Faster solutions of the inverse pairwise Ising problem

Tamara Broderick,[a] Miroslav Dudík,[b] Gašper Tkačik,[c] Robert E. Schapire[b] and William Bialek[c,d]
[a] *Department of Mathematics,* [b] *Department of Computer Science,*
[c] *Joseph Henry Laboratories of Physics,* [c] *Lewis–Sigler Institute for Integrative Genomics,*
and [d] *Princeton Center for Theoretical Physics, Princeton University, Princeton, NJ 08544*

(Dated: December 15, 2007)

Recent work has shown that probabilistic models based on pairwise interactions—in the simplest case, the Ising model—provide surprisingly accurate descriptions of experiments on real biological networks ranging from neurons to genes. Finding these models requires us to solve an inverse problem: given experimentally measured expectation values, what are the parameters of the underlying Hamiltonian? This problem sits at the intersection of statistical physics and machine learning, and we suggest that more efficient solutions are possible by merging ideas from the two fields. We use a combination of recent coordinate descent algorithms with an adaptation of the histogram Monte Carlo method, and implement these techniques to take advantage of the sparseness found in data on real neurons. The resulting algorithm learns the parameters of an Ising model describing a network of forty neurons within a few minutes. This opens the possibility of analyzing much larger data sets now emerging, and thus testing hypotheses about the collective behaviors of these networks.

I. INTRODUCTION

In the standard problems of statistical mechanics, we begin with the definition of the Hamiltonian and proceed to calculate the expectation values or correlation functions of various observable quantities. In the inverse problem, we are given the expectation values and try to infer the underlying Hamiltonian. The history of the inverse problems goes back (at least) to 1959, when Keller and Zumino [1] showed that, for a classical gas, the temperature dependence of the second virial coefficient determines the interaction potential between molecules uniquely, provided that this potential is monotonic. Subsequent work on classical gases and fluids considered the connection between pair correlation functions and interaction potentials in various approximations [2], and more rigorous constructions of Boltzmann distributions consistent with given spatial variations in density [3] or higher order correlation functions [4].

In fact the inverse problem of statistical mechanics arises in many different contexts, with several largely independent literatures. In computer science, there are a number of problems where we try to learn the probability distribution that describes the observed correlations among a large set of variables in terms of some (hopefully) simpler set of interactions. Many algorithms for solving these learning problems rest on simplifications or approximations that correspond quite closely to established approximation methods in statistical mechanics [5]. More explicitly, in the context of neural network models [6, 7], a family of models referred to as 'Boltzmann machines' lean directly on the mapping of probabilistic models into statistical physics problems, identifying the parameters of the probabilistic model with the coupling constants in an Ising–like Hamiltonian [8].

Inverse problems in statistical mechanics have received new attention because of attempts to construct explicit network models of biological systems. Physicists have long hoped that the collective behavior which emerges from statistical mechanics could provide a model for the emergence of function in biological systems, and this general class of ideas has been explored most fully for networks of neurons [6, 7]. Recent work shows how these ideas can be linked much more directly to experiment [9, 10] by searching for maximum entropy models that capture some limited set of measured correlations. At a practical level, implementing this program requires us to solve a class of inverse problems for Ising models with pairwise interactions among the spins, and this is the problem that we consider here.

To be concrete, we consider a network of neurons. Throughout the brain, neurons communicate by generating discrete, identical electrical pulses termed action potentials or spikes. If we look in a small window of time, each neuron either generates a spike or it does not, so that there is a natural description of the instantaneous state of the network by a collection of binary or Ising variables; $\sigma_{\rm i} = +1$ indicates that neuron i generates a spike, and $\sigma_{\rm i} = -1$ indicates that neuron i is silent. Knowing the average rate at which spikes are generated by each cell is equivalent to knowing the expectation values $\langle\sigma_{\rm i}\rangle$ for all i. Similarly, knowing the probabilities of coincident spiking (correlations) among all pairs of neurons is equivalent to knowing the expectation values $\langle\sigma_{\rm i}\sigma_{\rm j}\rangle$. Of course there are an infinite number of probability distributions $P(\sigma)$ over the states of the whole system ($\sigma \equiv \{\sigma_{\rm i}\}$) that are consistent with these expectation values, but if we ask for the distribution that is as random as possible while still reproducing the data—the maximum entropy distribution—then this has the form of an Ising model with pairwise interactions:

$$P(\sigma) = \frac{1}{Z}\exp\left[-\sum_{\rm i} h_{\rm i}\sigma_{\rm i} - \frac{1}{2}\sum_{\rm i\neq j} J_{\rm ij}\sigma_{\rm i}\sigma_{\rm j}\right]. \quad (1)$$

The inverse problem is to find the "magnetic fields" $\{h_{\rm i}\}$ and "exchange interactions" $\{J_{\rm ij}\}$ that reproduce the observed values of $\langle\sigma_{\rm i}\rangle$ and $\langle\sigma_{\rm i}\sigma_{\rm j}\rangle$.

The surprising result of Ref [9] was that this Ising model provides an accurate quantitative description of the combinatorial patterns of spiking and silence observed in groups of order $N = 10$ neurons in the retina as it responds to natural sensory inputs, despite only taking account of pairwise interactions. The Ising model allows us to understand how, in this system, weak correlations among pairs of neurons can coexist with strong collective effects at the network level, and this is even clearer as one extends the analysis to larger groups (using real data for $N = 40$, and extrapolating to $N = 120$), where there is a hint that the system is poised near a critical point in its dynamics [10]. Since the initial results, a number of groups have found that the maximum entropy models provide surprisingly accurate descriptions of other neural systems [11, 12, 13] and similar approaches have been used to look at biochemical and genetic networks [14, 15].

The promise of the maximum entropy approach to biological networks is that it builds a bridge from easily observable correlations among pairs of elements to a global view of the collective behavior that can emerge from the network as a whole. Clearly this potential is greatest in the context of large networks. Indeed, even for the retina, methods are emerging that make it possible to record simultaneously from hundreds of neurons [16, 17], so just keeping up with the data will require methods to deal with much larger instances of the inverse problem. The essential difficulty, of course, is that once we have a large network, even checking that a given set of parameters $\{h_{\rm i}, J_{\rm ij}\}$ reproduce the observed expectation values requires a difficult calculation. In Ref [10] we took an essentially brute force Monte Carlo approach to this part of the problem, and then adjusted the parameters to improve the match between observed and predicted expectation values using a relatively naive algorithm.

In this work we combine several ideas—taken both from statistical physics and from machine learning [18]—which seem likely to help arrive at more efficient solutions of the inverse problem for the pairwise Ising model. First, we adapt the histogram Monte Carlo method [19] to 'recycle' the Monte Carlo samples that we generate as we make small changes in the parameters of the Hamiltonian. Second, we use a coordinate descent method to adjust the parameters [20]. Finally, we exploit the fact that neurons use their binary states in a very asymmetric fashion, so that silence is much more common that spiking. Combining these techniques, we are able to solve the inverse problem for $N = 40$ neurons in tens of minutes, rather than many days for the naive approach, holding out hope for generalization to yet larger problems.

II. INGREDIENTS OF OUR ALGORITHM

A. Basic formulation

Our overall goal is to build a model for the distribution $P(\sigma)$ over the states of the a system with N elements, $\sigma \equiv \{\sigma_1, \sigma_2, \cdots, \sigma_N\}$. As ingredients for determining this model, we use low order statistics computed from a set of m samples $\{\sigma^1, \sigma^2, \cdots, \sigma^m\}$, which we can think of as samples drawn from the distribution $P(\sigma)$. The classical idea of maximum entropy models is that we should construct $P(\sigma)$ to generate the correct values of certain average quantities (e.g., the energy in the case of the Boltzmann distribution), but otherwise the distribution should be 'as random' as possible [21]. Formally this means that we find $P(\sigma)$ as the solution of a constrained optimization problem, maximizing the entropy of the distribution subject to conditions that enforce the correct expectation values. We will refer to the quantities whose averages are constrained as "features" of the system, $\mathbf{f} \equiv \{f_1, f_2, \cdots, f_K\}$, where each f_μ is a function of the state σ, $f_\mu(\sigma)$.

One special set of average features are just the marginal distributions for subsets of the variables. Thus we can construct the one–body marginals

$$P_{\rm i}(\sigma_{\rm i}) = \sum_{\{\sigma_{\rm j\neq i}\}} P(\sigma_1, \sigma_2, \cdots, \sigma_N), \tag{2}$$

the two–body marginals,

$$P_{\rm ij}(\sigma_{\rm i}, \sigma_{\rm j}) = \sum_{\{\sigma_{\rm k\neq i,j}\}} P(\sigma_1, \sigma_2, \cdots, \sigma_N), \tag{3}$$

and so on for larger subsets. The maximum entropy distributions consistent with marginal distributions up to K–body terms generates a hierarchy of models that capture increasingly higher–order correlations, monotonically reducing the entropy of the model as K increases, toward the true value [22].

Let $\tilde{P}$ denote the empirical distribution

$$\tilde{P}(\sigma) = \frac{1}{m}\sum_{n=1}^{m} \delta(\sigma, \sigma^n), \tag{4}$$

where $\delta(\sigma, \sigma')$ is the Kronecker delta, equal to one when $\sigma = \sigma'$ and equal to zero otherwise. The maximum-entropy problem is then

$$\max_P S[P] \text{ such that } \langle \mathbf{f}(\sigma)\rangle_P = \langle \mathbf{f}(\sigma)\rangle_{\tilde{P}} \ , \tag{5}$$

where $S[P]$ denotes the entropy of the distribution P, and $\langle\cdots\rangle_P$ denotes an expectation value with respect to that distribution. Using the method of Lagrange multipliers, the solution to the maximum entropy problem has the form of a Boltzmann or Gibbs distribution [21],

$$Q_\theta(\sigma) = \frac{1}{\mathbf{Z}(\theta)} \exp\left[-\sum_{\mu=1}^{K} \theta_\mu f_\mu(\sigma)\right], \tag{6}$$

where as usual the partition function $\mathbf{Z}(\theta)$ is the normalization constant ensuring that the distribution Q_θ sums to one, and the parameters $\{\theta_1, \theta_2, \cdots, \theta_K\} \equiv \theta \in \mathbb{R}^K$ correspond to the Lagrange multipliers. Note that the expression for Q_θ above describes the pairwise Ising model,

Eq (1), with $\theta = \{h_i, J_{ij}\}$, and the features $\mathbf{f}$ are the one–spin and two–spin combinations $\{\sigma_i, \sigma_i\sigma_j\}$.

Rather than thinking of our problem as that of maximizing the entropy subject to constraints on the expectation values, we can now think of our task as searching the space of Gibbs distributions, parameterized as in Eq (6), to find the values of the parameters θ that generate the correct expectation values. Importantly, because of basic thermodynamic relationships, this search can also be formulated as an optimization problem. Specifically, we recall that expectation values in statistical mechanics can be written as derivatives of the free energy, which in turn is the logarithm of the partition function (up to factors of the temperature, which isn't relevant here). Thus, for distribution in the form of Eq (6), we have

$$\langle f_\mu(\sigma)\rangle_{Q_\theta} = -\frac{\partial \ln \mathbf{Z}(\theta)}{\partial \theta_\mu}. \tag{7}$$

Enforcing that these expectation values are equal to the expectation values computed from our empirical samples means solving the equation

$$\langle f_\mu(\sigma)\rangle_{\tilde{P}} \equiv \frac{1}{m}\sum_{n=1}^{m} f_\mu(\sigma^n) = -\frac{\partial \ln \mathbf{Z}(\theta)}{\partial \theta_\mu}. \tag{8}$$

But this can be written as

$$\frac{1}{m}\sum_{n=1}^{m} f_\mu(\sigma^n) = \frac{1}{m}\sum_{n=1}^{m}\frac{\partial}{\partial\theta_\mu}\sum_{\nu=1}^{K}\theta_\nu f_\nu(\sigma^n) = -\frac{\partial \ln \mathbf{Z}(\theta)}{\partial \theta_\mu} \tag{9}$$

$$0 = \frac{\partial}{\partial\theta_\mu}\frac{1}{m}\sum_{n=1}^{m}\left[-\ln \mathbf{Z}(\theta) - \sum_{\nu=1}^{K}\theta_\nu f_\nu(\sigma^n)\right] \tag{10}$$

$$= \frac{\partial}{\partial\theta_\mu}\frac{1}{m}\sum_{n=1}^{m}\ln Q_\theta(\sigma^n). \tag{11}$$

Thus we see that matching the empirical expectation values is equivalent to looking for a local extremum (which turns out to be a maximum) of the quantity

$$\frac{1}{m}\sum_{n=1}^{m}\ln Q_\theta(\sigma^n). \tag{12}$$

But if all the samples $\{\sigma^1, \sigma^2, \cdots, \sigma^n\}$ are drawn independently, then the total probability that the Gibbs distribution with parameters θ generates the data is $P_{\rm total} = \prod_n Q_\theta(\sigma^n)$, and so the quantity in Eq (12) is (up to a factor of m), just $\ln P_{\rm total}$. Finding the maximum entropy distribution thus is equivalent to maximizing the probability or likelihood that our model generates the observed samples, within the class of models defined by the Gibbs distribution Eq (6).

We recall that, in information theory [23], probability distributions implicitly define strategies for encoding data, and the shortest codes are achieved when our model of the distribution actually matches the distribution from which the data are drawn. Since code lengths are related to the negative logarithm of probabilities, it is convenient to define the cost of coding or log loss $\mathrm{L}_{\tilde{P}}(\theta)$ that arises when we use the model with parameters θ to describe data drawn from the empirical distribution $\tilde{P}$:

$$\mathrm{L}_{\tilde{P}}(\theta) = -\frac{1}{m}\sum_{n=1}^{m}\ln Q_\theta(\sigma^n) = \langle -\ln Q_\theta(\sigma)\rangle_{\tilde{P}} \ . \tag{13}$$

Comparing with Eq (12), we obtain the *dual* formulation of the maximum entropy problem,

$$\min_\theta \mathrm{L}_{\tilde{P}}(\theta). \tag{14}$$

Why is the optimization problem in Eq (14) difficult? In principle, the convexity properties of free energies should make the problem well behaved and tractable. But it remains possible for $\mathrm{L}_{\tilde{P}}(\theta)$ to have a very sensitive dependence on the parameters θ, and this can cause practical problems, especially if, as suggested in Ref [10], the systems we want to describe are poised near a critical point. Even before encountering this problem, however, we face the difficulty that computing $\mathrm{L}_{\tilde{P}}(\theta)$ or even its gradient in parameter space involves computing expectation values with respect to the distribution $Q_\theta(\sigma)$. Once the space of states becomes large, it is no longer possible to do this by exact enumeration. We can try to use approximate analytical methods, or we can use Monte Carlo methods.

B. Monte Carlo methods

Our general strategy for solving the optimization problem in Eq (14) will be to use standard Monte Carlo simulations [24, 25, 26] to generate samples from the distribution $Q_\theta(\sigma)$, approximate the relevant expectation values as averages over these samples, and then use the results

to propose changes in the parameters θ so as to proceed toward a minimum of $\mathrm{L}_{\tilde{P}}(\theta)$. Implemented naively, as in Ref [10], this procedure is hugely expensive, because at each new setting of the parameters we have to generate a new set of Monte Carlo samples. Some of this cost can be avoided using the ideas of histogram Monte Carlo [19].

We recall that if we want to compute the expectation value of some function $\Phi(\sigma)$ in the distribution $Q_{\theta'}(\sigma)$, this can be written as

$$\langle\Phi(\sigma)\rangle_{\theta'} \equiv \sum_\sigma Q_{\theta'}(\sigma)\Phi(\sigma) \quad (15)$$

$$= \sum_\sigma Q_\theta(\sigma)\left[\frac{Q_{\theta'}(\sigma)}{Q_\theta(\sigma)}\Phi(\sigma)\right] \quad (16)$$

$$= \left\langle \frac{Q_{\theta'}(\sigma)}{Q_\theta(\sigma)}\Phi(\sigma)\right\rangle_\theta \quad (17)$$

$$= \frac{\langle\Phi(\sigma)\exp[-(\theta'-\theta)\cdot\mathbf{f}(\sigma)]\rangle_\theta}{\langle\exp[-(\theta'-\theta)\cdot\mathbf{f}(\sigma)]\rangle_\theta}, \quad (18)$$

where we denote expectation values in the distribution $Q_\theta(\sigma)$ by $\langle\cdots\rangle_\theta$, and similarly for θ'. We note that Eq (18) is exact. The essential step of histogram Monte Carlo is to use an approximation to this equation, replacing the expectation value in the distribution Q_θ by an average over a set of samples drawn from Monte Carlo simulation of this distribution.

Consider an algorithm $\mathcal{A}$ which searches for a minimum of $\mathrm{L}_{\tilde{P}}(\theta)$. As this algorithm progresses, the values of the parameters θ change slowly. We will divide these changes into stages $s = 1, 2, \cdots$, and within each stage we will perform $t = 1, 2, \cdots, T$ iterations. At the first iteration, we will generate, via Monte Carlo, M samples of the state σ drawn out of the distribution appropriate to the current value of $\theta = \theta(s, t = 1)$. Let us refer to averages over these samples as $\langle\cdots\rangle_{MC\theta}$, which should approximate $\langle\cdots\rangle_\theta$. At subsequent iterations, the parameters θ will be adjusted (see below for details), but we keep the same Monte Carlo samples and approximate the expectation values over the distribution with parameters $\theta' = \theta(s, t)$ as

$$\langle\Phi(\sigma)\rangle_{\theta'} \approx \frac{\langle\Phi(\sigma)\exp[-(\theta'-\theta)\cdot\mathbf{f}(\sigma)]\rangle_{MC\theta}}{\langle\exp[-(\theta'-\theta)\cdot\mathbf{f}(\sigma)]\rangle_{MC\theta}}. \quad (19)$$

We denote this approximation as $\langle\cdots\rangle_{\theta'} \approx \langle\cdots\rangle_{\theta'|\theta}$. Once we reach $t = T$, we run a new Monte Carlo simulation appropriate to the current value of θ, and the cycle begins again at stage $s = s + 1$. This strategy is summarized as pseudocode in Fig 1.

Once we chose the optimization algorithm $\mathcal{A}$, there are still two free parameters in our scheme: the number of samples M provided by the Monte Carlo simulation at each stage, and the number of iterations per stage T. In our experiments, we explore how choices of these parameters influence the convergence of the resulting algorithms.

C. Parameter adjustment and the use of sparsity

At the core of our problem is an algorithm which tries to adjust the parameters θ so as to minimize $\mathrm{L}_{\tilde{P}}(\theta)$. In Ref [10] we used a simple gradient descent method. Here we use a coordinate descent method [20], adapted from Ref [27] and tuned specifically for the maximum entropy problem. Where gradient descent methods attempt to find a vector in parameter space along which one can achieve the maximum reduction in L, coordinate descent methods explore in sequence the individual parameters θ_μ.

Beyond the general considerations in Refs [20, 27], coordinate descent may be especially well suited to our problem because of the the sparsity of the feature vectors. In implementing any parameter adjustment algorithm, it is useful to take account of the fact that in networks of real neurons, spikes and silences are used very asymmetrically. It thus is useful to write the basic variables as $n_\mathrm{i} = (\sigma_\mathrm{i} + 1)/2 \in \{0, 1\}$. This involves a slight redefinition of the parameters $\{h_\mathrm{i}, J_\mathrm{ij}\}$, but once this is done one finds that individual terms $\propto n_\mathrm{i}$ or $\propto n_\mathrm{i}n_\mathrm{j}$ are often zero, because spikes (corresponding to $n_\mathrm{i} = 1$) are rare. This is true not just in the experimental data, but of course also in the Monte Carlo simulations once we have parameters that approximately reproduce the overall probability of spiking vs. silence. This sparsity can lead to substantial time savings, since all expectation values can be evaluated in time proportional to the number of non–zero elements [28].

As an alternative to coordinate descent method, we also used a general purpose convex optimization algorithm known as the limited memory variable metric (LMVM) [29], which is known to outperform others such general purpose algorithms on a wide variety of problems [30]. While LMVM has the advantage of changing multiple parameters at a time, the cost of updating may outweigh this advantage, especially for very sparse data such as ours.

Both the coordinate descent and the LMVM schemes are initialized with parameters [in the notation of Eq (1)] $J_\mathrm{ij} = 0$ and the h_i chosen to exactly reproduce the expectation values $\langle\sigma_\mathrm{i}\rangle$. Our Monte Carlo method follows the discussion of the Gibbs sampler in Ref [26]. All computations were on the same computing cluster [31] used in Ref [10].

III. EXPERIMENTS

In this section, we evaluate the speed of convergence of our algorithms as a function of M (the number of Monte Carlo samples drawn in a sampling round) and T (the number of learning steps per stage) under a fixed time budget. We begin with synthetic data, for which we know the correct answer, and then consider the problem of constructing maximum entropy models to describe real data.

Input: empirical observations $\sigma^1, \cdots, \sigma^m$
parameters M and T
access to maxent inference algorithm $\mathcal{A}$
Algorithm:
initialize $\mathcal{A}$
$\theta(1,1) \leftarrow$ initial parameter vector computed by $\mathcal{A}$
for $s = 1, 2, \ldots$
generate M Monte Carlo samples using $\theta(s,1)$
for $t = 1, \ldots, T$
run one iteration of $\mathcal{A}$, approximating $\langle \cdots \rangle_{\theta(s,t)} = \langle \cdots \rangle_{\theta(s,t)|\theta(s,1)}$
$\theta(s,t+1) \leftarrow$ parameter vector computed by $\mathcal{A}$ on this iteration
$\theta(s+1,1) \leftarrow \theta(s,T+1)$

FIG. 1: Pseudocode for our scheme, which reuses a Monte Carlo sample set across T iterations of a maxent algorithm $\mathcal{A}$.

A. Synthetic data

The maximum entropy construction is a strategy for simplifying our description of a system with many interacting elements. Separate from the algorithmic question of finding the maximum entropy model is the natural science question of whether this model provides a good description of the system we are studying, making successful predictions beyond the set of low order correlations that are used to construct the model. To sharpen our focus on the algorithmic problem, we use as "empirical data" a set of samples generated by Monte Carlo simulation of an Ising model as in Eq (1). To get as close as possible to the problems we face in analyzing real data, we use the parameters of the Ising model found in Ref [10] in the analysis of activity in a network of $N = 40$ neurons in the retina as it responds to naturalistic movies. We note that this model has competing interactions, as in a spin glass, with multiple locally stable states; extrapolation to larger systems suggests that the parameters are close to a critical point.

Starting with the parameters $\theta_0 \equiv \{h_i, J_{ij}\}$ determined in Ref [10], we generate $m = 2 \times 10^5$ samples by Monte Carlo simulation. Let us call the empirical distribution over these samples $\hat{P}$. Then we proceed to minimize $\mathrm{L}_{\hat{P}}(\theta)$. To monitor the progress of the algorithm, we compute

$$\Delta\mathrm{L}(\theta) = \mathrm{L}_{\hat{P}}(\theta) - \mathrm{L}_{\hat{P}}(\theta_0). \tag{20}$$

Note that this computation requires us to estimate the log ratios of partition functions, for which we again use the histogram Monte Carlo approach,

$$\frac{\mathbf{Z}(\theta)}{\mathbf{Z}(\theta_0)} = \langle \exp\left[-(\theta-\theta_0)\cdot\mathbf{f}(\sigma)\right] \rangle_{\theta_0} \tag{21}$$

$$\approx \left\langle \exp\left[-(\theta-\theta_0)\cdot\mathbf{f}(\sigma)\right] \right\rangle_{MC\theta_0}. \tag{22}$$

As a check on this approximation, we can exchange the roles of θ and θ_0,

$$\frac{\mathbf{Z}(\theta_0)}{\mathbf{Z}(\theta)} \approx \left\langle \exp\left[+(\theta-\theta_0)\cdot\mathbf{f}(\sigma)\right] \right\rangle_{MC\theta}, \tag{23}$$

and test for consistency between the two results. Finally, to compare the performance of the two optimization algorithms, we withhold some fraction of the data, here chosen to be 10%, for testing.

Figure 2 illustrates the performance of the two algorithms on the synthetic dataset, measured by ΔL. for simplicity, we have held the number of Monte Carlo samples at each stage, $M = 3.2 \times 10^5$, fixed. Choosing $T = 1$ iteration per stage corresponds to a naive use of Monte Carlo, with a new simulation for each new setting of the parameters, and we see that this approach converges very slowly, as found in Ref [10]. Simply increasing this number to $T = 20$ produces at least an order of magnitude speed up in convergence.

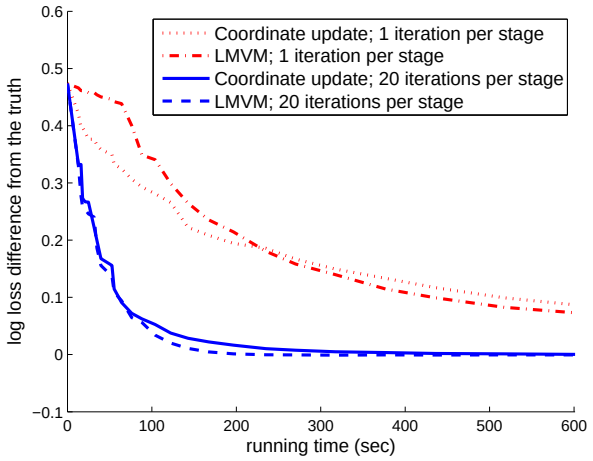

FIG. 2: Approximate difference in the test log loss between the current solution of an algorithm and the exact model that generates the test data, as a function of the algorithm running time. The number of Monte Carlo samples at each stage is set to 320,000. The two choices of optimization algorithm exhibit similar performance, but much greater efficiency is achieved in both cases by including multiple iterations of learning algorithm $\mathcal{A}$ per stage.

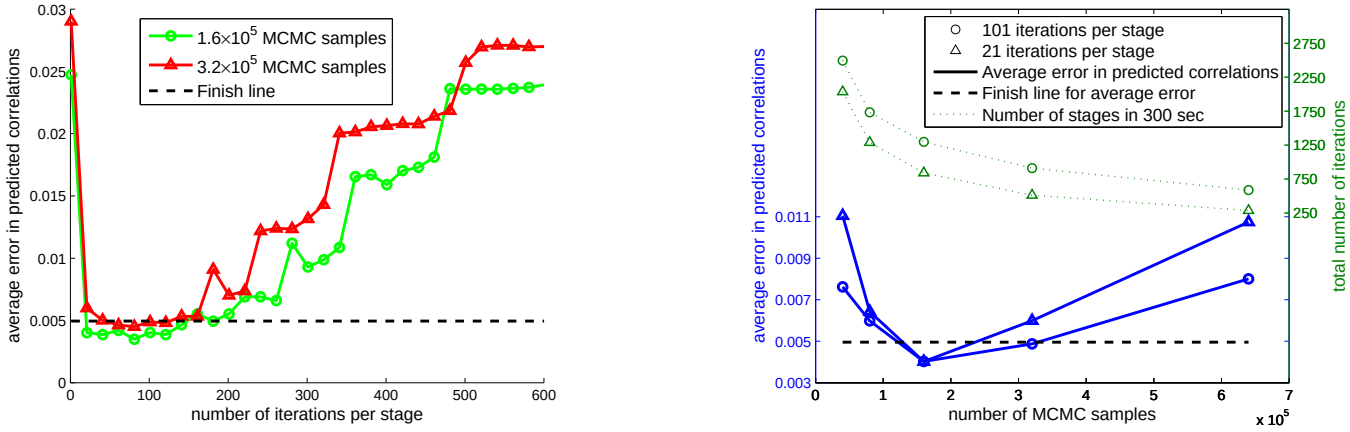

FIG. 3: (Left) Performance of our algorithm on the empirical data according to the absolute correlation difference metric, as a function of the number of iterations of the optimization algorithm and the number of Monte Carlo samples. The algorithm was terminated after the running time exceeded 300 seconds. Choosing too many or too few iterations per stage decreases efficiency (due to overlearning or inefficient use of samples, respectively). (Right) Performance of our algorithm on the empirical data according to the absolute correlation difference metric, as a function of the number of examples generated by the Monte Carlo sampler. The algorithm was terminated after the running time exceeded 300 seconds. Also displayed is the total number of iterations completed by the time cutoff. Fewer iterations per sample allow the generation of more samples, which is a computationally expensive process; therefore, fewer iterations result in fewer total optimization stages. Choosing too many or too few Monte Carlo samples per stage decreases efficiency, either because the algorithm samples for too long and terminates before enough learning stages can be completed, or because the sampling is too sparse and expectations are poorly estimated.

B. Real data

Here we consider the problem of constructing the maximum entropy distribution consistent with the correlations observed in real data. Our example is from Refs [9, 10], the responses of $N = 40$ retinal neurons. As noted at the outset, if we divide time into small slices of duration $\Delta\tau$ then each cell either does or does not generate an action potential, corresponding to a binary or Ising variable. The experiments of Ref [9], analyzed with $\Delta\tau = 0.02\,\mathrm{s}$, provide $m = 189,950$ samples of the state σ. We note that spikes are rare, and pairwise correlations are weak, as discussed at length in Ref [9].

As we try to find the parameters θ that describe these data, we need a metric to monitor the progress of our algorithm; of course we don't have access to the true distribution. Since we are searching in the space of Gibbs distributions which automatically have correct functional form to maximize the entropy, we check the quality of agreement between the predicted and observed expectation values. It is straightforward to insure that the model reproduces the one–body averages $\langle\sigma_i\rangle$ almost exactly. Thus we focus on the (connected) two–body averages $C_{ij} = \langle\sigma_i\sigma_j\rangle - \langle\sigma_i\rangle\langle\sigma_j\rangle$. We compute these averages via Monte Carlo in the distribution $Q_\theta(\sigma)$, and then form the absolute difference between computed and observed values of C_{ij}, and finally average over all pairs ij. The resulting average error ΔC measures how close we come to solving the maximum entropy problem. Note that since the observed values of C_{ij} are based on a finite set of samples, we don't expect that they are exact, and hence we shouldn't identify convergence with $\Delta C \to 0$. Instead we divide the real data in half and measure ΔC between these halves, and use this as the 'finish line' for our algorithms.

In Fig 3 we see the behavior of ΔC as a function of the number of iterations per stage (T), where we terminate the algorithm after 300 seconds of running time on our local cluster [31]. As expected intuitively, both small T and large T perform badly, but there is a wide range $T \sim 30{-}200$ in which the algorithm reaches the finish line within the alloted time. This performance also depends on M, the number of Monte Carlo samples per stage, so that again there is a range of $M \sim 1 - 3 \times 10^5$ that seems to work well. In effect, when we constrain the total run time of the algorithm there is a best way of apportioning this time between stages (in which we run new Monte Carlo simulations) and iterations (in which we adjust parameters using estimates based on a fixed set of Monte Carlo samples). By working within a factor of two of this optimum, we achieve substantial speed up of the algorithm, or an improvement in convergence at fixed run time.

IV. CONCLUSION

Our central conclusion is that recycling of Monte Carlo samples, as in the histogram Monte Carlo method [19], provides a substantial speedup in the solution of the in-

verse Ising problem. In detail, some degree of recycling speeds up the computations, while of course too much recycling renders the underlying approximations invalid, so that there is some optimal amount of recycling; fortunately it seems from Fig 3 that this optimum is quite broad. We expect that these basic results will be true more generally for problems in which we have to learn the parameters of probabilistic models to provide the best match to a large body of data. In the specific context of Ising models for networks of real neurons, the experimental state of the art is now providing data on populations of neurons which are sufficiently large that these issues of algorithmic efficiency become crucial.

Acknowledgments

We thank MJ Berry II, E Schneidman and R Segev for helpful discussions and for their contributions to the work which led to the formulation of the problems addressed here. This work was supported in part by NIH Grant P50 GM071508, and by NSF Grants IIS–0613435 and PHY–0650617.

[1] JB Keller & B Zumino, Determination of intermolecular potentials from thermodynamic data and the law of corresponding states. *J Chem Phys* **30,** 1351–1353 (1959).

[2] W Kunkin & HL Frisch, Inverse problem in classical statistical mechanics. *Phys Rev* **177,** 282–287 (1969).

[3] JT Chayes, L Chayes & E Lieb, The inverse problem in classical statistical mechanics. *Commun Math Phys* **93,** 57–121 (1984).

[4] E Caglioti, T Kuna, J Lebowitz & E Speer, Point processes with specified low order correlation. *J Markov Proc and Related Fields* **12,** 257–272 (2006).

[5] JS Yedidia, WT Freeman & Y Weiss, Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Trans Inf Theory* **51,** 2282–2312 (2005).

[6] JJ Hopfield, Neural networks and physical systems with emergent collective computational abilities. *Proc Nat'l Acad Sci (USA)* **79,** 2554–2558 (1982).

[7] DJ Amit, *Modeling Brain Function: The World of Attractor Neural Networks* (Cambridge University Press, Cambridge, 1989).

[8] GE Hinton & TJ Sejnowski, Learning and relearning in Boltzmann machines, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1,* ed. DE Rumelhart, JL McClelland & the PDP Research Group, pp. 282–317 (MIT Press, Cambridge, 1986).

[9] E Schneidman, MJ Berry II, R Segev & W Bialek, Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature* **440,** 1007–1012 (2006); q–bio.NC/0512013.

[10] G Tkačik, E Schneidman, MJ Berry II & W Bialek, Ising models for networks of real neurons. q–bio.NC/0611072 (2006).

[11] J Shlens et al, The structure of multi–neuron firing patterns in primate retina. *J Neurosci* **26,** 8254–8266 (2006).

[12] See, for example, the presentations at the 2007 meeting on Computational and Systems Neuroscience, `http://cosyne.org/wiki/Cosyne_07`: IE Ohiorhenuan & JD Victor, Maximum entropy modeling of multi–neuron firing patterns in V1. J Shlens et al, Spatial structure of large–scale synchrony in the primate retina. A Tang et al, A second–order maximum entropy model predicts correlated network states, but not their evolution over time.

[13] See also the presentations at the 2007 meeting of the Society for Neuroscience, `http://www.sfn.org/am2007/`: J Shlens et al, Spatial organization of large–scale concerted activity in primate retina, 176.17/JJ10. IE Ohiorhenuan & JD Victor, Maximum-entropy analysis of multi-neuron firing patterns in primate V1 reveals stimulus-contingent patterns, 615.8/O01. S Yu, D Huang, W Singer & D Nikolić, A small world of neural synchrony, 615.14/O07. MA Sacek, TJ Blanche, JK Seamans & NV Swindale, Accounting for network states in cortex: are pairwise correlations sufficient?, 790.1/J12. A Tang et al, A maximum entropy model applied to spatial and temporal correlations from cortical networks in vitro, 792.4/K27.

[14] TR Lezon et al, Using the principle of entropy maximization to infer genetic interaction networks from gene expression patterns. *Proc Nat'l Acad Sci (USA)* **103,** 19033–19038 (2006).

[15] G Tkačik, *Information Flow in Biological Networks* (Dissertation, Princeton University, 2007).

[16] AM Litke et al, What does the eye tell the brain? Development of a system for the large scale recording of retinal output activity. *IEEE Trans Nucl Sci* **51,** 1434–1440 (2004).

[17] R Segev, J Goodhouse, JL Puchalla & MJ Berry II, Recording spikes from a large fraction of the ganglion cells in a retinal patch. *Nat Neurosci* **7,** 1155–1162 (2004).

[18] For a general discussion of Monte Carlo methods in machine learning, see A Andrieu, N de Freitas, A Doucet & MI Jordan, An introduction to MCMC for machine learning, *Machine Learning* **50,** 5–43 (2003); RM Neal Probabilistic inference using Markov Chain Monte Carlo Methods, Technical report CRG–TR–93–1, Dep't of Computer Science, University of Toronto (1993).

[19] AM Ferrenberg & RH Swendsen, New Monte Carlo technique for studying phase transitions. *Phys Rev Lett* **61,** 2635–2638 (1988).

[20] M Dudík, SJ Phillips & RE Schapire, Performance guarantees for regularized maximum entropy density estimation, in *Learning Theory, 17th Annual Conference on Learning Theory*, J Shawe–Taylor & Y Singer, eds, pp 472–486 (Springer–Verlag, Berlin, 2004).

[21] ET Jaynes, Information theory and statistical mechanics. *Phys Rev* **106,** 62–79 (1957).

[22] E Schneidman, S Still, MJ Berry II & W Bialek, Network information and connected correlations. *Phys Rev Lett* **91,** 238701 (2003); physics/0307072.

[23] TM Cover & JA Thomas, *Elements of Information Theory* (John Wiley & Sons, New York, 1991).

[24] K Binder, *Monte Carlo Methods in Statistical Physics* (Springer–Verlag, Berlin, 1979).

[25] MEJ Newman & GT Barkema, *Monte Carlo Methods in Statistical Physics* (Oxford University Press, Oxford, 1999).

[26] S Geman & D Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans PAMI* **6,** 721–741 (1984).

[27] M Collins, RE Schapire & Y Singer, Logistic regression, AdaBoost and Bregman distances. *Machine Learning* **48,** 253–285 (2002).

[28] In early stages of our work we encountered some difficulties with convergence of the coordinate descent algorithm. One possible source of the problem is that if, for example, the Monte Carlo sample includes no spikes from a particular neuron, then trying to match the observed expectation values can become impossible. To ease this difficulty we sometimes bias expectation values over the Monte Carlo samples with a small contribution from expectation values taken from the real data. Further studies would be required to determine if this is essential once we have the optimal values for the parameters M and T.

[29] SJ Benson, L Curfman McInnes, J Moré & J Sarich, `TAO` user manual (revision 1.8). Technical Report ANL/MCS–TM–242, Mathematics and Computer Science Division, Argonne National Laboratory (2005); `http://www.mcs.anl.gov/tao`.

[30] R Malouf, A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Conference on Natural Language Learning*, D Roth & A van den Bosch, eds, pp 49–55 (Morgan Kaufman, 2002).

[31] The facility has 90 2.3 GHz Apple G5 and 16 1.3 GHz G4 CPUs, running under Sun's Grid Engine. See the description of the fafner grid at `http://genomics.princeton.edu/support/grids/`.