Approximate inference on the Ising grid

Ilyes KHEMAKHEM - Remi LE PRIOL - Brahim Khalil ABID

January 2017

1 Introduction

The Ising model is a very important model from statistical physics used to represent multi-particle systems with interactions. It is also a classical example of a probabilistic graphical model. We consider a 2D grid $G = (V, E)$, and we suppose that each random variable X_s attached to node s is Bernoulli, taking values in $-1, +1$. Those values may represent the spin of particles in the context of physics, or the colour of a pixel in image processing. Random variables X_s and X_t are allowed to interact directly only if they are joined by an edge, and the energy of their interactions is $-\frac{1}{2}a_{s,t}x_s x_t$. Thus, the probability for a certain configuration X is proportional to $exp(-E(x))$ where:

$$E(x) = -\frac{1}{2}\sum_s b_s x_s - \frac{1}{2}\sum_{s,t} a_{s,t}x_s x_t$$

$$\mathrm{P}(X) \propto exp(-E(x))$$

$a_{s,t}$ encodes the correlation between adjacent nodes. A large $a_{s,t}$ means that X_s and X_t will tend to have the same sign. b_s encodes the potential at each node. $b_s > 0$ means that X_s will tend to be equal to $+1$, whereas $b_s < 0$ means that X_s will tend to be equal to -1. In the following σ will stand for the logistic function: $\sigma(z) = \frac{1}{1+e^{-z}}$.
We consider the problem where the parameters $\theta = (a, b)$ are known. We would like to sample according to this distribution, infer the marginal probabilities, or equivalently the mean parameters $\mu = \mathbb{E}[X]$. In order to do so, we present three different algorithms in the next section, which we implemented in the particular case of an Ising grid, and compared to each other. Our work is based on [1] and [2].

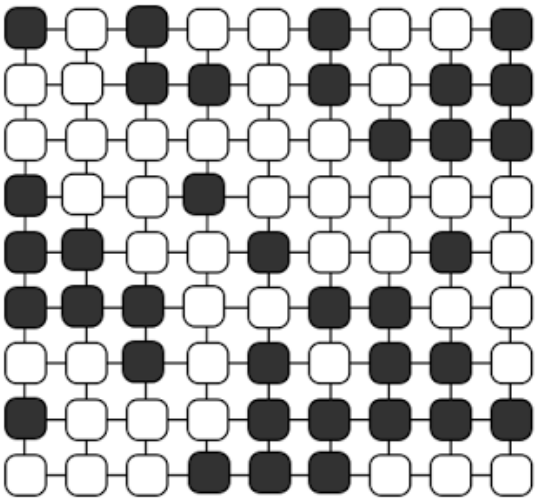

Figure 1: Example of an Ising grid

2 Algorithms

The problem of inference in graphs is not easy in the general case. We have an exact algorithm, the junction tree algorithm, that works on all triangulated undirected graphs, and consequently also on directed graphs via the procedures of moralization and triangulation. However, these procedures are quite heavy, and for complex graphs the computation costs grow exponentially with the size of the maximal clique in the graph. In our work we discuss algorithms for approximate inference that would be computationally more efficient. We will present the Gibbs algorithm for sampling, and the mean field and loopy belief algorithms for inference.

2.1 Gibbs sampling

Starting from an initial observation, Gibbs sampling allows us to sample from the initial probability $p(X)$ by considering the Markov property so that $\mathbb{P}(X_i = .|X_{-i} = x_{-i}) = \mathbb{P}(X_i = .|X_{N_i} = x_{N_i})$, where N_i is the Markov blanket of node i. Based on this, Gibbs sampling is an iterative algorithm that converges to the distribution P.

In the case of an Ising grid, the probability P takes a simple form:

$$\mathbb{P}(X_i = 1 \mid X_{N_i} = x_{N_i}) = \sigma(b_i + \sum_{j \in N_i} a_{ij} x_j)$$

Gibbs sampling updates can be done in multiple way. An efficient way is to sample an entire block of nodes at each iteration. In our example, we consider a checkerboard pattern, and at each iteration update half the grid.

Algorithm 1: Checkerboard Gibbs sampling

initialization: $t = 0$ and x^0
while $t < T$ **do**
  if $t \ mod \ 2 == 0$ **then**
    B = {first half of the checkerboard};
  else
    B = {second half of the checkerboard};
  end
  $\forall$ node $i \in B, x_i^t \sim Bernoulli(\sigma(b_i + \sum_{j \in N_i} a_{ij} x_j^{t-1}))$;
  $x_{-B}^t = x_{-B}^{t-1}$;
  $t = t + 1$;
end
return : x^T

Although Gibbs sampling is used to sample observations from a graphical model, it can also be used as an inference algorithm. To compute the mean parameter $\mu = \mathbb{E}[X]$, we just execute the algorithm on the grid for multiple trajectories, and use the empirical mean to approximate μ.

2.2 Mean field algorithm

The mean field algorithm comes from field theory. The main idea behind it is that each particle feels in average the mean of the interactions with its neighbours. The validity of this approximation comes from the Law of Large Numbers, so this approximation is more efficient on dense graphs.

Unlike the Gibbs algorithm which is for sampling, the mean field algorithm is used to infer the mean parameters. So instead of sampling iteratively a group X_B of nodes using the values of their Markov Blanket, we instead try to compute directly the mean parameters of the model. To approximate these, we compute the product model (empty graph) best approximating our model with regard to the Kullback-Leibler divergence. By doing this we get the update formula.

Algorithm 2: Mean field algorithm

initialization: $t = 0$ and m^0
while $t < T$ **do**
  $\mu_i^{t+1} = 2 * \sigma(b_i + \sum_{j \in N(i)} a_{i,j}\mu_j^t) - 1$
end
return : μ^T

Given the update formula obtained, the mean field problem is non convex. The fixed point depends on the initialization and consequently we could obtain very different results from different initialization methods.

2.3 Loopy belief algorithm

The sum-product message passing (or belief propagation) algorithm is a well known algorithm for exact inference on trees. It relies on the concept of messages, real valued functions passed through the edges of the graph. However, for graphs with cycles, there is no theoretical proof that this algorithm works, but we can still write it from an algorithmic point of view. For obvious reasons, it is then called loopy belief propagation.

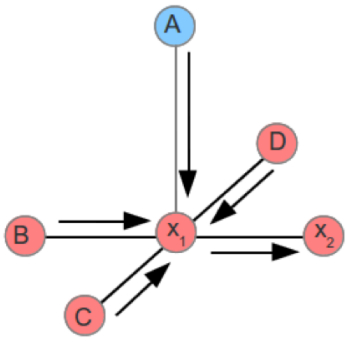

Figure 2: Message passing from x_1 to x_2 with potential A and neighbors B, C and D

The loopy belief algorithm consists in computing iteratively the messages until convergence, and then use the final values to compute the mean parameters. The updating formula is the following:

$$m_{j \to i}(x_i) = \sum_{x_j} \psi_j(x_j)\psi_{j,i}(x_i, x_j) \prod_{k \in N(j) \setminus \{i\}} m_{k \to j}(x_j) \tag{1}$$

Since we have no theoretical guarantees that this algorithm converges, we had to use some tricks to make it work in the loopy case. For instance, we do not update the messages with the formula 1, but we add a damping coefficient to partly keep the old messages. We also have to normalize the messages after each update to keep the messages bounded in value. Moreover, the update order can be done in more than one way: we can either update the four direction in order or choose randomly a direction to update it (while making sure we are updating all the direction equally in frequency).

3 Experiments

We tested the 3 algorithms for different parameters and observations.

For this section, if no indx is provided for parameters a and b, it means that these parameters are constant over the grid.
The color code for the figures is as follows:

- $+1$: white or green.

Algorithm 3: Loopy belief algorithm with damping

initialization: $t = 0$ and $m^0_{i\to j}$ (messages)

while $t < T$ *and for each direction (up, down, left, right)* **do**

Update messages in the given direction, with a damping coefficient d:

$$m^{(t+1)_0}_{j\to i}(x_i) = d * (\sum_{x_j} \psi_j(x_j)\psi_{j,i}(x_i, x_j) \prod_{k\in N(j)\setminus\{i\}} m^{(t+1)_0}_{k\to j}(x_j)) + (1-d) * m^t_{j\to i}(x_i) \quad (2)$$

Normalize the messages in the given direction:
$m^{t+1}_{i\to j}(x_j) = m^{(t+1)_0}_{i\to j}(x_j)/(m^{(t+1)_0}_{i\to j}(1) + m^{(t+1)_0}_{i\to j}(-1))$

end

return : $m^T_{i\to j}$

- -1: black or dark blue.
- 0: grey or turquoise.

3.1 Gibbs sampling

To test Gibbs sampling, we fix the potentials to 0 at each node ($b_i = 0 \forall i \in V$), and varied the initial percentage of $+1$ spins. We then plotted the final state of the grid for different values of the correlations $a = -1, 0, 0.7, 0.8, 0.9, 1, 2$ to observe the phase transition. Figure 3 shows the result.

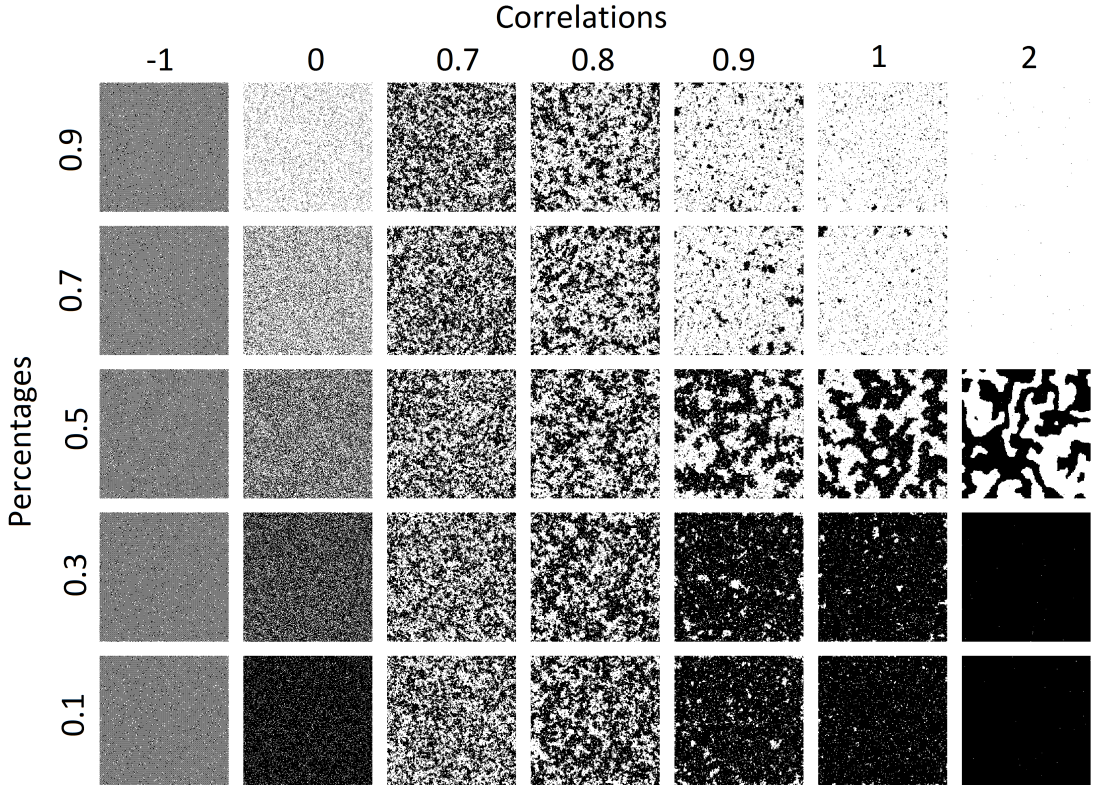

Figure 3: Final state of the grid after Gibbs sampling for different correlations and initial percentage of $+1$s

The phase transition happens for a certain correlation between 0.8 and 0.9. Spins tend to align in a neighbourhood, and we observe a clear separation between $+1$ spins and -1 spins. For an initial population with high $+1$ percentage, all the grid aligns to $+1$ if the correlations between nodes is strong enough (equal to 2 in our tests).

Interestingly though, for negative correlations, spins are completely random and we just observe noise, independently from the initial percentage of +1.

3.2 Mean field algorithm

Mean field is an iterative algorithm. Starting from a initial grid of mean parameters, the algorithm converges toward a state that minimizes the Kullback-Leibler divergence with respect to the true distribution P. Interestingly, the mean field problem is non convex. The minimum can thus be a local minimum, in which the algorithm is stuck. Different initialization can yield different fixed points, as illustrated in figure 4 with $a = 10$ and the potentials b as plotted. The starting point 0 tends to give the expected result.

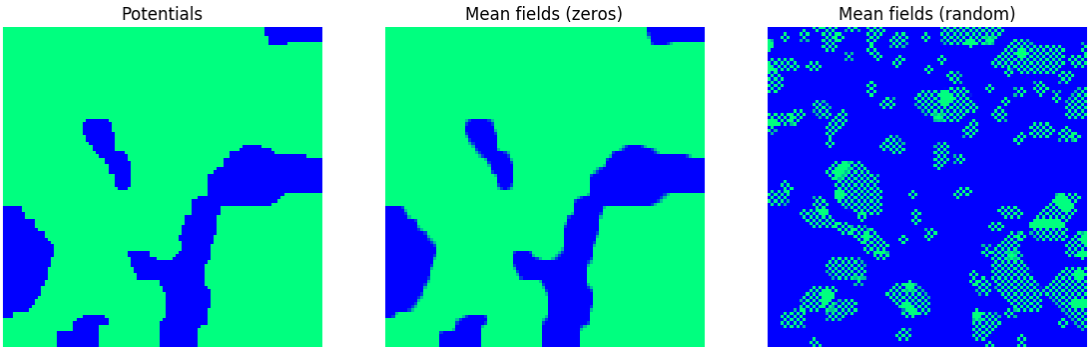

Figure 4: Convergence of meanfield for different initializations. A random initialization converges to a local minimum. Starting from the grid mean value 0 (nodes take values {-1, +1} usually converges to the global minimum.

3.3 Loopy belief algorithm

As we said earlier, loopy belief algorithm has no theoretical guarantee to work for graphs with cycles. Those very cycles, if not dealt with carefully, will just drive the messages to quickly diverge. Computational tricks should be used in the case of graphs with cycles, among them the grid. Before adding the damping to loopy belief, we observed a clear dependence of the result on the update order, as seen in figure 5.

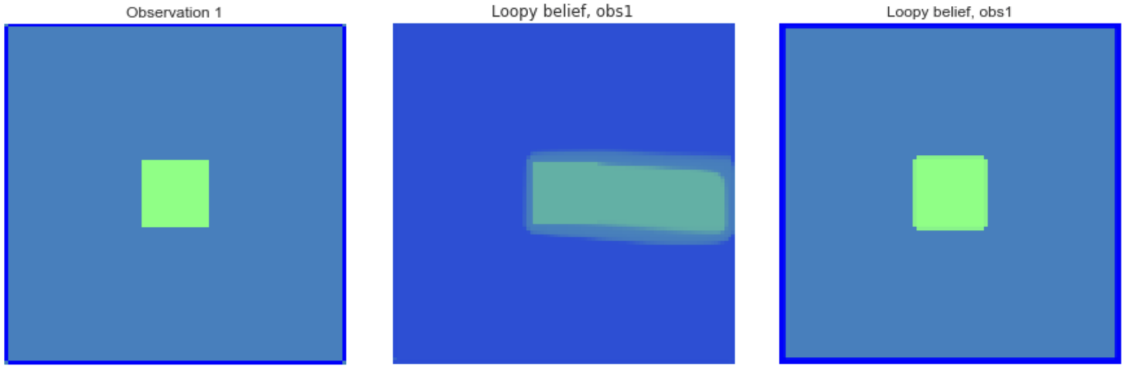

Figure 5: Loopy belief convergence for different update orders. The leftmost figure is the initial observation we have of the grid. Green is for +1, dark blue for -1, and turquoise for unobserved nodes (0). The figure in the center is obtained for a cyclic message update order, whereas the leftmost is obtained for a random message update order.

3.4 Comparison of the 3 algorithms

Practically, we want to start from observed nodes and infer the rest of the distribution. This is what we implemented for mean field, loopy belief propagation and Gibbs sampling for 3 different observations. Figure 6 shows the different results obtained from the 3 algorithms as well as the 3 observations. In every case, $a = 3$ and $b = 0$.
For each algorithm, the initial grid is the observation.
For loopy belief, the damping factor is 0.5, and the update order is cyclic.
For Gibbs sampling, we sample the grid 100 times for each observation, and then compute the mean grid to approximate to mean parameters.

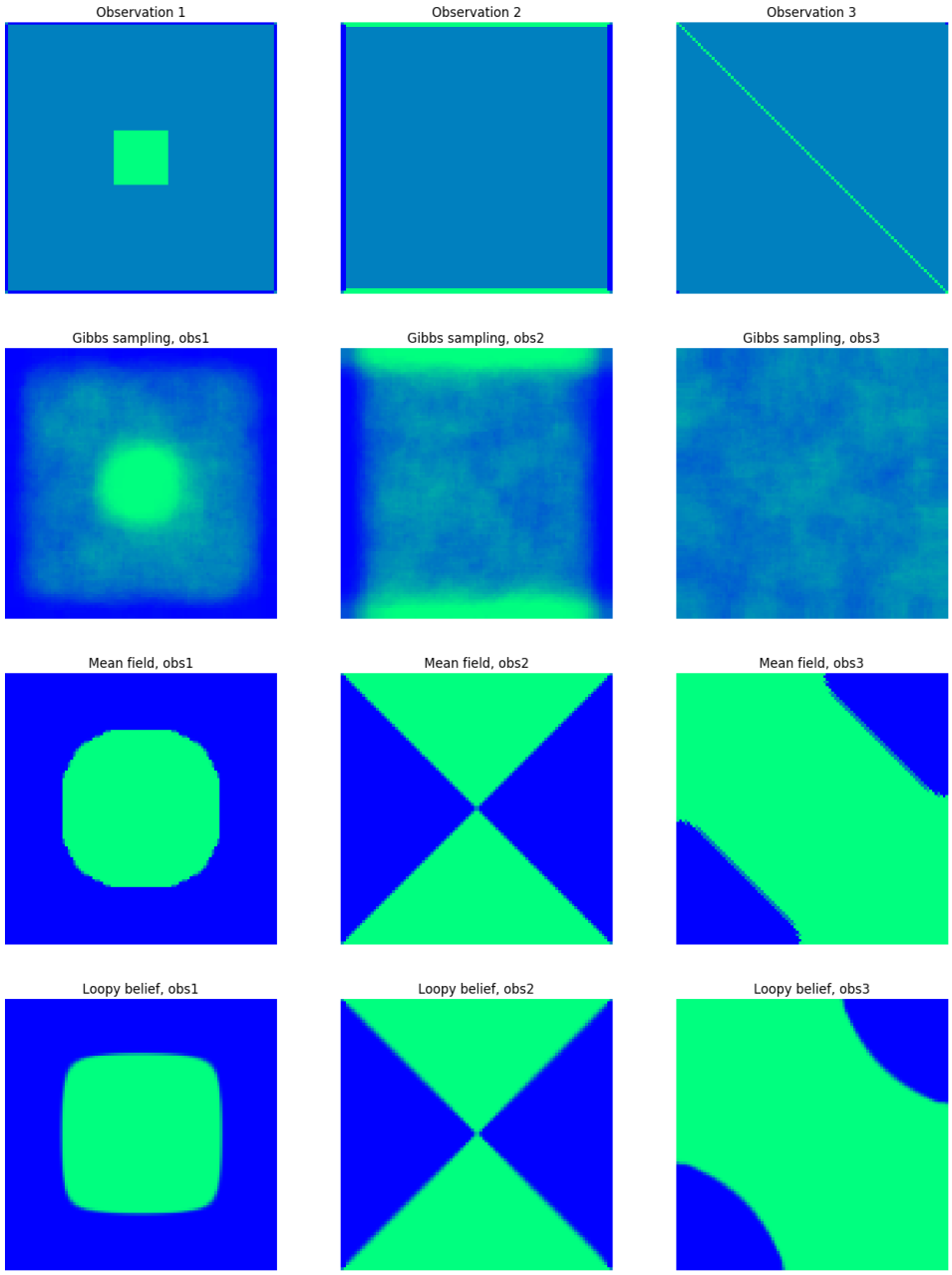

Figure 6: Mean parameters as computed by the 3 different algorithms for 3 different observations. The first row is for the observations. The second row is for the results of Gibbs sampling. The third row is for the results of mean field. The fourth row is for the results of loopy belief. All the algorithms were run on the same model of size 100×100, and with parameters $a = 3$ and $b = 0$.

We also compared the convergence speed of the 3 algorithms for the same parameters. Figure

7 shows the energy evolution as a function of the iteration. Energies have been normalized first to fit the interval $[0, 1]$ in order to allow for a valid comparison.

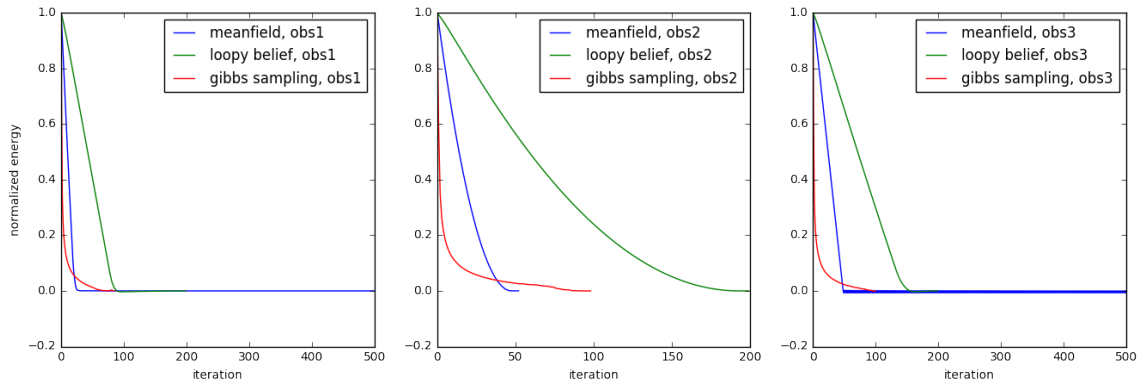

Figure 7: Comparison of convergence speed for the 3 observations

It is very important not to forget that Gibbs sampling needs to be run multiple times (100 in our tests) in order to compute the empirical means, so one should multiply the convergence time of Gibbs sampling by 100 for the real computation time. On the other hand, one loopy belief iteration takes slightly more time than an iteration of Gibbs sampling or mean field.

4 Conclusion

Approximate inference algorithms are used in practice for a variety of applications, even on trees or chains, for their efficiency and lower computation cost than exact inference algorithms. We've explored and tested 3 of these algorithms, each having its pros and cons. Gibbs sampling remains a sampling algorithm on the original problem, and thus the estimated mean parameters are estimated using the empirical mean of several observations. The mean field algorithm directly infers the mean parameters, but on a simpler graphical models (without the edges). This allows for a much lower computation time, but lower accuracy. Finally, Loopy belief algorithm estimates the mean parameters on the original graphical model, but is slower and has no theoretical convergence guarantee on graphs with cycles, in addition to computing an approximate and not the true mean parameters.

References

[1] T. S. Jaakkola M. I. Jordan, Z. Ghahramani and L. K. Saul. An introduction to variational methods for graphical models. *In M. I. Jordan (Ed.), Learning in Graphical Models*, 1999.

[2] Martin J. Wainwright and Michael I. Jordan. *Graphical Models, Exponential Families, and Variational Inference*. now, 2008.