Statistical Inference

Lukas Helmig

September 2, 2024

1 Theory

How is the Boltzmann machine learning algorithm obtained from the objective to maximize the log-likelihood?

First, we went from optimizing the Kullback-Leiber divergence to optimizing just the negative log-likelihood:

$$\mathcal{L}(\theta) = -\frac{1}{M}\sum_k \log P_\theta(\mathbf{s}^{(k)}). \tag{1}$$

To optimize, we can use a gradient descent approach. The gradient is here given by

$$\nabla = \begin{pmatrix} \partial_{h_1} \\ \dots \\ \partial_{h_n} \\ \partial_{J_{ij}} \\ \dots \\ \partial_{J_{i'j'}} \end{pmatrix} \tag{2}$$

and we follow the iterative procedure

$$\theta^{(n+1)} = \theta^{(n)} - \eta \nabla \mathcal{L}(P_\theta). \tag{3}$$

We first calculate $\nabla \mathcal{L}(P_\theta)$

$$\begin{aligned} \partial_{h_i}\mathcal{L}(P_\theta) &= -\frac{1}{M}\sum_k \partial_{h_i} \log P_\theta(\mathbf{s}^{(k)}) && (4)\\ &= -\frac{1}{M}\sum_k \frac{1}{P_\theta(\mathbf{s}^{(k)})}\partial_{h_i} P_\theta(\mathbf{s}^{(k)}). && (5) \end{aligned}$$

The derivative $\partial_{h_i} P_\theta(\mathbf{s}^{(k)})$ takes the following form

$$\begin{aligned} \partial_{h_i} P_\theta(\mathbf{s}) = \frac{1}{Z}\partial_{h_i} \exp(\sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j) \\ - \frac{1}{Z} P_\theta(\mathbf{s}) \partial_{h_i} Z \end{aligned} \tag{6}$$

$$\begin{aligned} \partial_{h_i} P_\theta(\mathbf{s}) &= s_i \cdot P_\theta(\mathbf{s}) - \langle s_i \rangle_\theta \cdot P_\theta(\mathbf{s}) && (7)\\ \partial_{J_{ij}} P_\theta(\mathbf{s}) &= s_i s_j \cdot P_\theta(\mathbf{s}) - \langle s_i s_j \rangle_\theta \cdot P_\theta(\mathbf{s}). && (8) \end{aligned}$$

Inserting this back into equation 5, we arrive at

$$\begin{aligned} \partial_{h_i}\mathcal{L}(P_\theta) &= \langle s_i \rangle_\theta \frac{1}{M}\sum_k 1 - \frac{1}{M}\sum_k s_i^{(k)} && (9)\\ &= \langle s_i \rangle_\theta - \langle s_i \rangle^D, && (10) \end{aligned}$$

which gives us the Boltzmann learning algorithm.

The model involves the partition sum with an exponential number of terms. Why does this exponential sum not affect the efficiency of the learning algorithm?

Calculating the partition sum is nearly impossible for large system sizes. However, we still need the thermal average for a set of model parameters. The solution is to set up a Markov chain Monte Carlo algorithm that allows us to calculate thermal averages without using the partition function.

Derive the derivatives of the non-equilibrium likelihood, which you need to implement the corresponding gradient descent optimization.

Taking the derivate $\partial_{\Theta_j}\theta$ brings us to

$$\begin{aligned}\partial_{\Theta_j}\theta &= \frac{1}{M}\sum_{t=1}^{M-1} s_j(t+1) - \partial_{\Theta_j}\ln(2\cosh(\Theta_j(t))) \\ &= \frac{1}{M}\sum_{t=1}^{M-1} s_j(t+1) - \tanh(\Theta_j) \\ &= \langle s_j\rangle_t - \tanh(\Theta_j). \end{aligned} \tag{11}$$

The iterative model parameters between steps of the gradient descent optimization are then given by

$$\Theta_i^{(n+1)} = \Theta_i^{(n)} + \eta\tanh(\Theta_i^{(n)}) - \eta\langle s_j\rangle_t, \tag{12}$$

where $\langle s_j\rangle_t$ the average over the time series.

2 Thermal distributions

Monitor the negative log-likelihood during the training procedure. Is the value being reduced systematically? How does this depend on the learning rate?

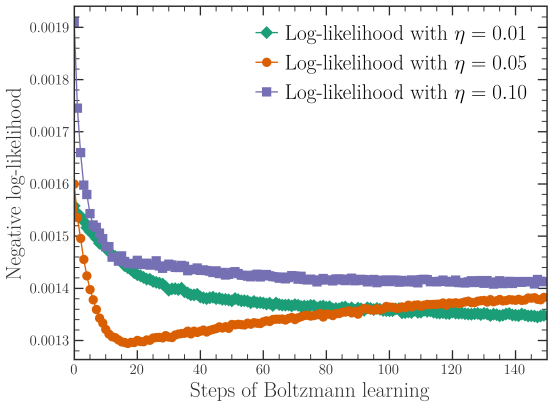

Figure 1: Minimization of the negative log-likelihood for different learning rates.

Figure 1 shows how the gradient descent algorithm minimizes the negative log-likelihood. It is seen that the lowest learning rate of $\eta = 0.01$ shows the slowest decline towards the minimal value. In contrast to that, a higher learning rate of $\eta = 0.05$ shows even that the current negative log-likelihood "overshoots" the minimal value of the negative log-likelihood. I expected the highest learning rate to behave more like the curve of $\eta = 0.05$, but this different behavior might stem from the different random couplings the simulations started with.

How does the quality of the inference depend on the number of training data?

Figures 2 and 3 show the infered parameters in comparison to the true model parameters for a varying size of the training data set. It is seen that even for a small set size of 50.000, the infered parameters agree quite accurately with the true model parameters. But it is noticeable that infering parameters works best, if the size of the training data is large.

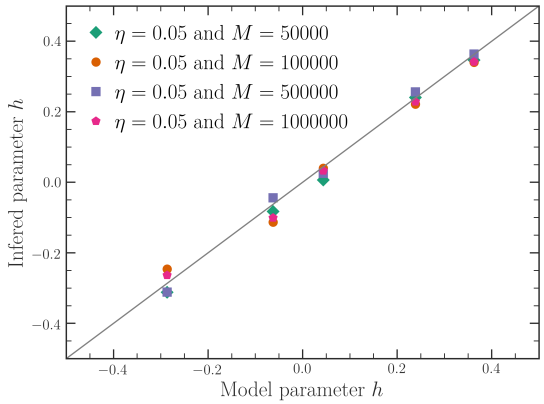

Figure 2: Infered parameters h in comparison to the true parameters h of the model.

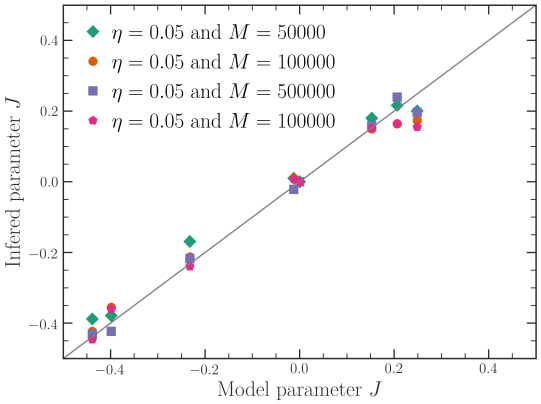

Figure 3: Infered parameters J in comparison to the true parameters J of the model.

3 The brain of the salamander - Inferring an equilibrium model

How well does the distribution of that spin conditioned on the others describe the missing data?

To test this, the training data set was cut in half in two ways: one way, where we just cut the training data set in the middle, and one way, where every second data point was assigned to the training data. Since, there is no really good measure of "how well" we describe the remaining data set with the inferred model, the normal euklidean norm is used to measure this. To be precise, the euklidean norm of

$$\langle s_i \rangle_{\text{inferred}} - \langle s_i \rangle_{\text{remaining data}} \tag{13}$$
$$\langle s_i s_j \rangle_{\text{inferred}} - \langle s_i s_j \rangle_{\text{remaining data}} \tag{14}$$

is calculated. The resulting distance per spin site is seen in table 1. One can notice, that slicing the training data set in half leads to better agreement with the remaining data set. In this way, the temporal correlation between training data set and control set is reduced to a minimum.

Method	Distance $\langle s_i \rangle$	Distance $\langle s_i s_j \rangle$
Cut in half	$5.871 \cdot 10^{-3}$	$0.664 \cdot 10^{-3}$
Every 2nd	$35.265 \cdot 10^{-3}$	$3.452 \cdot 10^{-3}$

Table 1: Success of inference depending on the way, the training data set is sliced.

To further test the model, calculate the correlations between triplets of spins both in the data and from your model and plot them against each other.

Figure 4 shows that the equilibrium model does not capture the three spin correlation at all. The correlation of three spins is around 0 in the training data, but the inferred model procudes a three spin correlation of around -1. To accurately describe these correlations, a three spin interaction needs to be added to the Hamiltonian, so it makes sense that our model is not capable of describing this. In conclusion, the equilibrium model is badly suited for modeling the activity in the salamanders brain.

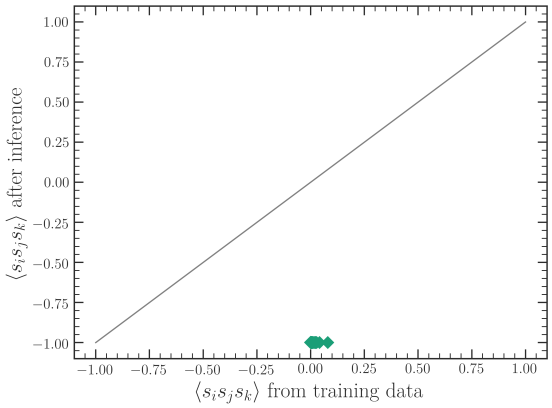

Figure 4: Three spin correlation with inferenced parameters compared to the three spin correlation of the training data.

4 The brain of the salamander - Inferring a non-equilibrium model

Infer the couplings and fields used to generate the training data. How accurately are the parameters recovered?

Figures 5 and 6 show the quality of the inference for $N = 5$. The gradient descent was done for 150 iterations and the size of the training data set was 100000 with a learning rate of $\eta = 0.1$. It is astounding how well the inferred parameters agree with the true parameters used to generate the data.

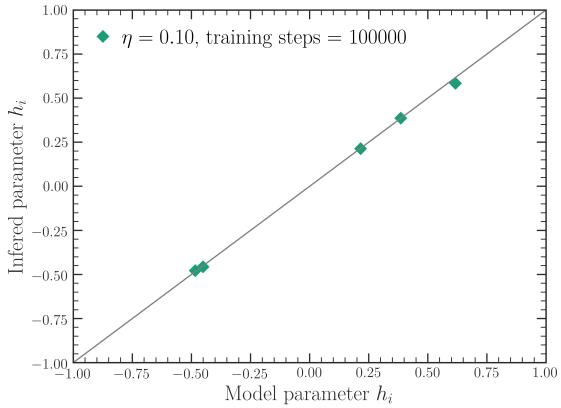

Figure 5: Infered parameters h in comparison to the true parameters h of the non-equilibrium model. 150 iterations of the gradient descent were done.

Infer symmetric couplings $J_{ij} = J_{ji}$ and fields that best describe the Salamander's brain activity. How well does the inferred model describe temporal correlations of the data?

To infer symmetric couplings, we need to make sure that these couplings start from the same value and that we adapt the parameters symmetrically. In practice, I symmetrized the change of $\Delta J'_{ij}$ via $\Delta J'_{ij} = 0.5 \cdot (\Delta J_{ij} + \Delta J_{ji})$.

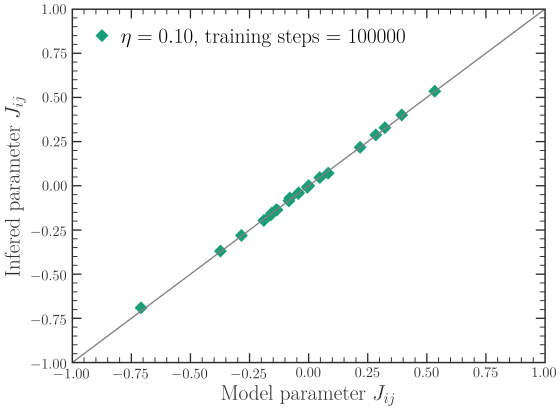

Figure 6: Infered parameters J in comparison to the true parameters J of the non-equilibrium model. 150 iterations of the gradient descent were done.

Figure 7 shows the comparison of temporal correlations. It is seen that the symmetric model captures the temporal correlations quite good.

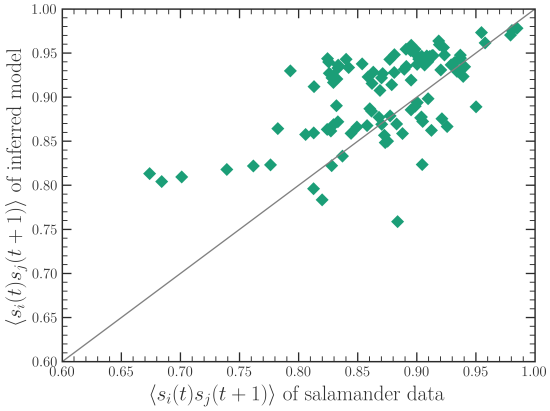

Figure 7: Comparison of temporal correlations generated with the inferred model with symmetric J_{ij} to temporal correlations of the data. The learning rate was $\eta = 0.05$, 150 iterations of gradient descent.

Infer couplings J_{ij} that are not necessarily symmetric and fields that best describe the

Salamander's brain activity. By comparing the resulting likelihoods, does the model with asymmetric couplings lead to a better description of the data?

First of all, figure 8 shows that the model accurately reproduces the temporal correlations from the data. In comparison to figure 7, it can be seen that the accuracy increased when taking the couplings to be asymmetric.

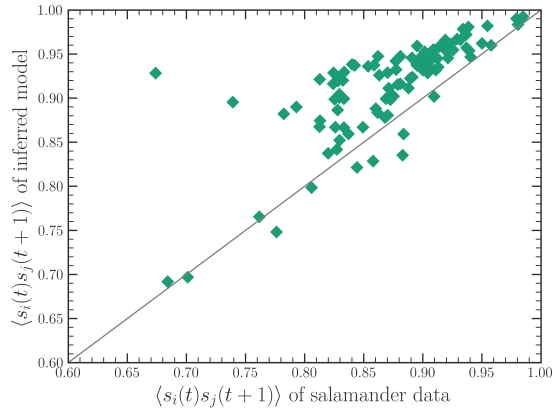

Figure 8: Comparison of temporal correlations generated with the inferred model with asymmetric J_{ij} to temporal correlations of the data. The learning rate was $\eta = 0.05$, 150 iterations of gradient descent.

Figure 9 shows a comparison of the resulting log-likelihood. It is seen that the simulation with symmetric couplings converges faster than the simulation with the asymmetric couplings. Overall, the simulation with symmetric couplings produces a end log-likelihood of 18.3567, where as the simulation with asymmetric couplings ended with a log-likelihood of 18.9343. Overall, the ansatz with symmetric couplings does seem to deliver a better result in terms of the log-likelihood. However, we have to keep in mind that the asymmetric couplings seem to deliver the best result at reproducing the temporal correlations from the data.

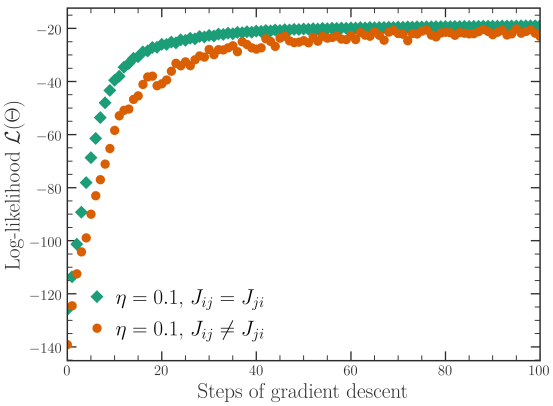

Figure 9: Log-likelihood of the inference with symmetric and asymmetric couplings. Both simulations were done with $\eta = 0.1$ and 200 iterations of gradient descent.