notes on ermons papers

fnd

September 2017

Ermon 2013 approximates log partition functions defined over a discrete, exponentially large set of configurations using universal hash functions. Hsu 2015 builds on this work by approximating log marginal densities with a provably tight variational approximation by computing an I-projection of the random projection. These two steps are summarized nicely in figure 1 from Hsu 2015. We provide intuition for why the random projection preserves important properties of p (and can subsequently approximate p with high accuracy) and why the I-projection of the random projection provides a variational approximation with tighter bounds compared to an I-projection of p.

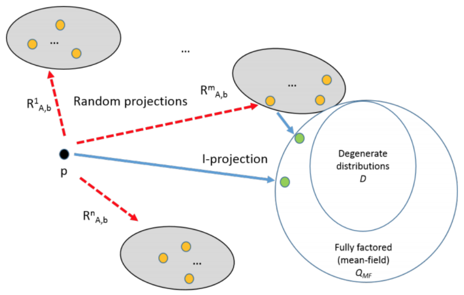
$R^1_{A,b}$
Random projections
$R^m_{A,b}$
p
I-projection
$R^n_{A,b}$
Degenerate distributions D
Fully factored (mean-field) Q_{MF}

Figure 1: Pictorial representation of the approach. p is the target intractable distribution. Random projections implemented by universal hashing alter the partition function of the model in a predictable way. Random projections reduce the degrees of freedom of the model so that it becomes easier to approximate using tractable distributions.

1 Step 1: Ermon 2013

Ermon 2013 proposes a randomized algorithm that gives a constant factor approximation of high dimensional log partition functions. Formally, we want to approximate :

$$Z = \sum_{x \in X} \prod_{\alpha \in I} \psi_\alpha(\{x\}_\alpha) = \sum_{x \in X} w(x)$$

X is the set of all possible variable assignments, ψ is the set of factors, and w is a weight function that assigns each configuration a weight that is proportional to its probability. Therefore, we can interpret our task to be estimating all weights w(x).

Lets order the set of configurations X in increasing order by weight (see left figure below).

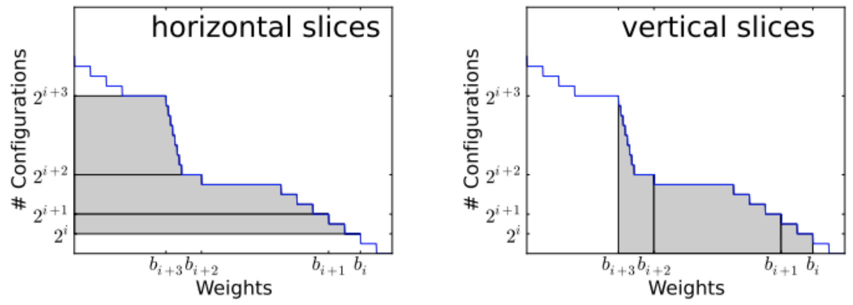

Figure 2. **Horizontal vs. vertical slices for integration.**

The intuition involves using an optimization oracle to estimate the most likely configuration (and the least likely), the 2^ith most likely configuration, the 2^{i+1}th most likely,....,and finally the 2^{n-1} most likely configuration. You can think of these n estimates as the critical points of the distribution where the set of configurations between any two points must have a weight that is bounded by the weights at the two points. Naturally, this will give us upper and lower bounds, where the upper bound assumes all configurations between two critical points have probaiblity mass equal to the point with higher mass and vice versa for the lower bound. *This procedure can get us to an estimate within a factor of 2 of the function we want to estimate.*

The question remains what is our optimization oracle? The key insight here is that we can use a randomized hash function (interpreted as a set of parity constraints) to compute a weighted average of the mode of increasingly sparser distributions (size of state space is a function of 2^{-i} where i is number of parity constraints...think of each parity constraint as reducing the degrees of freedom by 1). Computing the mode of any discrete distribution is a combinatorial optimization problem (which is "only" NP-hard) as opposed to computing the sum of all possible configurations which is #P-hard. For each optimization instance, use your favorite SAT/SMT/integer program solver, which nowadays can handle computing a satisfying solution from a million variables and 5 million constraints.

Lets now discuss the algorithm.

- Compute the mode of the function
- Introduce a single XOR/parity constraint involving roughly half of the

variables (with probability .5, we decide if a variable should be included in the constraint and then with probability .5 we decide if the sign of the constraint should be 1 or 0). If a configuration does not satisfy the parity constraint, set the weight to 0. This effectively reduces the state space by 2. (see second graph in below figure). Compute the mode of this distribution. Repeat with T parity constraints and take the average of the modes.

- Introduce a second parity constraint. Configurations must satisfy both constraints. Our state space is a fourth of the size of the original. Repeat T times and average modes.
- ...
- Introduce the nth constraint. Only one (or zero) configurations will satisfy this constraint.

We then estimate the function by taking a weighted average of the (averaged) modes from each set of parity constraints, where each mode is weighted by 2^{i-1}.

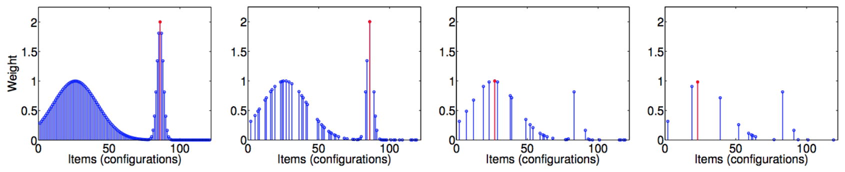

Figure 1. Visualization of the "thinning" effect of random parity constraints, after adding 0, 1, 2, and 3 parity constraints. Leftmost plot shows the original function to integrate. Constrained optimal solution in red.

In summary, we approximate the #P hard problem of computing the log partition function by instead computing the mode of a series of functions with an increasing number of parity constraints (NP hard but solvable). Each set of parity constraints increasingly restricts the subset of satisfiable configurations, which was used to weight the computed modes accordingly. The computational complexity was reduced from O(2^N) to $O(NlgN)$.

2 Step 2: Hsu 2015

Hsu 2015 estimates a variational lower bound to log marginal densities p(x) by computing the I-projection of the random projection of p(x). The advantage of this approach is that we can get bounds on our approximation. Integrating the random projection with an I-projection allows Hsu to ditch the combinatorial solvers for continuous non-convex optimization subject to parity constraints (e.g. non convex constraints) (e.g. standard mean-field coordinate ascent subject to non-convex constraints).

The random projection algorithm described in Ermon 2013 can be interpreted as either a random projection to a subspace of m << n or simply constraining the set of satisfying configurations in high dimensional p with m parity constraints (I find the latter interpretation *much* easier to understand, but the former is important for this paper, just know they're equivalent representations).

Given a random projection R of p(x), R is by definition simpler then p(x), has less degrees of freedom, and therefore has lower entropy (which is why it's closer to Q in the first figure compared to p). The trick here is to include the set of degenerate distributions D (defined as the set of distributions that place all of its probability mass on a single configuration) in our tractable family Q.

Why is including D in Q important? Because finding $q^* \in Q$ in the worst case is "equivalent to solving a Most Probable Explanation query which is NP-equivalent in the worst case", which is effectively the same task described in Ermon 2013! The big difference is instead of computing the mode of p(x) subject to parity constraints using a combinatorial optimization solver, we do traditional mean-field inference subject to non-convex constraints (the problem ends up being coordinate wise convex) to find (in the worst case) the degenerate distribution that minimizes the KL between q^* and R. Finding the "degenerate distribution" q^* is a similar task to finding the mode of p(x) subject to parity constraints.

We know the worst case because D is a subset of Q, so therefore the approximation can only improve if our optimization subroutine finds a more complex distribution that is not degenerate. Lastly, the algorithm is as follows: for random projection R_t, we compute the KL between q and R_t, for all q $\in Q$ using coordinate wise ascent subject to non-convex constraints. Repeat T times and compute an average across projections multiplied by $2^m k$ to get an estimate of p(x).