution. Therefore, samples corresponding to the tails of the target distribution can have extremely large importance weights. The same problem is present in SHME (though to a lesser degree) because the weight update steps are based on importance sampling where the proposal distribution is the posterior  $p(\mathbf{z}_{1:i}, \boldsymbol{\theta} | \mathbf{y}_{1:i})$ , and the target distribution is the slightly less peaked posterior  $p(\mathbf{z}_{1:i}, \boldsymbol{\theta} | \mathbf{y}_{1:i-1})$  (which conditions on one less data point). Therefore, the

individual weight updates, as well as the final output, may have large variance when the algorithm is interpreted as an estimator of  $p(\mathbf{y})$ .

Yet, when used as an estimator of  $\log p(\mathbf{y})$ , SHME can still yield accurate estimates. To justify this theoretically, suppose we have a model for which  $\boldsymbol{\theta}$  and  $\mathbf{z}_i^{(k)}$  can both be integrated out analytically in the predictive distribution  $p(\mathbf{y}_i | \mathbf{z}_{1:i-1}^{(k)}, \mathbf{y}_{1:i-1})$ . (E.g., this is the case for the clustering model we consider in our experiments.) We analyze the gap between the SMC (stochastic lower bound) and SHME (stochastic upper bound) estimates. For simplicity, assume that only a single particle is used in each algorithm, and that the MCMC transition operator yields exact samples in each step. (These correspond to assumptions made by Neal (2001a) when analyzing AIS.) In this case, the expected SMC estimate of  $\log p(\mathbf{y})$  is given by:

$$\mathbb{E}[\log \hat{p}_{\text{SMC}}(\mathbf{y})] = \sum_{i=1}^N \mathbb{E}_{p(\mathbf{z}_{1:i-1} | \mathbf{y}_{1:i-1})} [\log p(\mathbf{y}_i | \mathbf{z}_{1:i-1}, \mathbf{y}_{1:i-1})]. \quad (32)$$

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

$$\mathbb{E}[\log \hat{p}_{\text{SHME}}(\mathbf{y})] = \sum_{i=1}^N \mathbb{E}_{p(\mathbf{z}_{1:i-1} | \mathbf{y}_{1:i})} [\log p(\mathbf{y}_i | \mathbf{z}_{1:i-1}, \mathbf{y}_{1:i-1})]. \quad (33)$$

(The difference between these two equations is that the distribution in Eqn. 33 conditions on one additional data point.) Since  $\mathbb{E}[\log \hat{p}_{\text{SMC}}(\mathbf{y})] \leq \log p(\mathbf{y}) \leq \mathbb{E}[\log \hat{p}_{\text{SHME}}(\mathbf{y})]$ , we can bound the estimation error:

$$|\mathbb{E}[\log \hat{p}_{\text{SHME}}(\mathbf{y})] - \log p(\mathbf{y})| \leq \mathbb{E}[\log \hat{p}_{\text{SHME}}(\mathbf{y})] - \mathbb{E}[\log \hat{p}_{\text{SMC}}(\mathbf{y})] \quad (34)$$

$$\begin{aligned} &= \sum_{i=1}^N \mathbb{E}_{p(\mathbf{z}_{1:i-1} | \mathbf{y}_{1:i})} [\log p(\mathbf{y}_i | \mathbf{z}_{1:i-1}, \mathbf{y}_{1:i-1})] \\ &\quad - \mathbb{E}_{p(\mathbf{z}_{1:i-1} | \mathbf{y}_{1:i-1})} [\log p(\mathbf{y}_i | \mathbf{z}_{1:i-1}, \mathbf{y}_{1:i-1})]. \end{aligned} \quad (35)$$

To understand the terms in this sum, suppose we are predicting the next observation  $\mathbf{y}_i$  given our past observations. If we “cheat” by using  $\mathbf{y}_i$  to help infer the latent variables for past observations, we would expect this to improve the predictive likelihood. Each of the terms in Eqn. 35 corresponds to the magnitude of this improvement. This is, however, a very indirect way for  $\mathbf{y}_i$  to influence the model’s predictions, so arguably we should expect these terms to be relatively small. If they are indeed small, then SHME will have small error in estimating  $\log p(\mathbf{y})$ .

### 4.3 Bidirectional Monte Carlo

So far, we have discussed techniques for obtaining stochastic lower and upper bounds on the log marginal likelihood for simulated data. The stochastic lower bounds are obtained using standard sampling-based techniques, such as AIS or SMC. The stochastic upper bounds are obtained by running one of these algorithms in reverse, starting from an exact posterior sample. The methods are most useful in combination, since one can sandwich the true log-ML value between the stochastic bounds. Both AIS and SMC are consistent, in the sense that they approach the correct value in the limit of infinite computation (Neal, 2001a; del Moral et al., 2006). Therefore, it is possible to run both directions with enough computation that the two stochastic bounds agree. Once the

stochastic bounds agree closely (e.g. to within 1 nat), one has a log-ML estimate which (according to the above analysis based on Markov’s inequality) is very unlikely to be off by more than a few nats. Errors of this magnitude are typically inconsequential in the context of log-ML estimation, so one can consider the final log-ML estimate to be ground truth. We refer to this overall approach to computing ground truth log-ML values as bidirectional Monte Carlo (BDMC).

#### 4.4 Relationship with RAISE

Our BDMC method is similar in spirit to the reverse AIS estimator (Burda et al., 2015), which has been used to evaluate log-likelihoods of Markov random fields. Like BDMC, RAISE runs AIS both forwards and backwards in order to sandwich the log-likelihoods between two values. The main differences between the methods are as follows:

1. RAISE is used in the setting of Markov random fields, or undirected graphical models. Algorithms such as AIS compute stochastic lower bounds on the partition function, which correspond to stochastic upper bounds on the log-likelihood. The main difficulty was lower bounding the log-likelihoods.

BDMC is applied to models described as generative processes (and often represented with *directed* graphical models). The challenge is to integrate out parameters and latent variables in order to compute the likelihood of observations. Prior methods often returned stochastic *lower* bounds on the log-likelihood, and the technical novelty concerns stochastic *upper* bounds.

2. RAISE runs the reverse AIS chain starting from the data on which one wishes to evaluate log-likelihoods. BDMC runs the reverse chain starting from an exact posterior sample.
3. RAISE computes stochastic log-likelihood lower bounds for an *approximate* model. If the approximate model describes the data better than the original MRF, RAISE’s supposed lower bound may in fact overestimate the log-likelihood. By contrast, BDMC returns stochastic upper and lower bounds on the *original model*, so one can locate the true value between the bounds with high probability.

#### 4.5 Evaluating posterior inference

So far, the discussion has focused on measuring the accuracy of log-ML estimators. In some cases, BDMC can also be used to quantitatively measure the quality of an approximate posterior sampler on simulated data. To do this, we make use of a relationship between posterior inference and marginal likelihood estimation which holds for some sampling-based inference algorithms.

It is well known that the problems of inference and ML estimation are equivalent for variational Bayes (Section 5.1): the KL divergence between the approximate and true posteriors equals the gap between the variational lower bound and the true log-ML. A similar relationship holds for some sampling-based log-ML estimators, except that the equality may need to be replaced with an inequality. First, consider the case of simple importance sampling (Section 3.1). If  $q(\mathbf{z}, \boldsymbol{\theta})$  is the proposal distribution, then the expected log-ML estimate based on a single sample (see Eqn. 4) is given by:

$$\begin{aligned} \mathbb{E}_{q(\mathbf{z}, \boldsymbol{\theta})} [\log p(\mathbf{z}, \boldsymbol{\theta}, \mathbf{y}) - \log q(\mathbf{z}, \boldsymbol{\theta})] &= \log p(\mathbf{y}) + \mathbb{E}_{q(\mathbf{z}, \boldsymbol{\theta})} [\log p(\mathbf{z}, \boldsymbol{\theta} | \mathbf{y}) - \log q(\mathbf{z}, \boldsymbol{\theta})] \\ &= \log p(\mathbf{y}) - D_{\text{KL}}(q(\mathbf{z}, \boldsymbol{\theta}) \| p(\mathbf{z}, \boldsymbol{\theta} | \mathbf{y})). \end{aligned} \tag{36}$$

Interestingly, this formula is identical to the variational Bayes lower bound when  $q$  is used as the approximating distribution.

We have discussed two sampling-based ML estimators which can be seen as importance sampling on an extended state space: AIS (Section 3.3) and SMC with a single particle (Section 3.4.1). 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:

$$\begin{aligned} \mathbb{E}[\log \hat{p}(\mathbf{y})] &= \mathbb{E}_{q(\mathbf{z}, \boldsymbol{\theta}, \mathbf{v})} [\log p(\mathbf{z}, \boldsymbol{\theta}, \mathbf{v}, \mathbf{y}) - \log q(\mathbf{z}, \boldsymbol{\theta}, \mathbf{v})] \\ &= \log p(\mathbf{y}) - \text{D}_{\text{KL}}(q(\mathbf{z}, \boldsymbol{\theta}, \mathbf{v}) \parallel p(\mathbf{z}, \boldsymbol{\theta}, \mathbf{v} | \mathbf{y})) \\ &\leq \log p(\mathbf{y}) - \text{D}_{\text{KL}}(q(\mathbf{z}, \boldsymbol{\theta}) \parallel p(\mathbf{z}, \boldsymbol{\theta} | \mathbf{y})). \end{aligned} \tag{37}$$

This implies that the KL divergence of the approximate posterior samples from the true posterior is bounded by the bias of the log-ML estimator. Eqn. 37, in conjunction with BDMC, can be used to demonstrate the accuracy of posterior samples on simulated datasets. In particular, to measure the accuracy of AIS or SMC, one can compute the gap between its log-ML estimate and the stochastic upper bound from BDMC. This gap will be a stochastic upper bound on the KL divergence of the distribution of approximate samples from the true posterior.

## 5 Other marginal likelihood estimators

In this section, we overview some additional ML estimation algorithms not discussed in Section 3.

### 5.1 Variational Bayes

All of the methods described above are sampling-based estimators. Variational Bayes (Hinton and van Camp, 1993; Waterhouse et al., 1996; Attias, 2000; Ghahramani and Beal, 2001) 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 constrained to be independent, *i.e.*  $q(\mathbf{z}, \boldsymbol{\theta}) = q(\mathbf{z})q(\boldsymbol{\theta})$ , and the two factors may themselves have additional factorization assumptions. The objective function being maximized is the following:

$$\mathcal{F}(q) \triangleq \mathbb{E}_{q(\mathbf{z}, \boldsymbol{\theta})} [\log p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{y})] + \mathcal{H} [q(\mathbf{z}, \boldsymbol{\theta})], \tag{38}$$

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 Eqn 38 are:

$$q(\mathbf{z}) \propto \exp (\mathbb{E}_{q(\boldsymbol{\theta})} [\log p(\mathbf{z}, \boldsymbol{\theta}, \mathbf{y})]) \tag{39}$$

$$q(\boldsymbol{\theta}) \propto \exp (\mathbb{E}_{q(\mathbf{z})} [\log p(\mathbf{z}, \boldsymbol{\theta}, \mathbf{y})]) \tag{40}$$

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

$$\log \mathcal{F}(q) = \log p(\mathbf{y}) - \text{D}_{\text{KL}}(q(\mathbf{z}, \boldsymbol{\theta}) \parallel p(\mathbf{z}, \boldsymbol{\theta} | \mathbf{y})). \tag{41}$$

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

## 5.2 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

$$p(\mathbf{y}) = \frac{p(\mathbf{z}^*, \boldsymbol{\theta}^*, \mathbf{y})}{p(\mathbf{z}^*, \boldsymbol{\theta}^* | \mathbf{y})} \quad (42)$$

for any particular values  $(\mathbf{z}^*, \boldsymbol{\theta}^*)$  of the latent variables and parameters. While (42) holds for any choice of  $(\mathbf{z}^*, \boldsymbol{\theta}^*)$ , they are usually taken to be high probability locations, such as the maximum a posteriori (MAP) estimate. The numerator can generally be computed from the model definition. The denominator is based on a Monte Carlo estimate of the conditional probability obtained from posterior samples  $(\mathbf{z}^{(1)}, \boldsymbol{\theta}^{(1)}), \dots, (\mathbf{z}^{(K)}, \boldsymbol{\theta}^{(K)})$ . 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:

$$\hat{p}(\mathbf{z}^*, \boldsymbol{\theta}^* | \mathbf{y}) = \frac{1}{K} \sum_{k=1}^K \mathcal{T}(\mathbf{z}^*, \boldsymbol{\theta}^* | \mathbf{z}^{(k)}, \boldsymbol{\theta}^{(k)}, \mathbf{y}). \quad (43)$$

How should the estimator be expected to perform? Observe that if exact samples are used, (43) is an unbiased estimate of the denominator of (42). Therefore, following the analysis of Section 4.1, 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  $\tilde{\mathcal{T}}(\mathbf{z}, \boldsymbol{\theta} | \mathbf{z}^*, \boldsymbol{\theta}^*)$ , where

$$\tilde{\mathcal{T}}(\mathbf{z}', \boldsymbol{\theta}' | \mathbf{z}, \boldsymbol{\theta}) \triangleq \frac{\mathcal{T}(\mathbf{z}, \boldsymbol{\theta} | \mathbf{z}', \boldsymbol{\theta}') p(\mathbf{z}', \boldsymbol{\theta}' | \mathbf{y})}{\sum_{\mathbf{z}', \boldsymbol{\theta}'} \mathcal{T}(\mathbf{z}, \boldsymbol{\theta} | \mathbf{z}', \boldsymbol{\theta}') p(\mathbf{z}', \boldsymbol{\theta}' | \mathbf{y})} \quad (44)$$

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

## 5.3 Nested sampling

Nested sampling (NS; Skilling, 2006) is a marginal likelihood estimator which, like AIS, samples from a sequence of distributions where the strength of the evidence is gradually amplified. In this section, we denote the state as  $\mathbf{x} = (\boldsymbol{\theta}, \mathbf{z})$ , the prior as  $\pi(\mathbf{x}) = p(\boldsymbol{\theta}, \mathbf{z})$ , and the likelihood as  $L(\mathbf{x}) = p(\mathbf{y} | \boldsymbol{\theta}, \mathbf{z})$ . Central to the method is the notion of the *constrained prior*, which is the prior



distribution restricted to a region of high likelihood:

$$\check{\pi}_C(\mathbf{x}) \triangleq \begin{cases} \pi(\mathbf{x})/V(C) & L(\mathbf{x}) > C \\ 0 & L(\mathbf{x}) \leq C \end{cases} \quad (45)$$

$$V(C) \triangleq \pi(\{\mathbf{x} : L(\mathbf{x}) > C\}). \quad (46)$$

The *prior volume*  $V(C)$  is the fraction of the prior probability mass which lies within the likelihood constraint. For simplicity, we assume the set  $\{\mathbf{x} : L(\mathbf{x}) = C\}$  has measure zero for any  $C$ . Each step of NS attempts to sample from a constrained prior, where the cutoff  $C$  is increased (and hence the volume  $V(C)$  is decreased) at a controlled rate.

We first describe the idealized version of NS, where one assumes an oracle that returns an exact sample from the constrained prior. One begins with a set of  $K$  particles  $\{\mathbf{x}^{(k)}\}_{k=1}^K$  drawn *i.i.d.* from the prior  $\pi$ , with  $K \geq 2$ . In each step (indexed by  $t$ ), one chooses the particle with the smallest likelihood; call its index  $k_\star \triangleq \arg \min_k L(\mathbf{x}^{(k)})$ . The likelihood cutoff is updated to  $C_t = L(\mathbf{x}^{(k_\star)})$ . The particle  $\mathbf{x}^{(k_\star)}$  is then replaced with a sample from the constrained prior  $\check{\pi}_{C_t}$ . All of the other particles remain untouched. This process is repeated until a stopping criterion (described below) is met.

After step  $t$  of the algorithm, the  $K$  particles are independently distributed according to  $\check{\pi}_{C_t}$ . The prior volumes  $V(L(\mathbf{x}^{(k)}))$  are therefore uniformly distributed on the interval  $[0, C_t]$ . (By convention,  $C_0 = 0$ .) The particle  $\mathbf{x}^{(k_\star)}$  (which was chosen to have the minimum likelihood) has a prior volume of approximately  $V(C_t) \cdot K/(K+1)$ . Hence, the prior volume is expected to decrease by roughly a factor of  $K/(K+1)$  in each iteration. Since  $V(C_0) = V(0) = 1$ , this implies

$$V(C_t) \approx \left( \frac{K}{K+1} \right)^t. \quad (47)$$

Now, observe that for each  $t$ ,  $V(C_t) - V(C_{t+1})$  fraction of the prior volume has a likelihood value between  $C_t$  and  $C_{t+1}$ . Since the marginal likelihood is given by  $p(\mathbf{y}) = \int \pi(\mathbf{x})L(\mathbf{x}) d\mathbf{x}$ , we can bound the marginal likelihood above and below:

$$\sum_{t=0}^{\infty} (V(C_t) - V(C_{t+1})) C_t \leq p(\mathbf{y}) \leq \sum_{t=0}^{\infty} (V(C_t) - V(C_{t+1})) C_{t+1}. \quad (48)$$

The true volumes  $V(C_t)$  are unknown, but one can achieve a good approximation by plugging in Eqn 47. As a stopping criterion, one typically stops when the next term in the summation increases the total by less than a pre-specified ratio (such as  $1 + 10^{-10}$ ).

The idealized algorithm requires the ability to sample exactly from the constrained prior. In practice, this is typically intractable or inefficient. Instead, one tries to approximately sample from the constrained prior using the following procedure: first replace  $\mathbf{x}^{(k_\star)}$  with one of the other particles, chosen uniformly at random. Apply one or more steps of MCMC, where the transition operator has the constrained prior as its stationary distribution. If the transition operator mixes fast enough, then this should be a reasonable approximate sample from the constrained prior.

Like AIS, NS has the interpretation of moving through a series of distributions where the evidence gradually becomes stronger. One of the arguments made for NS, in contrast with AIS, is that its schedule for moving through its space of distributions is potentially more stable (Skilling, 2006). In particular, AIS has been observed to suffer from phase transitions: around some critical

temperature, the distribution may suddenly change from being very broad to very peaked. If the particles do not quickly find their way into the peak, the results might be inaccurate. In NS, the prior volume decreases at a controlled rate, so this sort of phase transition is impossible. This effect was shown to improve the stability of partition function estimation for Potts models (Murray et al., 2005).

On the flip side, NS does not have quite as strong a theoretical guarantee as AIS. AIS is guaranteed to be an unbiased estimator of the ML, even when MCMC operators (rather than exact samples) are used in each step. By contrast, the theoretical analysis of the variance of NS (Skilling, 2006) assumes exact samples in each step. Indeed, in our own experiments, we found that while NS tended to underestimate the ML on average (similarly to AIS), it overestimated the true value too often for it to be a stochastic lower bound in the sense of Section 4.1.

## 6 Experiments

In this section, we use our proposed bidirectional Monte Carlo technique to evaluate a wide variety of ML estimators on several latent variable models. In particular, we consider the following models:

- **Clustering.** Roughly speaking, this model is a Bayesian analogue of K-means. Each data point is assumed to be drawn from one of  $K$  mixture components. Each mixture component is associated with a spherical Gaussian distribution whose variance is fixed but whose mean is unknown. Mathematically,

$$\begin{aligned} z_i &\sim \text{Multinomial}(\boldsymbol{\pi}) \\ \theta_{kj} &\sim \mathcal{N}(0, \sigma_\theta^2) \\ y_{ij} &\sim \mathcal{N}(\theta_{z_i, j}, \sigma_n^2). \end{aligned}$$

The mixture probabilities  $\boldsymbol{\pi}$ , the between-cluster variance  $\sigma_\theta^2$ , and the within-cluster variance  $\sigma_n^2$  are all fixed. In the matrix decomposition grammar of Grosse et al. (2012), this model would be written as  $\text{MG} + \text{G}$ .

- **Low-rank approximation.** In this model, we approximate an  $N \times D$  matrix  $\mathbf{Y}$  as a low rank matrix plus Gaussian noise. In particular, we approximate it with the product  $\mathbf{UV}$ , where  $\mathbf{U}$  and  $\mathbf{V}$  are matrices of size  $N \times K$  and  $K \times D$ , respectively. We assume  $K < \min(N, D)$ , so the product has rank  $K$ . We assume a spherical Gaussian observation model, as well as spherical Gaussian priors on the components  $\mathbf{U}$  and  $\mathbf{V}$ . More precisely,

$$\begin{aligned} u_{ik} &\sim \mathcal{N}(0, \sigma_u^2) \\ v_{kj} &\sim \mathcal{N}(0, \sigma_v^2) \\ y_{ij} &\sim \mathcal{N}(\mathbf{u}_i^T \mathbf{v}_j, \sigma_n^2). \end{aligned}$$

The prior variances  $\sigma_u^2$  and  $\sigma_v^2$  and noise variance  $\sigma_n^2$  are all fixed. This model is roughly equivalent to probabilistic matrix factorization (Salakhutdinov and Mnih, 2008), except that in our experiments,  $\mathbf{Y}$  is fully observed. While the model is symmetric with respect to rows and columns, we follow the convention where rows of  $\mathbf{Y}$  represent data points. In this setup, we can think of the component  $\mathbf{V}$  as the parameters of the model and  $\mathbf{U}$  as the latent variables. In the grammar of Grosse et al. (2012), this model would be written as  $\text{GG} + \text{G}$ .

- **Binary attributes.** This model assumes each data point can be described in terms of  $K$  binary-valued attributes. In particular, we approximate the observation matrix  $\mathbf{Y}$  with the product  $\mathbf{Z}\mathbf{A}$ , where  $\mathbf{Z}$  is an  $N \times K$  binary-valued matrix and  $\mathbf{A}$  is a  $K \times D$  real-valued matrix. Each row of  $\mathbf{Y}$  corresponds to a single data point. Each row of  $\mathbf{Z}$  can be thought of as a latent vector explaining a data point, and each row of  $\mathbf{A}$  can be thought of as a set of parameters describing the effect one of the binary attributes has on the observations. Mathematically, this model is defined as

$$\begin{aligned} z_{ik} &\sim \text{Bernoulli}(\pi_k) \\ a_{kj} &\sim \mathcal{N}(0, \sigma_a^2) \\ y_{ij} &\sim \mathcal{N}\left(\sum_k z_{ik} a_{kj}, \sigma_n^2\right). \end{aligned}$$

The attribute probabilities  $\{\pi_k\}$ , feature variance  $\sigma_a^2$ , and noise variance  $\sigma_n^2$  are all fixed. This model is related to the Indian buffet process linear-Gaussian model (Griffiths and Ghahramani, 2005), with the important difference that both the number of attributes and the probability of each one are fixed. In the grammar of Grosse et al. (2012), this model would be written as BG + G.

We note that all of the models described above have hyperparameters, such as mixture probabilities or noise variance. In a practical setting, these hyperparameters are typically unknown. One would typically include them as part of the model, using appropriate priors, and attempt to infer them jointly with the model parameters and latent variables. Unfortunately, this does not work well in our setting, due to our need to simulate data from the model. One ordinarily assigns weak priors to the hyperparameters, but sampling from such priors usually results in pathological datasets where the structure is either too weak to detect or so strong that it is trivial to find the correct explanation. To avoid these pathologies, we assigned fixed values to the hyperparameters, and we chose these values such that the posterior distribution captures most of the structure, but is not concentrated on a single explanation.

We evaluated the following ML estimators on all three of these models:

- the Bayesian information criterion (BIC)
- likelihood weighting (Section 3.1)
- the harmonic mean estimator (HME) (Section 3.2), using a Markov chain starting from the exact sample
- annealed importance sampling (AIS) (Section 3.3)
- sequential Monte Carlo (SMC), using a single particle (Section 3.4)
- variational Bayes (Section 5.1). 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 5.2)
- nested sampling (Section 5.3)

## 6.1 Implementation

In order to make the running times of different algorithms directly comparable, the implementations share the same MCMC transition operators wherever possible. 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 require 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. It also requires computing the transition probabilities between any pair of states. The latter imposes a nontrivial constraint on the choice of MCMC operators, in that it disallows operators which compute auxiliary variables.
- Nested sampling requires an MCMC operator whose stationary distribution is the constrained prior.
- Unlike the other algorithms, variational Bayes is based on optimization, rather than sampling. In our implementation, the updates all involved optimizing one of the component distributions given the others.

For all three models, the MCMC transition operator was a form of Gibbs sampling. Here are some more model-specific details:

- **Clustering.** 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.** Each of the two factors  $\mathbf{U}$  and  $\mathbf{V}$  was resampled as a block. For computing predictive likelihood,  $\mathbf{V}$  was sampled from the posterior, and the  $\mathbf{U}$  was marginalized out analytically.
- **Binary attributes.** The feature matrix  $\mathbf{A}$  was collapsed out wherever possible, and the tricks of Doshi-Velez and Ghahramani (2009) were used to efficiently update the posterior distribution over  $\mathbf{A}$ .

ML estimators are notoriously difficult to implement correctly, as small bugs can sometimes lead to large errors in the outputs without any obvious indication that something is amiss. (Indeed, checking correctness of MCMC samplers and ML estimators is one potential application of this work.) Appendix B discusses in detail our approach to testing the correctness of the implementations of our algorithms.

In general, we face a tradeoff between performance and difficulty of implementation. Therefore, it is worth discussing the relative difficulty of implementing 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.<sup>4</sup> 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 required somewhat more math to derive the update rules. However, it was considerably simpler to test (see Appendix B). 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.

## 6.2 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  $\sigma$  denotes the logistic sigmoid function and  $\delta$  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  $(\theta^*, \mathbf{z}^*)$  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.

---

<sup>4</sup>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.

- **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.<sup>5</sup> As discussed in Section 6.4, there was little qualitative difference between the two conditions.

### 6.3 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:

$\log_{10} p_1(\mathbf{y}) - \log_{10} p_2(\mathbf{y})$	$p_1(\mathbf{y})/p_2(\mathbf{y})$	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, there is an empirical yardstick we can use, namely comparing the ML scores of different models fit to the same datasets. Table 1 shows the ML estimates for all three models under consideration, on all three of the simulated datasets.<sup>6</sup> The estimates were obtained from AIS with

<sup>5</sup>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.

<sup>6</sup>Unlike in the rest of our experiments, it did not make sense to freeze the model hyperparameters for the cross-model ML evaluation, as there is no “correct” choice of hyperparameters when the model is wrong. Therefore, for the results in this table, we included the hyperparameters in the model, and these were integrated out along with the parameters and latent variables. We used standard priors for hyperparameters: inverse gamma distributions for variances, Dirichlet distributions for mixture probabilities, and beta distributions for Bernoulli probability parameters. AIS was generally able to infer the correct hyperparameter values in this experiment.

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

Table 1: Marginal likelihood scores for all three models evaluated on simulated data drawn from all three models. **Rows:** the model used to generate the simulated 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.

30,000 intermediate distributions.<sup>7</sup> These 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.

## 6.4 Results

All of the ML estimation algorithms were run on all three of the models. A simulated 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 4.1, the ground truth value is accurate to within a few nats with high probability.

As mentioned in Section 6.2, 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 mean squared error (MSE) 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 6.3 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 root mean squared error (RMSE) of less than 10 nats, and how much time they required to do so.

**Clustering.** The results for the clustering model are shown in Figures 2 and 3. Figure 2 shows the log-ML estimates for all estimators, while Figure 3 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 the desired 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.

<sup>7</sup>The entries in this table are guaranteed only to be stochastic lower bounds. However, AIS with 30,000 intermediate distributions yielded accurate estimates in all comparisons against ground truth (see Section 6.4).

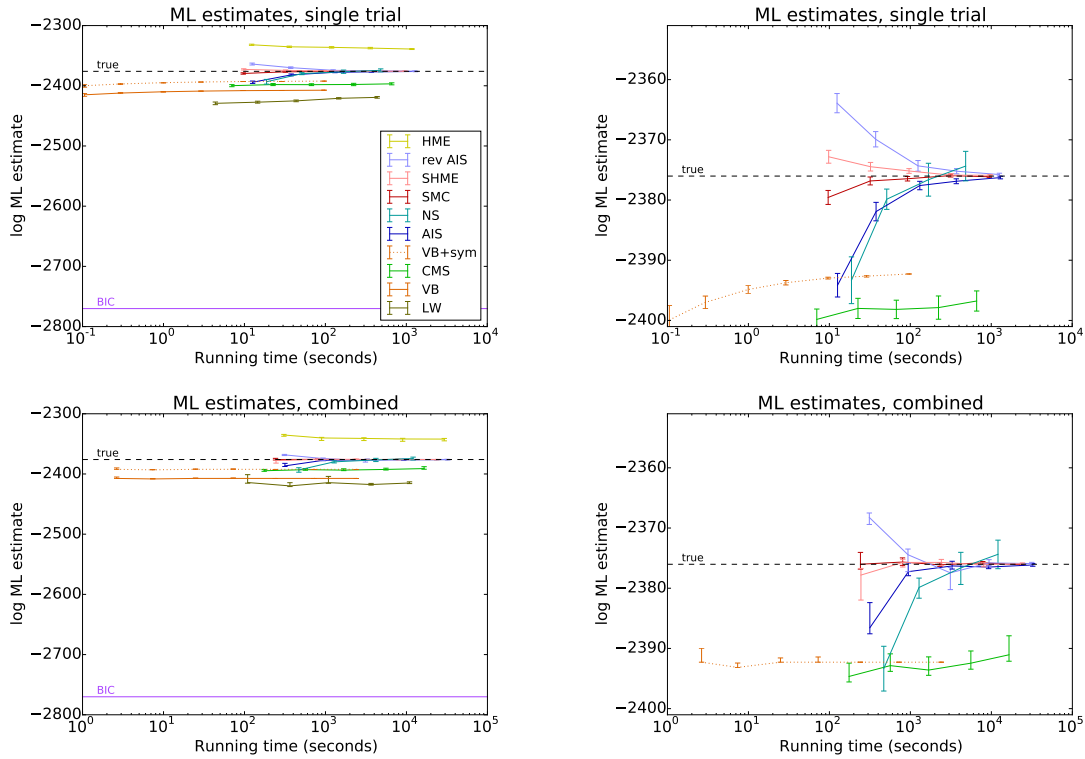


Figure 2: Comparison of marginal likelihood estimators on the clustering model. **Top:** average log-ML estimates for each of the 25 individual trials. (The right-hand figure is zoomed in.) **Bottom:** average log-ML estimates combined between the 25 trials. (The right-hand figure is zoomed in.) Note 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. Confidence intervals are given for the *expected log-ML estimate* for a given estimator. (They are not confidence intervals for the log-ML itself, so it is not problematic that they generally do not cover the true log-ML.)



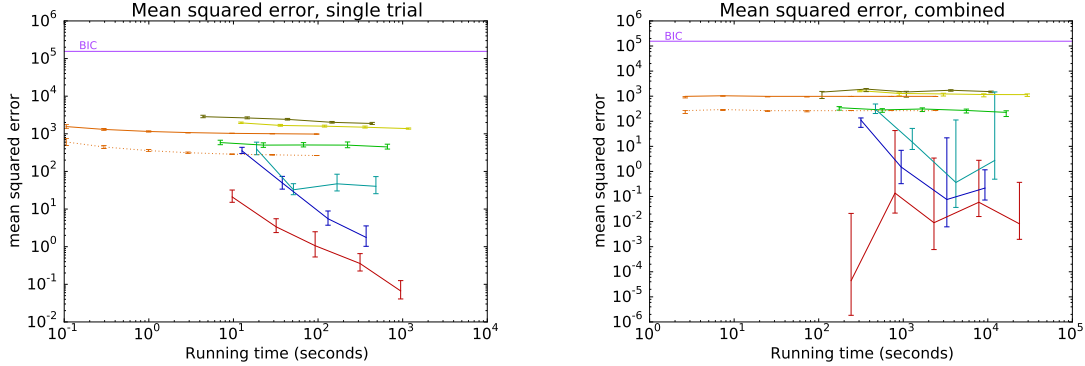


Figure 3: Mean squared error relative to ground truth for individual trials (left) and combined estimates (right) for the clustering model. See Figure 2 for the legend.

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 these methods 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 model classes. In order to determine if the gap was due to local optima, we ran an additional experiment where we gave VB a “hint” by initializing it to a point estimate on the sample which generated the data. In this experiment (as well as for the low rank and binary attribute models), VB with random initializations was able to find the same optimum, or a slightly better one, compared with the condition where it was given the hint. This suggests 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 low rank and binary attribute models.) Roughly speaking, if one has a fixed computational budget, it is better 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 Grosse et al. (2013), high-accuracy results were often obtained by averaging over many AIS runs, even though a large fraction of individual runs gave inaccurate estimates.)

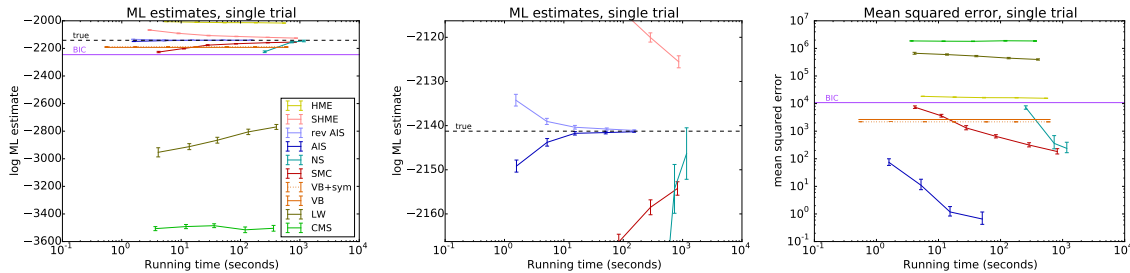


Figure 4: Comparison of marginal likelihood estimators on the low rank model. **Left:** average log-ML estimates across the 25 trials. **Middle:** same as left, but zoomed in. **Right:** MSE of individual samples. See Figure 2 for the abbreviation key.

**Low rank.** The results on the low rank factorization 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.

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  $p(\mathbf{U}^*, \mathbf{V}^* | \mathbf{Y})$  was estimated using an MCMC chain starting close to the point estimate  $p(\mathbf{U}^*, \mathbf{V}^*)$ . The model has a large space of symmetries which the Markov chain explored slowly, because  $\mathbf{U}$  and  $\mathbf{V}$  were tightly coupled. Therefore, the first few samples dramatically overestimated the probability of transitioning to  $(\mathbf{U}^*, \mathbf{V}^*)$ , and it was impossible for later samples to cancel out this bias because the transition probabilities were averaged arithmetically.

In general, variational Bayes is also known to have a similar difficulty when tightly coupled variables are constrained to be independent in the variational approximation. Interestingly, in this experiment, 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 are shown in Figure 5. 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.

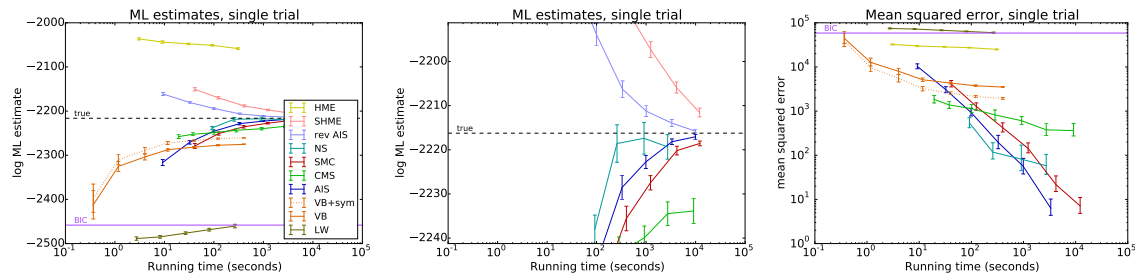


Figure 5: Comparison of marginal likelihood estimators on the binary attribute model. **Left:** average log-ML estimates across the 25 trials. **Middle:** same as left, but zoomed in. **Right:** MSE of individual samples. See Figure 2 for the abbreviation key.

## 7 Discussion

Based on our experiments, we observe that no single algorithm consistently dominated the others. The relative performance of different estimators varied tremendously depending on the model class, and it seems likely that other factors, such as the signal-to-noise ratio or the number of data points, could make a big difference. More work is required to understand which algorithms perform well on which models and why. However, we can draw some tentative recommendations from our results. As a general rule of thumb, we would suggest trying AIS first, 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. Likelihood weighting, the harmonic mean estimator, and the BIC are unlikely to give accurate results. These recommendations are consistent with the folklore in the field, and it is reassuring that we can now support them with quantitative evidence.

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 3.4.1). Any algorithm which shares this structure can be reversed using our proposed technique to obtain a stochastic upper bound on the log-ML of simulated data. Therefore, if better algorithms are devised which build upon AIS and SMC (either separately or in combination), they can automatically be used in the context of BDMC to obtain more precise ground truth log-ML values on simulated data.

We believe the ability to rigorously and quantitatively evaluate algorithms is what enables us to improve them. In many application areas of machine learning, especially supervised learning, benchmark datasets have spurred rapid progress in developing new algorithms and clever refinements to existing algorithms. One can select hyperparameters, such as learning rates or the number of units in a neural network, by quantitatively measuring performance on held-out validation data. This process is beginning to be automated through Bayesian optimization (Snoek et al., 2012). So far, the lack of quantitative performance evaluations in marginal likelihood estimation, and in sampling-based inference more generally, has left us fumbling around in the dark. ML estimators often involve design choices such as annealing schedules or which variables to collapse out. Just as careful choices of learning rates and nonlinearities can be crucial to the performance of a neural network, similar algorithmic hyperparameters and engineering choices may be crucial to building effective samplers and marginal likelihood estimators. We hope the framework we have presented for quantitatively evaluating ML estimators will enable the same sort of rapid progress in posterior

inference and ML estimation which we have grown accustomed to in supervised learning.

## References

- H. Attias. A variational Bayesian framework for graphical models. In *Neural Inf. Proc. Systems*, 2000.
- A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 1998.
- 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.
- Y. Burda, R. B. Grosse, and R. Salakhutdinov. Accurate and conservative estimates of MRF log-likelihood using reverse annealing. In *Artificial Intelligence and Statistics*, 2015.
- 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.
- 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.
- Finale Doshi-Velez and Zoubin Ghahramani. Accelerated sampling for the Indian buffet process. In *Int'l. Conf. on Machine Learning*, 2009.
- 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.
- 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.
- Z. Ghahramani and M. J. Beal. Propagation algorithms for variational Bayesian learning. In *Neural Inf. Proc. Systems*, 2001.
- V. Gogate, B. Bidyuk, and R. Dechter. Studies in lower bounding probability of evidence using the Markov inequality. In *Uncertainty in Artificial Intelligence*, 2007.
- 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. In *Neural Information Processing Systems*, 2005.

- R. B. Grosse and D. K. Duvenaud. Testing MCMC code. In *Neural Information Processing Systems Workshop on Software Engineering for Machine Learning*, 2014.
- R. B. Grosse, C. J. Maddison, and R. Salakhutdinov. Annealing between distributions by averaging moments. In *Neural Information Processing Systems*, 2013.
- 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.
- 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 D. van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *Computational Learning Theory*, 1993.
- M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22(5):1087–1116, 1992.
- 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.
- D. J. C. MacKay. Comparison of approximate methods for handling hyperparameters. *Neural Computation*, 11(5):1035–1068, 1999.
- DJC MacKay. Bayesian interpolation. *Neural Computation*, 1992.
- X. Meng and W. H. Wong. Simulating ratios of normalizing constants via a simple identity: a theoretical exploration. *Statistica Sinica*, 6:831–860, 1996.
- I. Murray and R. Salakhutdinov. On the quantitative analysis of deep belief networks. In *International Conference on Machine Learning*, 2008.
- I. Murray and R. Salakhutdinov. Evaluating probabilities under high-dimensional latent variable models. In *Neural Information Processing Systems*, 2009.
- I. Murray, D. J. C. MacKay, Z. Ghahramani, and J. Skilling. Nested sampling for Potts models. In *Neural Information Processing Systems*, 2005.
- 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. 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/>.
- 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.
- J. G. Propp and D. B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 1996.
- 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.
- Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, 2008.
- 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.
- M. Teyssier and D. Koller. Ordering-based search: a simple and effective algorithm for learning Bayesian networks. In *UAI*, 2005.
- M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. In *Uncertainty in Artificial Intelligence*, 2002.
- H. M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno. Evaluation methods for topic models. In *Int’l. Conf. on Machine Learning*, 2009.
- S. Waterhouse, D. MacKay, and T. Robinson. Bayesian methods for mixture of experts. In *Neural Information Processing Systems*, 1996.

## A Caveats about marginal likelihood

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 (as used to motivate Bayesian model comparison) may not be meaningful if none of the models under consideration 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 caveat should be kept in mind when applying the methods of this paper: since the estimators are evaluated on simulated data, the results might not be representative of practical situations if the model is a poor match to the data. In general, 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).

Another frequent 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 in the big data setting, because with enough data, the likelihood function will overwhelm the prior. By contrast, the ML can be significantly hurt by a poor choice of hyperparameters, even for arbitrarily large datasets. This sensitivity to hyperparameters 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 problem is not limited to the practice of explicitly comparing

models by their ML: it also applies to the (more common) practice of tuning model complexity as part of posterior inference, for instance using reversible jump MCMC (Green, 1995) or Bayesian nonparametrics (Ghahramani, 2012). Just as with explicit ML comparisons, these techniques can suffer from the bias towards simple models when the priors are misspecified.

Several techniques have been proposed which aim to alleviate the problem of hyperparameter sensitivity. Berger and Pericchi (1996) proposed the *intrinsic Bayes factor*, which is 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 hyperparameters 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 alternative approaches require computing high-dimensional integrals over model parameters and possibly latent variables, and these integrals closely resemble the ones needed for ML. Therefore, one will run into the same computational obstacles as in computing ML, and the techniques of this paper will still be relevant.

## B Testing correctness of the implementation

ML estimators are notoriously difficult to implement correctly, for several reasons. First, an ML estimator returns a scalar value, and it’s not clear how to recognize if that value is far off. This is in contrast with supervised learning, where one can spot if the algorithm is making silly predictions, or much unsupervised learning, where it is apparent when the algorithm fails to learn important structure in the data. 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, we believe ML estimation presents challenges for testing which are unusual in the field of machine learning. In this section, we discuss methods for testing mathematical correctness of ML estimator implementations.

In this work, we used several strategies, which we recommend following in any work involving ML estimation. Grosse and Duvenaud (2014) discuss some of these techniques in more detail.

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

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

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 of 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; this 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, we recommend that it be run 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 easy problem instances: in particular, instances with extremely small or large single-to-noise ratios (SNR), or small numbers of data points.

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 effects that the experiments of this paper are intended to measure.