A Hybrid Approach for Probabilistic Inference using Random Projections

Michael H. Zhu MHZHU@CS.STANFORD.EDU
Stefano Ermon ERMON@CS.STANFORD.EDU
Stanford University, Stanford, CA 94305 USA

Abstract

We introduce a new meta-algorithm for probabilistic inference in graphical models based on random projections. The key idea is to use approximate inference algorithms for an (exponentially) large number of samples, obtained by randomly projecting the original statistical model using universal hash functions. In the case where the approximate inference algorithm is a variational approximation, this approach can be viewed as interpolating between sampling-based and variational techniques. The number of samples used controls the trade-off between the accuracy of the approximate inference algorithm and the variance of the estimator. We show empirically that by using random projections, we can improve the accuracy of common approximate inference algorithms.

1. Introduction

Sampling based techniques (Andrieu et al., 2003; Jerrum & Sinclair, 1997; Madras, 2002) and variational approaches (Jordan et al., 1999; Wainwright & Jordan, 2008) are the two main families of probabilistic inference techniques. Sampling based techniques answer queries by looking at a small number of representative samples from the entire exponentially large state space. Variational techniques, on the other hand, take a more global view and attempt to *globally* approximate a complex probability distribution using a family of more tractable ones. These two approaches have advantages and disadvantages with respect to each other, and neither approach dominates the other.

In this paper, we introduce a new meta-algorithm for probabilistic inference in graphical models. Our approach can be seen as a general sampling scheme based on universal hashing, and can be used in combination with any partition function or marginal probability approximation scheme, including both Markov Chain Monte Carlo (MCMC) sampling and variational approaches. The key idea is to consider a number of samples which can be extremely large, potentially *exponential* in the dimensionality (number of variables in a graphical model). This seemingly impossible task (simply enumerating the samples would take exponential space) can be achieved using powerful universal hashing techniques (Goldreich, 2011; Vadhan, 2011; Ermon et al., 2013b; Chakraborty et al., 2013), which can be used to represent the set of samples in an *implicit* and *compact* way. In particular, we exploit randomness amplification ideas to generate an exponentially large number of random variables (one for each possible state of the world), using as input a limited amount of randomness. This process can also be interpreted as applying a random projection (perturbation) of the original graphical model (Tarlow et al., 2012; Papandreou & Yuille, 2011; Hazan & Jaakkola, 2012; Gane et al., 2014; Maddison et al., 2014; Chakraborty et al., 2014), where each possible state of the world is selected with some probability (or more generally, its likelihood is perturbed by a random amount). Since in general we cannot enumerate such a large number of resulting samples, we use approximate inference techniques to estimate their statistical properties (such as marginals or partition function). The quality of the approximate inference for the samples can be substantially better than the one obtained for the entire state space. For example, in the extreme case where the number of samples is small enough that enumeration is possible, exact inference can be performed.

In the case where the approximate inference algorithm is a variational approximation, this hybrid approach can be seen as smoothly interpolating between a pure sampling strategy and a pure variational one. Intuitively, if we choose a small enough number of samples, we can avoid the approximation entirely by enumerating the samples (as in traditional Monte Carlo sampling methods). At the other side of the spectrum, we can use the entire state space as our sample space (no sampling) obtaining the original variational approximation. Our method allows for a broad spectrum of possibilities in between these two extremes.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

Additional properties of the approximate inference algorithm carry over to our hybrid scheme. For example, if the inference algorithm computes the exact partition function of a graphical model (e.g., in the case of low treewidth or small number of samples), our sampling-based estimator is provably unbiased. If the approximate inference algorithm provides lower (or upper) bounds for the partition function or probability of evidence queries, our estimator provides a lower (or upper) bound *in expectation*. The key trade-off involved is in terms of the accuracy of the approximate inference algorithm vs. the variance of the estimator. In fact, the number of samples used and statistical properties of the hash function (essentially, the clash probability) affect both the variance of the estimator and the accuracy of the approximate inference algorithm. We explore this trade-off experimentally and show that for several Ising models and real world datasets from genetics and medical diagnosis we can substantially improve the accuracy of the approximate inference algorithm by averaging over randomly subsampled models.

2. Problem Setup

Given an undirected graphical model with n binary variables[1], let $\mathcal{X} = \{0,1\}^n$ be the set of all possible configurations (variable assignments or possible states of the world). Define a weight function $w : \mathcal{X} \to \mathbb{R}^+$ that assigns to each configuration x a score proportional to its probability $p(x)$: $w(x) = \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\{x\}_\alpha)$. The *partition function* of the model Z is defined as $Z = \sum_{x \in \mathcal{X}} w(x) = \sum_{x \in \mathcal{X}} \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\{x\}_\alpha)$ so that $p(x) = w(x)/Z$. Computing Z is typically intractable because it involves a sum over an exponential number of configurations, and is often the most challenging inference task for many families of graphical models. Computing Z is however needed for many inference and learning tasks, such as evaluating the likelihood of data for a given model, computing marginal probabilities, and parameter estimation (Wainwright & Jordan, 2008; Koller & Friedman, 2009).

Given that probabilistic inference problems are intractable in the worst case (Roth, 1996), a number of approximate inference algorithms have been developed. There are two main families of algorithms: Monte Carlo sampling techniques and variational approximations. Since we will utilize these approaches in this paper, we give a very brief overview. We refer the reader to standard references for more details (Koller & Friedman, 2009).

[1]Our approach applies more generally to discrete graphical models. We restrict ourselves to binary variables for the ease of exposition.

2.1. Sampling

The simplest (naive) approach is to sample $x_1, \cdots, x_M$ uniformly from $\mathcal{X}$, and estimate $\widehat{Z} = \frac{1}{M}\sum_{i=1}^{M} w(x_i)2^n$. This is an unbiased estimator since $\mathbb{E}[\widehat{Z}] = \frac{1}{M}\sum_{i=1}^{M}\sum_{x \in \mathcal{X}} \frac{1}{2^n}2^n w(x) = Z$. The variance of this estimator can be very large since we are limited to a small number of samples M, while the number of configurations is exponential in n. The variance can be reduced using *importance sampling* techniques, i.e. sampling using a proposal distribution (which is closer to $p(x)$) rather than uniformly (Andrieu et al., 2003; Jerrum & Sinclair, 1997; Madras, 2002). Unfortunately, it is usually the case that the closer the proposal distribution is to the original intractable $p(x)$, the harder it gets to sample from it.

Markov Chain Monte Carlo sampling is another leading sampling method. The key idea is to draw proper representative samples from $p(x)$ by setting up a Markov Chain over the entire state space which has to reach an equilibrium distribution. For many statistical models of interest, reaching the equilibrium distribution will require simulating the chain for a number of steps which is exponential in the dimensionality (number of variables). Unless there are special regularity conditions, if the random walk does not visit all the possible states it might miss some important parts. In practice, the approach will therefore only give approximate answers. There is generally little or no information on the quality of the approximation. In fact, the Markov Chain may get trapped in less relevant areas and completely miss important parts of the state space.

2.2. Variational Approximations

The basic idea is to approximate the intractable target probability distribution $p(x) \propto w(x)$ with one that is more tractable. This is typically achieved by choosing a distribution from a family of distributions that are computationally easier to work with, and minimizing a measure of divergence (usually, KL divergence). We refer the reader to (Jordan et al., 1999; Wainwright & Jordan, 2008) for more details. In this paper we will use Mean Field, which provides a lower bound to the value of the partition function Z, and Belief Propagation.

2.3. Randomized Hashing

We provide standard definitions and constructions for universal hash functions (cf. Valiant & Vazirani, 1986; Vadhan, 2011; Goldreich, 2011). Randomized hash functions will be used to randomly select (exponentially) large portions of the state space.

Definition 1. A family of functions $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is pairwise independent if the following two conditions hold when H is chosen uniformly at random from $\mathcal{H}$. 1) $\forall x \in \{0,1\}^n$, the random variable $H(x)$ is uniformly distributed in $\{0,1\}^m$. 2) $\forall x_1, x_2 \in \{0,1\}^n$ $x_1 \neq x_2$, the random variables $H(x_1)$ and $H(x_2)$ are independent.

Optimal hash functions (in a statistical sense) can be constructed by considering the family $\mathcal{H}$ of all possible functions from $\{0,1\}^n$ to $\{0,1\}^m$. It is easy to verify that this is a family of *fully* independent functions. If we define a sample set (e.g., from the set of all possible configurations) as $S = H^{-1}(0) \subseteq \{0,1\}^n$, where H is chosen at random from $\mathcal{H}$, the resulting sampled configurations S would be chosen independently (i.i.d. samples). Unfortunately, specifying a function from this family requires $m2^n$ bits (m bits for each possible variable assignment), making this construction not very useful for large n. On the other hand, *pairwise independent* hash functions can be specified compactly. They are generally based on modular arithmetic constraints of the form $Ax = b \mod 2$, referred to as parity or XOR constraints.

Proposition 1. *Let $A \in \{0,1\}^{m\times n}$, $b \in \{0,1\}^m$. The family $\mathcal{H} = \{h_{A,b}(x) : \{0,1\}^n \to \{0,1\}^m\}$ where $h_{A,b}(x) = Ax + b \mod 2$ is a family of pairwise independent hash functions.*

Note that in order to get pairwise independence, we must choose a hash function H uniformly at random from the family $\mathcal{H}$. In particular, this is equivalent to generating A, b by the following process: choose each entry $A_{i,j} \overset{iid}{\sim}$ Bernoulli($\frac{1}{2}$) and $b_i \overset{iid}{\sim}$ Bernoulli($\frac{1}{2}$). The resulting A matrix generated this way is relatively dense since, on average, half of the entries will be ones, and the parity constraints generated this way will involve many variables.

If we are willing to sacrifice pairwise independence for a weaker notion of independence, we can use sparse parity constraints, where each parity constraint involves significantly fewer variables than required for pairwise independence. In practice, these sparse parity constraints are much more easily decoded by optimization solvers. The weaker notion of independence that sparse parity constraints satisfy is known as Average Universal and defined in (Ermon et al., 2014), where the following proposition is proven:

Proposition 2. *Let $A \in \{0,1\}^{m\times n}$ be a random matrix whose entries are Bernoulli i.i.d. random variables of parameter $f \leq 1/2$, i.e., $\mathbb{P}[A_{ij} = 1] = f$. Let $b \in \{0,1\}^m$ be chosen uniformly at random, independently from A. Then the family $\mathcal{H}^f = \{h_{A,b}(x) : \{0,1\}^n \to \{0,1\}^m\}$, where $h_{A,b}(x) = Ax + b \mod 2$ and $H \in \mathcal{H}^f$ is chosen randomly according to this process, is a family of Average Universal hash functions (for some parameters).*

3. Combining Random Projections with Probabilistic Inference Algorithms

3.1. Computing the Partition Function

We now introduce our hybrid approach for probabilistic inference utilizing random projections. The key idea is that instead of directly choosing x uniformly at random from $\mathcal{X}$ and then scaling $w(x)$ by 2^n to use as an unbiased estimator of Z, we will choose an exponentially large subset of configurations $S \subseteq \mathcal{X}$ at random using a universal hash function and then scale our estimate $\tilde{Z}$ for $\sum_{x\in S} w(x)$ appropriately to use as our estimator of $Z = \sum_{x\in\mathcal{X}} w(x)$.

Concretely, given a graphical model G with partition function $Z = \sum_{x\in\mathcal{X}} w(x)$, we will accomplish the random projection by adding m random XOR factor nodes to G, where each XOR factor node represents an XOR constraint on our model. We will then run a probabilistic inference algorithm on each randomly projected graphical model to obtain estimates $\tilde{Z}$ that we will then scale and average appropriately to obtain our estimate of Z.

The specific construction for adding the i^{th} XOR constraint to our graphical model is as follows. Generate a parity bit b uniformly at randomly from $\{0,1\}$. Add each variable node j from G to our XOR constraint with probability $f \in (0, \frac{1}{2})$. Suppose that in the previous step we've added l variable nodes $x_{r_1}, ..., x_{r_l}$ from G. If the parity of these l nodes $x_{r_1} \oplus x_{r_2} \oplus ... \oplus x_{r_l} = b$, then let the potential function over this factor $\psi_i(x_{r_1}, ..., x_{r_l}) = 1$. Otherwise, let $\psi_i(x_{r_1}, ..., x_{r_l}) = 0$. Such a parity constraint implements a universal hash function and can be compactly represented even when l is large (Ermon et al., 2013b;a). This fact is well known in the coding theory and parity check codes literature (MacKay, 1999).

More formally, let $A_{i,j} = 1$ if node j is included in the i^{th} XOR constraint and $A_{i,j} = 0$ otherwise. Let b_i be the parity bit generated for the i^{th} XOR constraint. A configuration $x \in \mathcal{X}$ satisfies the set of XOR constraints if and only if $Ax \equiv b$ (mod 2). After M random trials, we obtain M randomly projected graphical models $G^{(k)}$, each with m XOR constraints determined by the outcomes of the random variables $\left\{\{A_{i,j}^{(k)}\}_{j=1}^n, b_i^{(k)}\right\}_{i=1}^m$ where each $A_{i,j}^{(k)} \overset{iid}{\sim}$ Bernoulli(f) and $b_i^{(k)} \overset{iid}{\sim}$ Bernoulli($\frac{1}{2}$). Such a construction implements the sparse parity constraints described in Section 2.3. Let us denote the partition function of the k^{th} randomly projected graphical model $G^{(k)}$ as $Z^{(k)}$. Incorporating the XOR factor potentials explicitly, we have that $Z^{(k)} = \sum_{x\in\mathcal{X}} w(x) \prod_{i=1}^m \psi_{\{A_i^{(k)}, b_i^{(k)}\}}(x_{\{A_i^{(k)}\}})$.

Notice that, on average, adding one XOR constraint will remove half of the terms in the sum $\sum_{x\in\mathcal{X}} w(x)$ since from Proposition 2, half of the configurations in $\mathcal{X}$ will not satisfy the XOR constraint. We define our set of samples S as

Algorithm 1 RP-InfAlg$(G, \mathcal{A}, M, m, f, p)$

```
for $k = 1, \cdots, M$ do
    Construct $G^{(k)}$ from $G$ by adding $m$ random XOR factor nodes to $G$ as follows:
    for $i = 1, \cdots, m$ do
        Generate parity bit $b_i^{(k)} \overset{iid}{\sim}$ Bernoulli$(\frac{1}{2})$
        For each variable node $j$ from $G$, randomly choose $A_{i,j}^{(k)} \overset{iid}{\sim}$ Bernoulli$(f)$
        Let $\{x_{r_1,...,}x_{r_l}\} = \{x_j | A_{i,j}^{(k)} = 1\}$
        Let the potential over this factor be $\phi_i^{(k)}(x_{r_1}, ..., x_{r_l}) = 1$ if $x_{r_1} \oplus x_{r_2} \oplus ... \oplus x_{r_l} = b_i^{(k)}$ and $\phi_i^{(k)}(x_{r_1}, ..., x_{r_l}) = p$ otherwise
    end for
    $\widetilde{Z}^{(k)} \leftarrow$ Run inference algorithm $\mathcal{A}$ on the graphical model $G^{(k)}$ which is a random projection (perturbation) of $G$
end for
Return $\frac{1}{M} \sum_{k=1}^{M} \left(\frac{2}{p+1}\right)^m \widetilde{Z}^{(k)}$
```

the subset of all configurations satisfying these m randomly generated parity constraints. The number of XOR nodes m we add controls how large the set S is, or in other words what is the size of the resulting state space after the random projection. f determines the average length of each XOR constraint and controls the quality of the sampling procedure. In particular, note that samples in S are *not* chosen independently of each other. If the length of each XOR constraint is sufficiently large, we obtain a strongly universal hash function and the samples are chosen *pairwise independently*.

Let M be the number of trials. We will construct M random projections of G using the construction above and run an inference algorithm on each projected graphical model to get M estimates $\widetilde{Z}^{(k)}$ of $Z^{(k)} = \sum_{x \in \mathcal{X}} w(x) \prod_{i=1}^{m} \psi_{\{A_i^{(k)}, b_i^{(k)}\}}(x_{\{A_i^{(k)}\}})$ that we will then average and scale by 2^m to get our estimate of Z.

Proposition 3. *The estimator* $\frac{1}{M} \sum_{k=1}^{M} 2^m Z^{(k)}$ *is an unbiased estimator of the partition function* Z.

Proof. Let $Z = \sum_{x \in \mathcal{X}} w(x)$. Suppose we add m XOR constraints according to the construction above. Fix any k, and consider the indicator random variable $y(x) = \prod_{i=1}^{m} \psi_{\{A_i^{(k)}, b_i^{(k)}\}}(x_{\{A_i^{(k)}\}})$. Thus, $y(x) = 1$ if the configuration $x \in \mathcal{X}$ is allowable under our set of XOR constraints and 0 otherwise. By definition, $Z^{(k)} = \sum_{x \in \mathcal{X}} w(x) y(x)$, so $\mathbb{E}[Z^{(k)}] = \sum_{x \in \mathcal{X}} w(x) \mathbb{E}[y(x)] = \sum_{x \in \mathcal{X}} w(x) \prod_{i=1}^{m} \mathbb{E}\big[\psi_{\{A_i^{(k)}, b_i^{(k)}\}}(x_{\{A_i^{(k)}\}})\big] = \frac{Z}{2^m}$, and this proves that our estimator is unbiased. □

Since $\mathbb{E}[|S|] = \mathbb{E}\big[\sum_{x \in \mathcal{X}} \prod_{i=1}^{m} \psi_{\{A_i^{(k)}, b_i^{(k)}\}}(x_{\{A_i^{(k)}\}})\big] = \frac{|\mathcal{X}|}{2^m}$, computing an estimate $\widetilde{Z}^{(k)}$ of $\sum_{x \in S} w(x)$ requires summing over a significantly smaller state space than $\mathcal{X}$, and probabilistic inference algorithms which compute an approximation to this sum might compute a much more accurate approximation $\widetilde{Z}^{(k)}$ for the projected graphical model than for the original one. Furthermore, variational techniques are known to be empirically very effective at decoding low-density parity check codes, which are also implemented with XOR constraints (MacKay, 1999; Ermon et al., 2013a). Intuitively, the I-projections used by variational techniques to minimize KL-divergences tend to be overconfident and focus on particular modes of the true posterior distribution (Koller & Friedman, 2009). Randomly projecting the model, on the other hand, can force the I-projection to focus on other parts of the state space.

More generally, we can use the potential function $\phi(x_{r_1}, ..., x_{r_l}) = 1$ if the parity of the l randomly chosen nodes $x_{r_1} \oplus x_{r_2} \oplus ... \oplus x_{r_l} = b$, and $\phi(x_{r_1}, ..., x_{r_l}) = p$ otherwise for $0 \leq p \leq 1$. This generalization introduces a "softer", smoothed perturbation to our graphical model than the 0 and 1 "hard" XOR constraints, where each constraint, on average, removes half of the terms in the sum. Since $\phi(x) = p + (1-p)\psi(x)$, it follows that $\mathbb{E}[\phi(x)] = p + (1-p)/2 = (p+1)/2$, and we can recover an unbiased estimator for Z from $Z^{(k)}$ using the scale factor $\left(\frac{2}{p+1}\right)^m$.

Note that when we start using the "soft" parity constraints, we are not really doing a random projection of the state space but instead a random perturbation (the operator is not idempotent). A possible intuitive explanation for why perturbations with $p > 0$ work well is that they can attenuate the "overconfidence" problem of I-projections by "flattening" the distribution. Similarly, when running Gibbs sampling, the Markov chain may get trapped and miss important parts of the state space. When we average over different perturbations, it's likely that we'll cover a much larger area of the state space and avoid getting trapped in a few areas, leading to more accurate results.

The pseudocode of the algorithm, called RandomlyProjected-InferenceAlgorithm (RP-InfAlg), is reported. It takes as input a graphical model G, a probabilistic inference algorithm $\mathcal{A}$ for estimating the partition function, and some parameters for the random projection (perturbation), and outputs an estimate of the partition function of G. The parameters for the random projection are the number of samples $M \geq 1$, number of XOR constraints $m \geq 1$, sparsity parameter $f \in (0, \frac{1}{2})$, and smoothness parameter $p \in [0, 1]$.

3.2. Bounds on the Partition Function

Proposition 4. *If a probabilistic inference algorithm $\mathcal{A}$ provides a lower (resp. upper) bound for the partition function Z of a graphical model, then for any graphical model G and choice of parameters $M \geq 1$, $m \geq 1$, $f \in (0, \frac{1}{2})$, $p \in [0,1]$,* RP-InfAlg$(G, \mathcal{A}, M, m, f, p)$ *is a lower (resp. upper) bound for Z in expectation, i.e.,* $\mathbb{E}[$RP-InfAlg$(G, \mathcal{A}, M, m, f, p)] \leq Z$.

Proof. Suppose the inference algorithm $\mathcal{A}$ always returns a lower bound $\widetilde{Z}^{(k)}$ for $Z^{(k)}$ satisfying $\widetilde{Z}^{(k)} \leq Z^{(k)}$. Then $\mathbb{E}\big[\frac{1}{M}\sum_{k=1}^{M}\big(\frac{2}{p+1}\big)^m \widetilde{Z}^{(k)}\big] = \frac{1}{M}\sum_{k=1}^{M}\big(\frac{2}{p+1}\big)^m \mathbb{E}[\widetilde{Z}^{(k)}] \leq \frac{1}{M}\sum_{k=1}^{M}\big(\frac{2}{p+1}\big)^m \mathbb{E}[Z^{(k)}] = Z$. □

For the case of lower bounds, we can further obtain a guarantee that some constant multiple of our estimator is a lower bound for Z with high probability using Markov's inequality.

Proposition 5. *If a probabilistic inference algorithm $\mathcal{A}$ provides a lower bound for the partition function Z of a graphical model, then for any graphical model G and choice of parameters $M \geq 1$, $m \geq 1$, $f \in (0, \frac{1}{2})$, $p \in [0,1]$, $1/c$ times the value of* RP-InfAlg$(G, \mathcal{A}, M, m, f, p)$ *is a lower bound for Z with probability at least* $1 - 1/c$.

Proof. By Markov's inequality, we have that $\mathbb{P}\big[\frac{1}{M}\sum_{k=1}^{M}\big(\frac{2}{p+1}\big)^m \widetilde{Z}^{(k)} > cZ\big] \leq \frac{\mathbb{E}\left[\frac{1}{M}\sum_{k=1}^{M}\left(\frac{2}{p+1}\right)^m \widetilde{Z}^{(k)}\right]}{cZ} \leq \frac{1}{c}$. □

For example, setting $c = 100$, $\log_{10}\big(\frac{1}{M}\sum_{k=1}^{M}\big(\frac{2}{p+1}\big)^m \widetilde{Z}^{(k)}\big) - 2$ is a lower bound for $\log_{10} Z$ with probability at least 0.99. In Section 4.2, we empirically show improvements in computing the lower bound of the partition function of Ising grid models using Mean Field, where the lower bounds provably hold with high probability.

3.3. Discussion of Variance

Recall that in the case where we have the exact partition functions, our estimator is unbiased, and the expected value does not depend on our choice of parameters M, m, f, or p in the random projection step. However, the variance of our estimator does depend on these parameters in a crucial way. Clearly, increasing the number of trials M that we use gives us a more accurate answer. Similarly, using a smoother perturbation (larger p) also decreases the variance. The parameters m and f control the trade-off between having a simple model with easy inference but also high variance for our estimator (e.g., when we have m XOR constraints and include only one variable in each XOR constraint, we are clamping m variables to a random value), and having low variance for our estimator but only a slightly simpler model with slightly easier inference than the one we started off with. We can quantify the effects of the parameters on the variance with the following Proposition.

Proposition 6. *For any graphical model G and choice of parameters $M \geq 1$, $m \geq 1$, $f \in (0, \frac{1}{2})$, $p \in [0,1]$, $\frac{1}{M}\sum_{k=1}^{M}\big(\frac{2}{p+1}\big)^m Z^{(k)}$ is an unbiased estimator of the partition function Z. Furthermore, the variance of this estimator is bounded from above by the following expressions:*

$$
\begin{aligned}
&Var\left[\frac{1}{M}\sum_{k=1}^{M}\left(\frac{2}{p+1}\right)^m Z^{(k)}\right] \\
&\leq \frac{1}{M}\left(\frac{4(p+\frac{1}{2}(1-f)(1-p)^2)}{(p+1)^2}\right)^m Z^2 \\
&\qquad + \frac{1}{M}\left(\frac{2(p^2+1)}{(p+1)^2}\right)^m \sum_{x\in\mathcal{X}} w(x)^2 - \frac{Z^2}{M} \\
&\leq \frac{2}{M}\left(\frac{2(p^2+1)}{(p+1)^2}\right)^m Z^2
\end{aligned}
$$

Proof. Refer to Appendix □

In the case of $p = 0$, the variance is bounded from above by $Var\big[\frac{1}{M}\sum_{k=1}^{M} 2^m Z^{(k)}\big] \leq \frac{2}{M} 2^m Z^2$.

If we choose $p = 1/2$, the variance is bounded from above by $Var\big[\frac{1}{M}\sum_{k=1}^{M}\big(\frac{4}{3}\big)^m Z^{(k)}\big] \leq \frac{2}{M}\big(\frac{10}{9}\big)^m Z^2$.

Thus, we see that using the $p = 0$ hard XOR constraints gives a variance bound which grows exponentially with the number of XOR constraints m as 2^m while using the smoothed $p = 1/2$ XOR factor potentials gives a variance bound which only grows as $\big(\frac{10}{9}\big)^m$, a much more slowly growing function.

In the case where we do not have the exact value of the partition function $Z^{(k)}$ but only an approximation $\widetilde{Z}^{(k)}$, the variance of our estimator obviously also depends on the accuracy of the approximate inference algorithm. Intuitively, the variance is not too large in practice when m is not too large because our estimate $\widetilde{Z}^{(k)}$ is scaled by $\big(\frac{2}{p+1}\big)^m$ with $\big(\frac{2}{p+1}\big)^m \ll \widetilde{Z}^{(k)}$.

3.4. Computing Marginal Probabilities

In addition to computing partition functions using random perturbations, we can also compute marginal probabilities using the same framework. Given a graphical model G, consider the problem of computing the marginal probability of a variable x_1. Concretely, we want to estimate $P(x_1) = \sum_{x\in\mathcal{X}} \delta_{x_1}(x) \frac{w(x)}{\sum_{z\in\mathcal{X}} w(z)}$.

To do so, construct M randomly perturbed graphical models $G^{(k)}$ using Algorithm 1 (also described in Section 3.1). Suppose we have $P^{(k)}(x_1)$, which is the marginal for $G^{(k)}$, and $Z^{(k)}$, which is the partition function of $G^{(k)}$. Let our estimator for $P(x_1)$ be $T_M = \frac{\frac{1}{M}\sum_{k=1}^{M} P^{(k)}(x_1)Z^{(k)}}{\frac{1}{M}\sum_{k=1}^{M} Z^{(k)}}$. We will show that as the number of trials M becomes sufficiently large, bias$(T_M) \to 0$ as $M \to \infty$.

Proposition 7. *The estimator $T_M = \frac{\frac{1}{M}\sum_{k=1}^{M} P^{(k)}(x_1)Z^{(k)}}{\frac{1}{M}\sum_{k=1}^{M} Z^{(k)}}$ is an asymptotically unbiased estimator of the marginal probability $P(x_1)$.*

Proof. Note that $P^{(k)}(x_1) = \sum_{x\in\mathcal{X}} \delta_{x_1}(x) \frac{w(x)\prod_{i=1}^{m} \phi_{\{A_i^{(k)},b_i^{(k)}\}}(x_{\{A_i^{(k)}\}})}{\sum_{z\in\mathcal{X}} w(z)\prod_{i=1}^{m} \phi_{\{A_i^{(k)},b_i^{(k)}\}}(z_{\{A_i^{(k)}\}})}$ and $Z^{(k)} = \sum_{x\in\mathcal{X}} w(x)\prod_{i=1}^{m} \phi_{\{A_i^{(k)},b_i^{(k)}\}}(x_{\{A_i^{(k)}\}})$.

Define Y_k to be the k^{th} term in the numerator and Z_k to be the k^{th} term in the denominator, so $T_M = \overline{Y}/\overline{Z}$, a ratio of sample means. Computing the expected value of the numerator and denominator, we get that $\mathbb{E}[Z_k] = \mathbb{E}\left[\sum_{x\in\mathcal{X}} w(x)\prod_{i=1}^{m} \phi_{\{A_i^{(k)},b_i^{(k)}\}}(x_{\{A_i^{(k)}\}})\right] = \left(\frac{p+1}{2}\right)^m \sum_{x\in\mathcal{X}} w(x)$ and similarly $\mathbb{E}[Y_k] = \mathbb{E}\left[\sum_{x\in\mathcal{X}} \delta_{x_1}(x)w(x)\prod_{i=1}^{m} \phi_{\{A_i^{(k)},b_i^{(k)}\}}(x_{\{A_i^{(k)}\}})\right] = \left(\frac{p+1}{2}\right)^m \sum_{x\in\mathcal{X}} \delta_{x_1}(x)w(x)$.

The marginal we want to estimate, $P(x_1)$, is therefore equal to $\frac{\mathbb{E}[\overline{Y}]}{\mathbb{E}[\overline{Z}]}$. Since the M samples $(Y_1, Z_1), ..., (Y_M, Z_M)$ are i.i.d. random 2-vectors, a straightforward calculation shows that the estimator T_M is an asymptotically unbiased estimator of the marginal probability $P(x_1)$ (Shao, 2003).

$\square$

4. Experimental Results

4.1. Experimental Methodology

The probabilistic inference algorithms we will test in this section in conjunction with random projections are Mean Field, Belief Propagation, and Gibbs sampling, as implemented in the LibDAI library (Mooij, 2010).

We choose $p = 1/2$ for our XOR factor potentials to introduce a smoothed perturbation to our graphical model and reduce the variance of our estimator. In addition, instead of adding each variable node to our XOR constraint with probability f, we use a fixed number, l, of variable nodes in each XOR constraint. Constant XOR lengths further reduce the variance of the results and improve the quality of the sampling. Intuitively, using fixed length constraints rules out the generation of trivial constraints of length 0, which can happen if each variable is added to the constraints with fixed probability f.

Experimentally, we find that compared to computing marginal probabilities using the asymptotically unbiased estimator derived in section 3.4, averaging the marginal probabilities across all M trials gives us better results. This is because weighting the marginal probabilities by an estimate $\widetilde{Z}^{(k)}$ of the partition function is unreliable when the estimates $\widetilde{Z}^{(k)}$ can differ by orders of magnitude from the true values $Z^{(k)}$. Concretely, we estimate a marginal probability $P(X_i = x_i)$ as $\frac{1}{M}\sum_{k=1}^{M} \widetilde{P}^{(k)}(X_i = x_i)$. In practice, averaging the marginal probabilities gives us good results when the number of trials M is much larger than the number of variables n in our graphical model, because the distribution of variables that each random projection affects will tend to smooth out when we average over all M trials.

The evaluation criterion that we will use for marginal probabilities is average L1 error. Given the exact marginal for a variable $P_{true}(X_i = x_i)$ and the marginal returned by an approximate inference algorithm $\widetilde{P}(X_i = x_i)$, the error is given by $\frac{1}{d}\sum_{i=1}^{n}\sum_{x_i\in X_i} |\widetilde{P}(X_i = x_i) - P_{true}(X_i = x_i)|$ where $d = \sum_{i=1}^{n}\sum_{x_i\in X_i} 1$ is the total dimensionality. In the case where all variables are binary, then the error reduces to $\frac{1}{n}\sum_{i=1}^{n} |\widetilde{P}(X_i = 1) - P_{true}(X_i = 1)|$.

4.2. Ising grid models

We evaluate our randomly projected inference algorithms on 10 by 10 Ising grid models, with 100 binary variables, for marginal probabilities and on 15 by 15 Ising grid models, with 225 binary variables, for Mean Field partition function. Ising grid models have unary potentials $\psi_i(x_i) = e^{f_i x_i}$ and (mixed) binary interactions $\psi_{ij}(x_i, x_j) = e^{w_{ij} x_i x_j}$ where $w_{ij} \overset{iid}{\sim} \text{Uniform}(-w, w)$ and $f_i \overset{iid}{\sim} \text{Uniform}(-f, f)$. We report our results for $w \in [1, 10]$ with external field $f = 0.1$.

Partition function results for Mean Field are presented in Figure 1. We construct $M = 50$ random projections of G with XOR lengths $l = 1, 2, 4$ and number of XOR constraints $m = 20$. We use 10 random initializations for Mean Field on each randomly projected $G^{(k)}$ and report the maximum value over the 10 restarts for $\widetilde{Z}^{(k)}$. We obtain an estimate $\tilde{Z}$ by averaging over the $\widetilde{Z}^{(k)}$'s for each choice of XOR length $l = 1, 2, 4$ and we choose the maximum over these 3 XOR lengths as our final estimate $\tilde{Z}$, which we then compare to the estimate obtained by running Mean Field on G with $10M = 500$ random restarts. Note that we choose 500 random restarts for Mean Field to ensure a fair comparison between MF and RP-MF in terms of computational budget. For the hardest Ising models, we did observe that having more random restarts improved the lower bound provided by Mean Field.

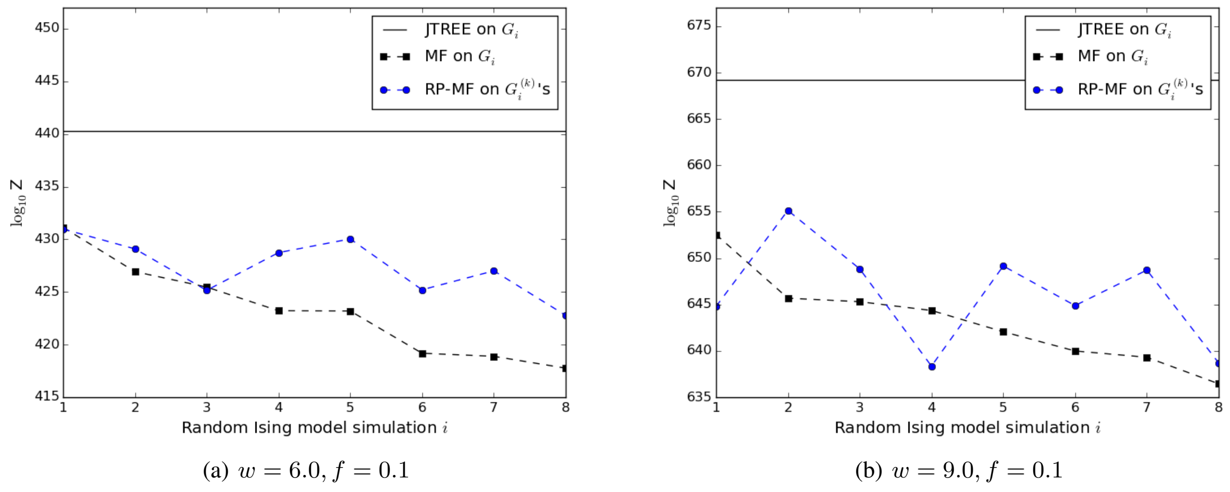


Figure 1. Partition function results for Ising grids using Mean Field, with and without random projections. Note that true Z is denoted by the solid, horizontal line while recovered Z is denoted by the dashed lines.

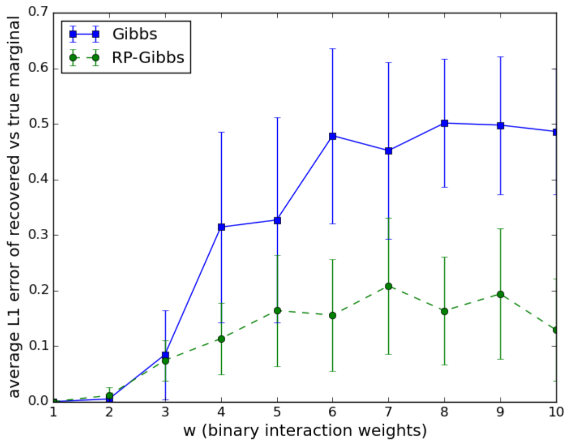
Our results for 8 random Ising models G_i are presented in Figure 1 for selected values of w. For clarity, for each i, we shift all the $\log_{10} Z_i$'s by a constant c_i along the y-axis with c_i determined by $\log_{10} Z_i^{TRUE} + c_i = \frac{1}{8}\sum_{j=1}^{8} \log_{10} Z_j^{TRUE}$. Then, for each i, the vertical distance between the solid horizontal line and the dashed line represents the difference $\log_{10} Z_i^{TRUE} - \log_{10} \widetilde{Z}_i^{MF}$ or $\log_{10} Z_i^{TRUE} - \log_{10} \widetilde{Z}_i^{RP-MF}$. (For clarity, we also sort the 8 random Ising models by decreasing $\log_{10} \widetilde{Z}_i^{MF}$.)

From Figure 1, we see that running Mean Field on G produces a lower bound (the dashed black line) that is many orders of magnitude away from the true value of the partition function (the solid, horizontal black line), as computed by running Junction Tree on G. We see that the lower bound on Z computed by running Mean Field on random projections of G is often orders of magnitude better than the lower bound obtained from directly running Mean Field on G. Furthermore, we know that $\log_{10} \widetilde{Z}_i^{RP-MF} - 2$ is a lower bound for $\log_{10} Z_i^{TRUE}$ with probability at least 0.97 (by the union bound). Thus, we see substantial improvements in the accuracy of computing lower bounds for the partition function. Notice that these results are obtained giving the two techniques essentially the same computational budget. Our approach, however, can be easily run in parallel, as inference on each perturbed graphical model can be carried out independently.

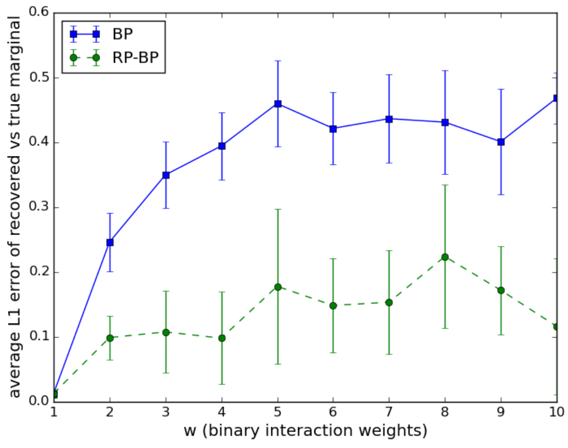
Marginal results with Gibbs sampling and Belief Propagation are presented in Figure 2. We construct M = 1,000 random projections of G with XOR length $l = 4$ and number of XOR constraints m = 20. We run up to 10,000 iterations of Gibbs sampling and Belief Propagation on each $G^{(k)}$ to obtain our estimate of the marginal probabilities $\widetilde{P}^{(k)}$. For fairness, we compare our final estimate $\tilde{P}$ with the estimate of P obtained by running Gibbs sampling or Belief Propagation on G with 10,000M = 10,000,000 iterations. Our results, the L1 error between the Gibbs marginals and the true marginals versus the L1 error between the RP-Gibbs marginals and the true marginals, are presented in Figure 2 across many values of the Ising grid parameter w (the binary interaction weights). For $f = 0.1$, we see substantial improvements in the accuracy of the recovered marginals for almost all values of w. In particular, we note that for large values of w, the marginals recovered by running Belief Propagation and Gibbs sampling on G, even with a large number of iterations, have close to a 0.5 L1 error compared to the true marginals, an error that can be achieved by random guessing. Using random projections, we are able to drastically improve the accuracy of the recovered marginals to an L1 error of around 0.1-0.2.

Note that all of the experiments we run are designed to give a fair comparison in terms of computational budget between an approximate inference algorithm with the use of random projections and without the use of random projections. Specifically, this means that we allow both the InfAlg and the RP-InfAlg (across its many random samples) to run for the same (maximum) number of iterations. Given the addition of new factors, each iteration of the RP-InfAlg might be slightly more expensive than InfAlg, but, in practice, we observe that running times are comparable due to the use of sparse constraints.

When computing the partition function with Mean Field, we observe fast convergence in roughly the same num-



(a) Gibbs marginals, $f = 0.1$



(b) BP marginals, $f = 0.1$

Figure 2. Marginal results for Ising grids using Gibbs sampling and Belief Propagation, with and without random projections

Table 1. Average L1 error of recovered vs. true marginals on real-world datasets using Gibbs sampling, with and without random projections

	Linkage	Promedus
Number of instances	17	28
RP-Gibbs	0.09 ± 0.02	0.21 ± 0.06
Gibbs	0.29 ± 0.02	0.24 ± 0.06

ber of iterations for MF and RP-MF with comparable total running times. When estimating marginals with Gibbs sampling and Belief Propagation, the total number of iterations run on the original and projected models are mostly the same (for Ising grids under most choices of parameters, BP does not converge within the specified maximum number of iterations). As we add more XOR factors, each iteration becomes only slightly more expensive. For example, 10,000 iterations of BP takes around 5.5 seconds, while 10,000 iterations of RP-BP (20 XOR constraints of length 4) takes around 7.0 seconds. Similarly, 10,000 iterations of Gibbs takes around 2.6 seconds, while 10,000 iterations of RP-Gibbs takes around 2.8 seconds. For reference, these running times were obtained from running LibDAI on a cluster of servers with Intel Xeon E5520 and E5620 processors. We emphasize that the main advantage of using random projections is the improved performance, and the algorithm can easily be parallelized.

4.3. Real world data

In addition to Ising grid models, we evaluate the use of random projections for computing marginal probabilities with Gibbs sampling on two real-world datasets from the UAI 2014 Inference Competition (Gogate, 2014). Similar to the setup in Section 4.2, we run Gibbs on the given graphical model G with 10,000,0000 iterations and compare the results with those from running Gibbs on $M = 1{,}000$ random projections of G, each with 10,000 iterations.

We report our results for two selected datasets, Linkage (genetic linkage) and Promedus (medical diagnosis), in Table 1. These two datasets were chosen because they were the most challenging ones for Gibbs sampling in the UAI dataset. The results in Table 1 show that for one dataset, RP-Gibbs produces marginals which are only slightly more accurate than Gibbs alone, while for the other dataset, RP-Gibbs produces marginals which are much more accurate than just Gibbs, a noticeable improvement consistent with the results that we had for Ising grid models.

5. Conclusion

We introduced a new probabilistic inference meta-algorithm based on running existing approximate inference algorithms on stochastic perturbations of a probabilistic model. Specifically, we propose to augment a graphical model with randomly generated parity constraints implementing universal hash functions, and we showed that inference on these perturbed graphical models is often easier and solved more accurately by existing approximate inference algorithms because of the simpler structure of the state space. By performing inference on a small number of these perturbed graphical models, we obtained better approximations for the partition function and marginal probabilities, as demonstrated on a range of synthetic and real world datasets.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful feedback.

References

Andrieu, Christophe, de Freitas, Nando, Doucet, Arnaud, and Jordan, Michael I. An introduction to MCMC for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

Chakraborty, S., Meel, K., and Vardi, M. A scalable and nearly uniform generator of SAT witnesses. In *Proc. of the 25th International Conference on Computer Aided Verification (CAV)*, 2013.

Chakraborty, Supratik, Fremont, Daniel J., Meel, Kuldeep S., Seshia, Sanjit A., and Vardi, Moshe Y. Distribution-aware sampling and weighted model counting for sat. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

Ermon, Stefano, Gomes, Carla P., Sabharwal, Ashish, and Selman, Bart. Optimization with parity constraints: From binary codes to discrete integration. In *Proc. of the 29th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2013a.

Ermon, Stefano, Gomes, Carla P., Sabharwal, Ashish, and Selman, Bart. Taming the curse of dimensionality: Discrete integration by hashing and optimization. In *Proc. of the 30th International Conference on Machine Learning (ICML)*, 2013b.

Ermon, Stefano, Gomes, Carla P., Sabharwal, Ashish, and Selman, Bart. Low-density parity constraints for hashing-based discrete integration. In *Proc. of the 31st International Conference on Machine Learning (ICML)*, pp. 271–279, 2014.

Gane, A., Hazan, T., and Jaakkola, T. Learning with maximum a-posteriori perturbation models. In *Proc. of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2014.

Gogate, Vibhav. UAI 2014 inference competition. `http://www.hlt.utdallas.edu/~vgogate/uai14-competition/index.html`, 2014.

Goldreich, O. Randomized methods in computation. *Lecture Notes*, 2011.

Hazan, T. and Jaakkola, T. On the partition function and random maximum a-posteriori perturbations. In *Proc. of the 29th International Conference on Machine Learning (ICML)*, 2012.

Jerrum, Mark and Sinclair, Alistair. The markov chain monte carlo method: An approach to approximate counting and integration. In *Approximation Algorithms for NP-hard Problems*, pp. 482–520. PWS Publishing, Boston, MA, 1997.

Jordan, Michael I., Ghahramani, Z., Jaakkola, Tommi, and Saul, L.K. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Koller, Daphne and Friedman, Nir. *Probabilistic graphical models: principles and techniques*. MIT Press, 2009.

MacKay, David JC. Good error-correcting codes based on very sparse matrices. *Information Theory, IEEE Transactions on*, 45(2):399–431, 1999.

Maddison, Chris J, Tarlow, Daniel, and Minka, Tom. A* sampling. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Madras, N.N. *Lectures on Monte Carlo Methods*. American Mathematical Society, 2002.

Mooij, J.M. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, 11:2169–2173, 2010.

Papandreou, George and Yuille, Alan L. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 193–200. IEEE, 2011.

Roth, Dan. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1):273–302, 1996.

Shao, Jun. *Mathematical Statistics*. Springer, 2003.

Tarlow, Daniel, Adams, Ryan P, and Zemel, Richard S. Randomized optimum models for structured prediction. *Journal of Machine Learning Research - Workshop and Conference Proceedings*, 22:1221–1229, 2012.

Vadhan, S. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 2011.

Valiant, L.G. and Vazirani, V.V. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

Wainwright, Martin J. and Jordan, Michael I. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.