Advanced Machine Learning - Boltzmann Machines

Benedetta Felici, Daria Mihaila, Jasper Pieterse

March 31, 2024

1 Introduction

In this report, we will compare different learning implementations for Boltzmann Machines (BMs). Specifically, we will investigate exact computation, Metropolis-Hastings (MH) sampling, and Mean-Field (MF) solutions. We will begin with inferring parameters for a small-scale Ising model using each method. Next, we'll train a BM on the Salamander retina dataset to highlight the strengths and limitations of each approach.

Our research question is as follows: *What are the key advantages and disadvantages of exact computation, MH sampling, and MF equations within the context of BM learning?*

2 Theory

The Ising Model

A general 2D Ising model consists of a 2D lattice of N binary spins $s = \{s_1, \ldots, s_N\}$ with $s_i = \pm 1$. The energy scalar associated with a state is given by the Hamiltonian equation:

$$E(s) = -\sum_{ij} w_{ij} s_i s_j - \sum_i h_i s_i \tag{1}$$

where w_{ij} are the weights encoding the interaction strengths between the spins and h_i are the local field interactions of the spins. The probability of a state s at a time t is given by the Boltzmann distribution:

$$p(s) = \frac{e^{-\beta E(s)}}{Z} \tag{2}$$

Here, $\beta = 1/T$ is the inverse temperature and Z is the partition function given by summation over all probabilities: $Z = \sum_{k=1}^{2^N} p(s^k)$ where k denotes different states.

Boltzmann machines

The Boltzmann Machine (BM) is a statistical machine learning model based on the Boltzmann distribution. It represents data as a thermal state of an Ising model Hamiltonian. A BM can be used to infer the weights of the general Ising Hamiltonian (1) that best describe an empirical data distribution $q(s)$. The learning is done by maximizing the log-likelihood between the target distribution $q(s)$ and the model distribution $p(s)$ using gradient descent. The log-likelihood function is given by:

$$L(w,h) = \frac{1}{P} \sum_{\mu} \log p(s^{\mu}|w,h) \tag{3}$$

with P training patterns $s = s^1, \ldots s^P$. Substituting the target Boltzmann probabilities (2) in this equation, we can rewrite the equation in the form:

$$L(w,h) = \frac{1}{P} \sum_{\mu} -\beta * E(s^{\mu}|w,h) - \log Z \tag{4}$$

with $E(s^\mu|w,h)$ being given by (1). The learning rules can be obtained by maximizing this equation with respect to the parameters (w,h):

$$w_{ij}^t = w_{ij}^{t-1} + \eta \frac{\delta L}{\delta w_{ij}} = w_{ij}^{t-1} + \eta(\langle s_i s_j \rangle_q - \langle s_i s_j \rangle_p) \tag{5}$$

and

$$h_i^t = h_i^{t-1} + \eta \frac{\delta L}{\delta h_i} = h_i^{t-1} + \eta(\langle s_i \rangle_q - \langle s_i \rangle_p \tag{6}$$

We refer to $\langle s_i \rangle_q$ and $\langle s_i s_j \rangle_q$ as the clamped statistics and $\langle s_i \rangle_p$ and $\langle s_i s_j \rangle_p$ as the free statistics. The clamped statistics can be computed from the data. These update rules ensure that the observed data is the most probable under the Boltzmann distribution (2) with the learned parameters (w,h). Learning is terminated once the first and second-order statistics of the target and model distribution have converged within a certain threshold:

$$\langle s_i s_j \rangle_\rho = \langle s_i s_j \rangle_\eta \qquad \langle s_i \rangle_\rho = \langle s_i \rangle_\eta \tag{7}$$

3 Methods

3.1 Boltzmann Machine Implementation Details

First we compute the clamped statistics $\langle s_i \rangle_q$ and $\langle s_i s_j \rangle_q$ from our training data. We then initialize the Boltzmann machine with a set of parameters (w^*, h^*) drawn from a normal distribution with a specified mean μ_w/N and μ_h/N and a variance $\sigma/\sqrt{N}$. The factors N and $\sqrt{N}$ ensure that the BM scales properly with increased neurons. The self-interactions are set to zero $w_{ii} = 0$ in order to satisfy detailed balance.

At each iteration, we compute the free statistics $\langle s_i \rangle_p$ and $\langle s_i s_j \rangle_p$ using a different solver (Exact, MH or MF) and update the parameters according to the update rules (5,6). This loop continues until learning terminates by either reaching the maximum number of iterations or the statistics converging within a certain threshold.

Evaluation Metrics

For the simulations on the Ising model, we have the oracle weight parameters (w^O, h^O) that generated our statistics (these were generated in the same manner as the initial parameters, but with different μ_w and μ_h and a variance σ). So for these simulations, our convergence criterion is the absolute distance between our inferred parameters and our oracle parameters

$$\Delta w = |w^O - w| \quad \text{and} \quad \Delta h = |h^O - h|$$

The advantage of this is that we can visualize the exact error the BM infers. For the simulations on the Salamander retina data, there are no oracle parameters and thus we opt for plotting the difference in parameters between this epoch and the previous epoch:

$$\Delta w = |w^{\text{previous}} - w| \quad \text{and} \quad \Delta h = |h^{\text{previous}} - h|$$

For exact computation, the algorithm has converged when there are no fluctuations in parameters anymore ($\Delta w = \Delta h = 0$). For MH sampling, the inherent noise in the MH simulation is proportional to $\sigma = \frac{1}{\sqrt{P}}$, with P the number of MCMC samples. So when the algorithm has converged, there will still be fluctuations. We settled on the mean over 50 iterations not changing more than ϵ for a convergence criterion.

For both simulations we also plot the log-likelihood according to equation (4). For $N = 10$ neurons, it is possible to compute Z exactly. For $N = 160$, exact computation of Z becomes intractable, due to the computational complexity of the partition function.

Therefore, we estimate Z from the entropy using the result in MK 31.1:

$$\log(Z) = S - \bar{E} \tag{8}$$

where we can compute S in the mean field approximation using:

$$S = \frac{1}{2}\sum_{i=1}^{N}(1+m_i)\log(1+m_i) + (1-m_i)\log(1-m_i) \tag{9}$$

3.2 Solvers

Exact Computation

In order to obtain clamped statistics from some oracle parameter set (J, h) or to compute free statistics for a single iteration of the BM when using exact computation, we have to take the expectation values over all 2^N states:

$$\langle s_i \rangle = \frac{1}{2^N}\sum_{k=1}^{2^N} p(s^k)s_i^k$$

and

$$\langle s_i s_j \rangle = \sum_{k=1}^{2^N} p(s^k)s_i^k s_j^k$$

where the superscript k denotes a particular spin state and subscript i denotes components of a spin state. Here, the energy is computed using equation (1) and the probability $p(s^k)$ using equation (2).

Metropolis-Hasting Sampling

An alternative way to estimate the clamped statistics is to obtain the samples with a Metropolis-Hasting scheme. Our MH sampler uses no burn-in time. For efficiency reasons, we computed acceptance rates using the *local field* energies:

$$\alpha(\mathbf{s} \to \mathbf{s}') = \min\left(1, \exp\left(-2\beta\Delta E_{\text{loc}}\right)\right) \tag{10}$$

where $\mathbf{s}'$ is a new proposed state with a single random spin s_a randomly flipped and the change in the local field energy is given as:

$$\Delta E_{\text{loc}} = s'_a \sum_j W_{aj}s'_j + s'_a * h_a$$

Mean Field Approximation

The mean field (MF) approximation can also be used to effectively estimate the free statistics of the model. It yields the following expression for the mean spin values:

$$\langle s_i \rangle_= \sum_{j \neq i} tanh(w_{ij}m_j + \theta_i) \tag{11}$$

To approximate the correlations, we rely on the following expression, which follows from the linear response theorem [1]. Indeed, it was shown that relying on a naive mean field approximation to estimate the correlations - $\langle s_i s_j \rangle = \langle s_i \rangle \langle s_j \rangle$ - leads to a non-convergent learning process [2].

$$\begin{aligned}\langle s_i s_j \rangle &= \chi_{ij} + \langle s_i \rangle \langle s_j \rangle \\ \chi_{ij} &= A_{ij}^{-1} \\ A_{ij} &= \frac{\delta_{ij}}{1 - m_i^2} - w_{ij}\end{aligned} \tag{12}$$

In order to solve Equation 11 we use the fixed point iteration method, which is described in Algorithm 1. The underlying idea behind the method is that by starting from a random m we obtain a progressively better approximation of the solution by iteratively applying Equation 11 to m.

In the absence of hidden units in the Boltzmann machine, we can rely on the fixed point equations to solve the Ising model, as shown by Kappen et al [2]. This is particularly useful as the solution is computed in one shot, without performing gradient based learning. In this case, the statistics of the model can be computed directly from the data. The method works as follows: as in absence of hidden units the gradients in Equation 6 are 0, it follows:

$$\langle s_i \rangle_q = \langle s_i \rangle_p \tag{13}$$

Therefore, if we define

$$\begin{aligned}C_{ij} &= \langle s_i s_j \rangle_p - \langle s_i \rangle_p \langle s_j \rangle_p \\ \chi &= \langle s_i s_j \rangle_q - \langle s_i \rangle_q \langle s_j \rangle_q\end{aligned}$$

From $\frac{\partial L}{\partial w_{ij}} = 0$, it follows that:

$$\chi_{ij} = C_{ij}$$

By applying the result to Equation 12, we thus obtain:

$$w_{ij} = \delta_{ij} \left(1 - m_i^2\right) - \left(C^{-1}\right)_{ij}$$

Algorithm 1 Mean Field Approximation

```
1: Initialize a random vector $m$
2: $dm = 1$
3: $\epsilon = 1 \times 10^{-8}$
4: while $dm < \epsilon$ do
5:     $m_{\text{old}} = m$
6:     Compute $m = \tanh\left(\sum_{j \in N(i)} w_{ij} m_j + \theta_i\right)$
7:     $dm = \max(|m - m_{\text{old}}|)$
8: end while
9: The mean field approximation of the mean spin values is $m$
```

3.3 Data Preprocessing

Toy Ising Model

For the Toy Ising model, we generated clamped statistics $\langle s_i \rangle_q$ and $\langle s_i s_j \rangle_q$ by simulating an Ising lattice using exact computation.

Salamander Retina Dataset

For the Salamander data, we generated clamped statistics $\langle s_i \rangle_q$ and $\langle s_i s_j \rangle_q$ as follows: $\langle s_i \rangle$ is given by the mean of the neurons over time:

$$\langle s_i \rangle = \frac{1}{T} \sum_{t=1}^{T} s_i^t$$

The covariance matrix of the salamander dataset as the following summation:

$$\langle s_i s_j \rangle = \frac{1}{T} \sum_{t=1}^{T} s_i^t s_k^t$$

Where now the upper indices t denotes the neuron state at different times and T is the total amount of time steps.

4 Results

4.1 Simulating Ising Models

We first present our Boltzmann Machine algorithms applied to the Toy Ising Model, using the methods described in the previous sections. All plots have the set parameters as followed: $N = 10$ spins. $T = 1$, $\mu_J = 0.01$, $\mu_h = 0.2$, $\sigma = 0.1$. $\eta = 0.1$ and 500 epochs.

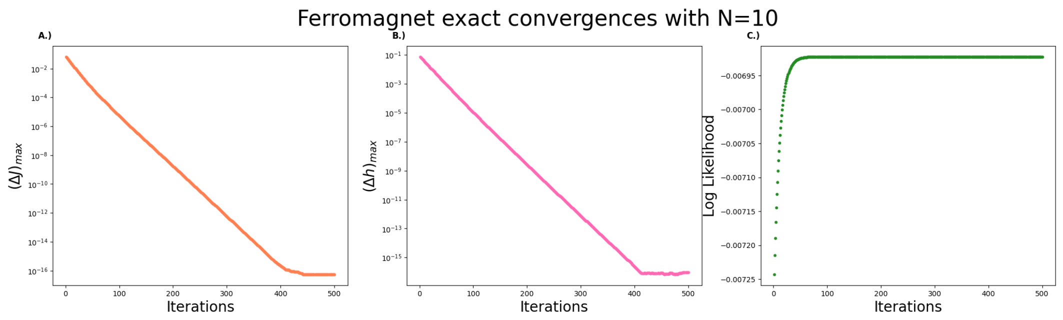

Figure 1: BM on the Ferromagnet Toy Model, generating the free statistics using exact computations

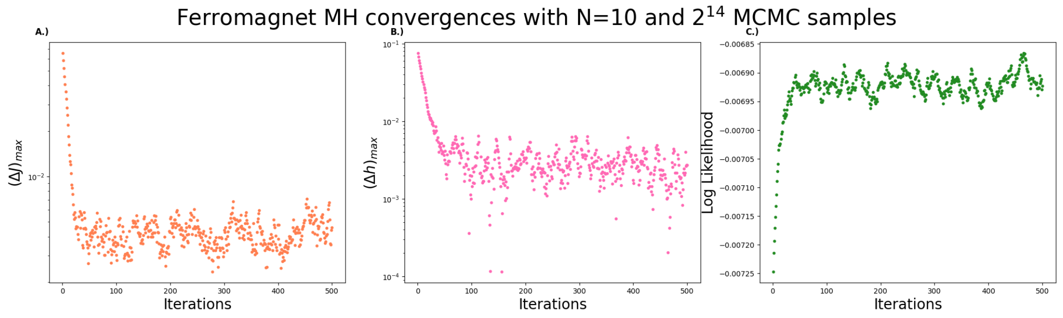

Figure 2: Ferromagnet Toy Model using the Metropolis Hasings algorithm and 2^{14} samples to compute free statistics at each epoch

For the exact computations of the free statistics (Figure 1), we see very clear convergences to the absolute distance between our inferred parameters and our oracle parameters, as expected. Moreover, the log likelihood has a clear increasing trend and reached a peak rapidly.

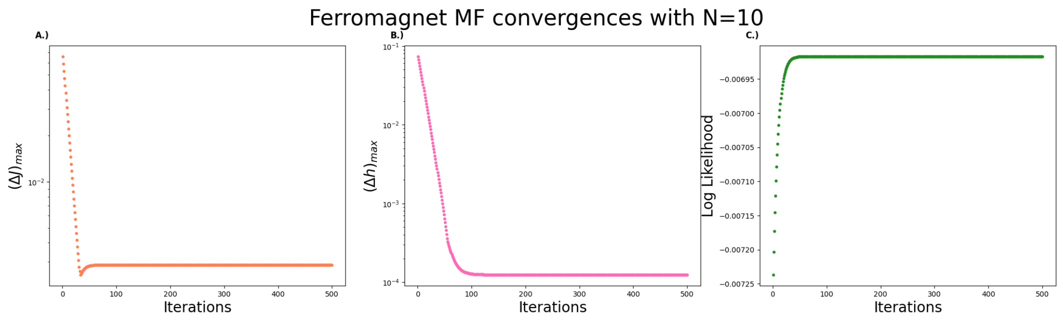

Figure 3: Mean Field and Linear Response algorithm used to infer the weight parameters of the Toy Ferromagnet Model, where we used a maximum of 100 iterations for MF convergence at each epoch

If we apply the sampling with a MH scheme (Figure 2), the convergences of the weights differences and the log likelihood are noisy. Despite the noisy behaviour, the curves follow the expected increasing trend. As we already mentioned, the inherent noise in the MH simulation is proportional to $\sigma = \frac{1}{\sqrt{P}}$, with P being the number of samples we generate. We see that the algorithm is only able to converge towards the magnitude of the variance. For the free statistics $\langle s_i \rangle$ and $\langle s_i s_j \rangle$ we need more than $10^6 \approx 2^{20}$ samples per iteration of the gradient descent loop to get the parameters to converge to the order of 10^{-3}. However, in this report, we only used 2^{14} samples and as observed, the convergence averages to above 10^{-3} with these settings for 500 iterations.

Finally, implementing the Mean Field and Linear Response algorithm, as shown in Figure 3, we once again see a very clear convergence to the peak of the log likelihood and good performance in finding the weights and biases of the Boltzmann Machine.

An advantage of Metropolis Hastings is that it can handle larger lattice sizes. This is because MH-sampling scales linear in N, while exact computation scales exponentially in N. On the other hand, the Mean Field algorithm is only good when the parameters are small i.e. high temperature regime.

4.2 Salamander Retina Data

Now, we present the resulting graphs of applying our algorithm to the salamander retina data. The fixed parameters are as follows: $N = 10$, $T = 1$, $\mu_J = 0$, $\mu_h = 0$, $\sigma = 0.01$, $\eta = 0.1$, 100 Epochs.

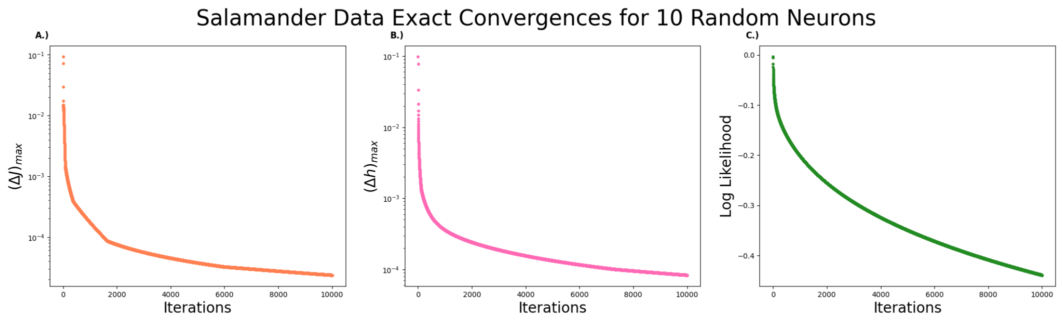

Figure 4: Illustrates the exact computation of the free statistics parameters on 10 randomly selected neurons of the salamander data

First, we demonstrate, in figure 4, our application of the exact computation algorithm on 10 randomly selected neurons from the 160 neurons of the salamander retina. As mentioned in

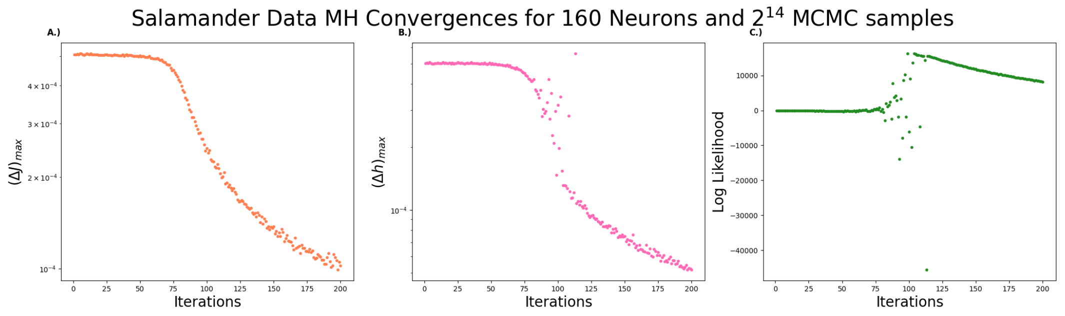

Figure 5: Metropolis Hastings algorithm applied for the learning of the BM parameters for the full set of 160 neurons of the salamander retina data

the Evaluation Metric section above, for this part of the assignment, as we do not have the oracle parameters, the first two subplots of the figure evidence the difference in parameters of the current epoch and the previous one.

Implementing the Metropolis Hastings algorithm, but now in all 160 neurons, as shown in figure 5, we notice the correct decrease in weight differences throughout the iterations. We observe a clear decrease after about 75 iterations and then a slow convergence. This would perhaps become more evident from running our program for more iterations. However, for the purpose of this illustration, it suffices, as this decrease shows the correct function of our algorithms.

On the other hand, we can notice a rather off behavior in the C) subplots, the log likelihood plots, for both runs on the restricted and full salamander data (Figures 4 and 5).

There could be several explanation for this behaviour: first, it could indicate an issue with the likelihood computation. This would be an error in the implementation of the evaluation metrics, and would not necessarily imply that the Boltzmann machine is not learning properly. On the other hand, it is also likely that there is an error in the preprocessing of our data, which would then imply that the data statistics are computed wrongly. Thus, the convergence of the learning procedure to a distribution that doesn't reflect the one of the data.

We could not at all apply the MF algorithm to the salamander data. Unfortunately, the learning algorithm stops after a few iterations after finding that the approximation of the mean spin values consists of -1 values. This leads to a divide by 0 error encountered in the implementation of Equation 12, which we ultimately could not fix.

5 Conclusion

Despite some difficulties to apply the learning procedures to the Salamander Dataset, we implemented successfully three different procedures to train Boltzmann Machines on small synthetic datasets.

Regarding the unsuccessful completion of the same task for the Salamander dataset, we hypothesize it might be due wither to the implementation of the likelihood function or to the data preprocessing. Further limitations might be the scalability or computational cost of the salamander data. As shown in figure 5, convergence might have become more clear with more iterations. However, that would be more computationally costly. Additionally, as mentioned in the Ising Model figure 2, there is also the potential problem of being stuck in the local minima, rather than the global one.

A future exploration of this assignment's tasks would imply further investigation into the likelihood function implementation due to its malfunction on the Salamander Data implementation, and checking the Mean Field application on this data.

References

[1] G. Parisi, *Statistical Field Theory*. 1988.

[2] H. J. Kappen and F. B. Rodríguez, "Efficient Learning in Boltzmann Machines Using Linear Response Theory," *Neural Computation*, vol. 10, no. 5, pp. 1137–1156, Jul. 1998, ISSN: 0899-7667. DOI: 10.1162/089976698300017386.