Convenient Interface to Inverse Ising (ConIII): A Python 3 package for solving maximum entropy models

Edward D Lee,[1] Bryan C Daniels[2]
[1]*Department of Physics, 142 Sciences Dr, Cornell University, Ithaca NY 14853,*
[2]*ASU–SFI Center for Biosocial Complex Systems, Arizona State University, Tempe, AZ 85287*

ConIII (pronounced CON-ee) is an open-source Python project providing a simple interface to solving maximum entropy models, with a focus on the Ising model. We describe the maximum entropy problem and give an overview of the algorithms that are implemented as part of ConIII (`https://github.com/eltrompetero/coniii`) including Monte Carlo histogram, pseudolikelihood, minimum probability flow, a regularized mean field method, and a cluster expansion method. We briefly explain how one should approach and validate maxent models from the perspective of model selection. Our goal is to make a variety of maximum entropy techniques accessible to those unfamiliar with the techniques and accelerate workflow for users.

INTRODUCTION

Many biological and social systems are characterized by collective behavior: the correlated pattern of neural firing [25], protein diversity in the immune system [21], conflict participation in monkeys [11], flocking in birds [4], statistics of letters in words [29], or consensus voting in the US Supreme Court [18**?**]. Statistical physics is a natural approach to probing such systems precisely because they are collective [7]. Recently, the development of numerical, analytic, and computational tools have made it feasible in these large collective systems to solve for the maximum entropy (maxent) model that reproduces system behavior, corresponding to solving an "inverse problem." This approach contrasts with the typical problem in statistical physics where one postulates the microscopic model (the Hamiltonian) and works out the physical behavior of the system. In the inverse problem, we find the parameters that correspond to observed behavior of a known system. In many cases, this is a very difficult problem to solve and does not have an analytical solution, and we must rely on analytic approximation and numerical techniques to estimate the parameters.

The pairwise maxent model, the Ising model, has been of particular interest because of its simplicity and generality. A variety of algorithms have been proposed to solve the inverse Ising problem, but different approaches are disparately available on separate code bases in different coding languages, which makes comparison difficult and pedagogy more complicated. With ConIII, it is possible to solve the inverse Ising problem with a variety of algorithms in just a few lines of code.

ConIII is intended to provide a centralized resource for the inverse Ising problem and easy extension to other maxent problems. Although some of the implemented algorithms are specific to the Ising model and convenient, Ising-specific functions are included as part of package, other maxent models can be solved as well by specifying the particular constraints of the maxent model of interest.

In the first few sections of this paper, we give a brief overview of maxent and describe the algorithms implemented in this package. Then, we discuss at a high level some of the model fitting issues that are relevant for building and fitting maxent models. Finally, we describe the architecture of the package and how to contribute.

WHAT IS MAXIMUM ENTROPY?

Shannon introduced the concept of information entropy in his seminal paper about communication over a noisy channel [26]. Information entropy is the unique measure of uncertainty that follows from insisting on elementary principles of consistency. According to Shannon, the entropy over the probability distribution $p(\mathrm{s})$ of possible discrete configurations $\mathcal{S}$ of a system is

$$S[p] = -\sum_{\mathrm{s}\in\mathcal{S}} p(\mathrm{s}) \log p(\mathrm{s}). \quad (1)$$

These configurations could be on-off patterns of firing in neurons, the arrangement of letters in a word, or the orientation of spins in a material.

When there is no structure in the distribution, meaning that the probability is uniform, entropy is at a maximum. In the context of communication theory as Shannon first discussed, this means that there is no structure to exploit to make a prediction about the next part of an incoming message; thus, maximum entropy means that each new part of the message is maximally "surprising." At the other extreme, when the message consists of the same bit over and over again, we can always guess at the following part of the message and the signal has zero entropy. In the context of modeling, we use entropy not to refer to the difficulty of the message, but to our state of knowledge about it. Entropy precisely measures our uncertainty about the configuration in which we expect to find the system.

Maximum entropy, or maxent, is the formal framework for building models that are consistent with statistics from the data but otherwise as structureless as possible [15, 16]. We begin by choosing a set of K useful or important features from the data $f_{\mathrm{k}}(\mathrm{s})$ that should be true for the model that we are trying to build. These could be the averaging firing probabilities of a set of neurons or the average pairwise coocurrence for primates in a sequence of conflicts. The average of this feature across set of states in the data set $\mathcal{D}$ with R elements is

$$\langle f_{\mathrm{k}}\rangle_{\mathrm{data}} = \frac{1}{R}\sum_{\mathrm{s}\in\mathcal{D}} f_{\mathrm{k}}(\mathrm{s}). \tag{2}$$

According to the model in which each observation s occurs with some probability $p(\mathrm{s})$, the same average is calculated over all possible states

$$\langle f_{\mathrm{k}}\rangle = \sum_{\mathrm{s}\in\mathcal{S}} p(\mathrm{s}) f_{\mathrm{k}}(\mathrm{s}). \tag{3}$$

We assert that the model should fit the K features while maximizing entropy. The standard procedure is to solve this by the method of Langrangian multipliers. We construct the Langrangian functional $\mathcal{L}$ by introducing the multipliers λ_{k}.

$$\mathcal{L}[p] = -\sum_{\mathrm{s}\in\mathcal{S}} p(\mathrm{s})\log p(\mathrm{s}) - \sum_{\mathrm{k}=1}^{K}\lambda_{\mathrm{k}}\big(\langle f_{\mathrm{k}}\rangle - \langle f_{\mathrm{k}}\rangle_{\mathrm{data}}\big) \tag{4}$$

Then, we solve for the fixed point by taking the derivative with respect to λ_{k}. The resulting maxent model is a Boltzmann distribution over states:

$$p(\mathrm{s}) = e^{-E(\mathrm{s})}\Big/ Z, \tag{5}$$

with relative negative log-likelihood (also known as the energy or Hamiltonian)

$$E(\mathrm{s}) = -\sum_{\mathrm{k}=1}^{K}\lambda_{\mathrm{k}} f_{\mathrm{k}}(\mathrm{s}), \tag{6}$$

and normalization factor (also known as the partition function)

$$Z = \sum_{\mathrm{s}\in\mathcal{S}} e^{-E(\mathrm{s})}. \tag{7}$$

Entropy is a convex function of p and the constraints are linear with respect to p, so the problem is convex and the maxent distribution unique. Readers familiar with statistical physics will recognize this as an alternative derivation of the microcanonical ensemble, demonstrating that statistical mechanics can be viewed as an inference procedure using the maxent principle [15].

Finding the parameters λ_{k} that match the constraints $\langle f_{\mathrm{k}}\rangle_{\mathrm{data}}$ is equivalent to minimizing the Kullback-Leibler divergence between the model and the data [10]

$$D_{\mathrm{KL}}(p_{\mathrm{data}}||p) = \sum_{\mathrm{s}} p_{\mathrm{data}}\log\left(\frac{p_{\mathrm{data}}(\mathrm{s})}{p(\mathrm{s})}\right) \tag{8}$$

$$\frac{\partial D_{\mathrm{KL}}}{\partial\lambda_{\mathrm{k}}} = \sum_{\mathrm{s}} p_{\mathrm{data}}(\mathrm{s})\frac{\partial(-E(\mathrm{s}) - \log Z)}{\partial\lambda_{\mathrm{k}}} = 0$$

$$\implies \langle f_{\mathrm{k}}\rangle_{\mathrm{data}} = \langle f_{\mathrm{k}}\rangle\,. \tag{9}$$

In other words, the parameters of the maxent model are the ones that minimize the information theoretic “distance” to the distribution of the data given the constraints. Note that these parameters are given by the data: once the constraints have been chosen, there is a single maxent solution, with no free parameters.

The Ising model

The Ising model is a statistical physics model of magnetism [14]. It consists of a set of spins $\mathrm{s_i}$ with 2 possible orientations (up and down), each coupled to an external magnetic field h_{i} and coupled to each other with pairwise couplings J_{ij}. The strength of the magnetic field determines the tendency of each of the spins to orient in a particular direction and the couplings determine whether the spins tend to point together ($J_{\mathrm{ij}} > 0$) or against each other ($J_{\mathrm{ij}} < 0$). Typically, neighbors are defined as spins that interact with one another given by some underlying network structure. Figure 1 shows a fully-connected example.

The energy of each configuration determines its probability via (5):

$$E(\mathrm{s}) = -\sum_{\mathrm{i<j}}^{N} J_{\mathrm{ij}}\mathrm{s_i s_j} - \sum_{\mathrm{i}=1}^{N} h_{\mathrm{i}}\mathrm{s_i}. \tag{10}$$

We can derive the Ising model from the perspective of maxent. Fixing the the means and pairwise correlations to those observed in the data

$$\langle \mathrm{s_i}\rangle = \langle \mathrm{s_i}\rangle_{\mathrm{data}} \tag{11}$$

$$\langle \mathrm{s_i s_j}\rangle = \langle \mathrm{s_i s_j}\rangle_{\mathrm{data}} \tag{12}$$

we go through the procedure of constructing the Langrangian from Eq 4

$$\mathcal{L}[p] = -\sum_{\mathrm{s}} p(\mathrm{s})\log p(\mathrm{s}) + \sum_{\mathrm{i<j}}^{N} J_{\mathrm{ij}}\langle \mathrm{s_i s_j}\rangle + \sum_{\mathrm{i}}^{N} h_{\mathrm{i}}\langle \mathrm{s_i}\rangle \tag{13}$$

$$\frac{\partial\mathcal{L}[p]}{\partial p(\mathrm{s})} = -\log p(\mathrm{s}) - 1 + \sum_{\mathrm{i<j}}^{N} J_{\mathrm{ij}}\mathrm{s_i s_j} + \sum_{\mathrm{i}}^{N} h_{\mathrm{i}}\mathrm{s_i} \tag{14}$$

$$\log p(\mathrm{s}) = -1 + \sum_{\mathrm{i<j}}^{N} J_{\mathrm{ij}}\mathrm{s_i s_j} + \sum_{\mathrm{i}}^{N} h_{\mathrm{i}}\mathrm{s_i} \tag{15}$$

$$p(\mathrm{s}) = e^{-E(\mathrm{s})}\Big/ Z \tag{16}$$

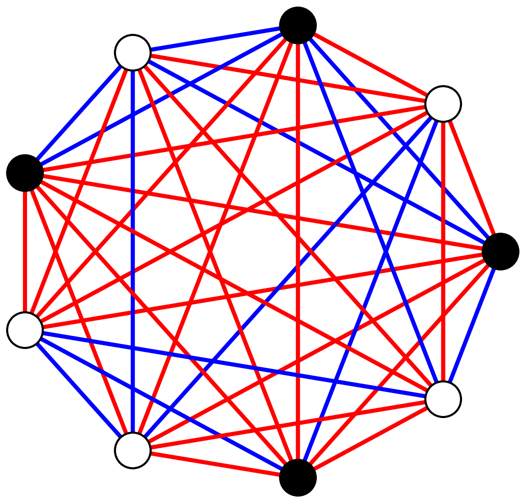

FIG. 1. Example of a fully connected Ising model with random couplings. Each spin s_i (circle) can take one of two states (black or white, corresponding to -1 and $+1$) and is connected to every other spin in the system with a positive or negative coupling. These states could describe the on-off patterns of firing in neurons, the arrangement of letters in a word, or the orientation of spins in a material. This schematic represents one possible state of the spins. The inverse Ising problem starts with statistics about the states observed in the system and infers the coupling strengths J_{ij} and individual biases h_i that reproduce such statistics.

where the -1 in Eq 15 has been absorbed into the normalization factor

$$Z = \sum_{s} e^{-E(s)}. \tag{17}$$

such that the probability distribution is normalized $\sum_s p(s) = 1$. Thus, the resulting model is exactly the Ising model mentioned earlier.

Despite the simplicity of the Ising model, the structure imposed by the discrete nature of the spins means that finding the parameters is challenging analytically and computationally. In the last few years, numerous techniques have been suggested for solving the inverse Ising problem exactly or approximately [24]. We have implemented some of them in ConIII and designed a package structure to make it easily extensible to include more methods. Here, we briefly describe the algorithms that are part of the first official version of the package. The goal is to give the user a sense (or reminder) of how they work without getting bogged down in heavy detail. For more detail, we suggest perusing the papers referenced in each section or the review [24]. For a complete beginner, it may be useful to first get familiar with a slower introduction like in the Appendices of Ref [18], Ref [5], or Ref [16].

INVERSE ISING METHODS IMPLEMENTED IN CONIII

Enumeration

The naïve approach that only works for small systems is to write out the equations from Eq 9 and solve them numerically. After writing out all K equations,

$$\langle f_k \rangle = -\frac{\partial \ln Z}{\partial \lambda_k} = \langle f_k \rangle_{\text{data}}\,, \tag{18}$$

we can use any standard root-finding algorithm to find the parameters λ_k. This approach, however, involves enumerating all states of the system, whose number grows exponentially with system size.

For the Ising model, writing down the equations has a number of steps $\mathcal{O}(K^2 2^N)$, where K is the number of constraints and N the number of spins. Each evaluation of the objective in the root-finding algorithm will be of the same order. For relatively small systems, around $N \leq 15$, this approach is feasible on a typical desktop computer and is a good way to test the results of a more complicated algorithm.

This approach is implemented by the `Enumerate` class.

Monte Carlo method

Perhaps the most straightforward and most expensive computational approach is Monte Carlo Markov Chain (MCMC) sampling. A series of states sampled from a proposed $p(s)$ is produced by MCMC to approximate $\langle f_k \rangle$ and determine how close we are to matching $\langle f_k \rangle_{\text{data}}$. The parameters are then adjusted using a learning rule, and both sampling and learning are repeated until a stopping criterion is met. This can be combined with a variety of approximate gradient descent methods to reduce the number of sampling steps by predicting how the distribution will change if we modify the parameters slightly. The particular technique implemented in ConIII is the Monte Carlo Histogram (MCH) method [6].

Since the sampling step is expensive, the idea behind MCH is to reuse a sample for more than one gradient descent step [6]. Given that we have a sample with probability distribution $p(s)$ generated with parameters λ_k, we would like to estimate the proposed distribution $p'(s)$ from adjusting our parameters $\lambda'_k = \lambda_k + \Delta\lambda_k$. We can leverage our current sample to make this extrapolation.

$$p' = \frac{p'}{p} p \tag{19}$$

$$p'(s) = \frac{Z}{Z'} e^{\sum_k \Delta\lambda_k f_k(s)} p(s) \tag{20}$$

To estimate the average,

$$\sum_{\mathrm{s}} p'(\mathrm{s}) f_{\mathrm{k}}(\mathrm{s}) = \frac{Z}{Z'} \sum_{\mathrm{s}} p(\mathrm{s}) e^{\sum_{\mathrm{k}} \Delta\lambda_{\mathrm{k}} f_{\mathrm{k}}(\mathrm{s})} f_{\mathrm{k}}(\mathrm{s}) \tag{21}$$

To be explicit about the fact that we only have a sampled approximation to p, we replace p with the sample distribution.

$$\langle f_{\mathrm{k}} \rangle' = \frac{Z}{Z'} \left\langle e^{\sum_{\mathrm{k}} \Delta\lambda_{\mathrm{k}} f_{\mathrm{k}}(\mathrm{s})} f_{\mathrm{k}}(\mathrm{s}) \right\rangle_{\mathrm{sample}} \tag{22}$$

Likewise, the ratio of the partition function can be estimated

$$\frac{Z}{Z'} \approx 1 \Big/ \left\langle e^{\sum_{\mathrm{k}} \Delta\lambda_{\mathrm{k}} f_{\mathrm{k}}(\mathrm{s})} \right\rangle_{\mathrm{sample}} \tag{23}$$

At each step, we update the Lagrangian multipliers $\{\lambda_{\mathrm{k}}\}$ while being careful to stay within the bounds of a reasonable extrapolation. One suggestion is to update the parameters with some inertia [30]

$$\Delta\lambda_{\mathrm{k}}(t+1) = \Delta\lambda_{\mathrm{k}}(t) + \epsilon \Delta\lambda_{\mathrm{k}}(t-1) \tag{24}$$

$$\Delta\lambda_{\mathrm{k}}(t) = \eta \left(\langle f_{\mathrm{k}} \rangle' - \langle f_{\mathrm{k}} \rangle \right) \tag{25}$$

This has a fixed point at the correct parameters.

In practice, MCH can be difficult to tune properly and one must check in on the progress of the algorithm often. One issue is choosing how to set the learning rule parameters η and ϵ. One suggestion for η is to shrink it as the inverse of the number of iterations [30]. Another issue is that parameters cannot be changed by too much when using the MCH approximation step or the extrapolation to λ'_{k} will be inaccurate and the algorithm will fail to converge. In ConIII, this can be controlled by setting a bound on the maximum possible change in each parameter $\Delta\lambda_{\mathrm{max}}$ and restricting the norm of the vector of change in parameters $\sum_{\mathrm{k}} \sqrt{\Delta\lambda_{\mathrm{k}}^2}$. Another issue is setting the parameters of the MCMC sampling routine. Both the burn time (the number of iterations before starting to sample) and sampling iterations (number of iterations between samples) must be large enough that we are sampling from the equilibrium distribution. Typically, these are found by measuring how long the energy or individual parameter values remain correlated as MCMC progresses. The parameters may need to be updated during the course of MCH because the sampling parameters may need to change with the estimated parameters of the model. For some regimes of parameter space, samples are correlated over long times and alternative sampling methods like Wolff or Swendsen-Wang should be used. We do not discuss these sampling details here, but see Refs [19, 23] for examples.

The main computation cost for MCH lies in the sampling step. For each iteration of MCH, the runtime is proportional to the number of samples n, number of MCMC iterations T, and the number of constraints for the Ising model N^2, $\mathcal{O}(TnN^2)$, whereas the MCH estimate is relatively quick $\mathcal{O}(tnN^2)$ because the number of MCH approximation steps needed to converge is much smaller than the number of MCMC sampling iterations $t << T$.

MCH is implemented in the `MCH` class.

Pseudolikelihood

The pseudolikelihood approach is an analytic approximation to the likelihood that drastically reduces the computational complexity of the problem and is exact as $N \to \infty$ [1]. We calculate the conditional probability of each spin $\mathrm{s_i}$ given the rest of the system $\{\mathrm{s_{j\neq i}}\}$

$$p\left(\mathrm{s_i} | \{\mathrm{s_{j\neq i}}\}\right) = \left(1 + e^{-2\mathrm{s_i}\left(h_{\mathrm{i}} + \sum_{j\neq i} J_{\mathrm{ij}} \mathrm{s_j}\right)}\right)^{-1} \tag{26}$$

Taking the logarithm, we define the approximate log-likelihood by summing over data points indexed by r:

$$f(h_{\mathrm{i}}, \{J_{\mathrm{ij}}\}) = \sum_{\mathrm{r}=1}^{R} \ln p\left(\mathrm{s_i^{(r)}} \middle| \{\mathrm{s_{j\neq i}}\}^{(\mathrm{r})}\right). \tag{27}$$

In the limit where the ensemble is well sampled, the average over the data can be replaced by an average over the ensemble:

$$f(h_{\mathrm{i}}, \{J_{\mathrm{ij}}\}) = \sum_{\mathrm{s}} \ln p\left(\mathrm{s_i} | \{\mathrm{s_{j\neq i}}\}\right) p\left(\mathrm{s}; h_{\mathrm{i}}, \{J_{\mathrm{ij}}\}\right). \tag{28}$$

To find the point of maximum likelihood for a single spin $\mathrm{s_i}$, we calculate the analytical gradient and Hessian, $\partial f/\partial J_{\mathrm{ij}}$ and $\partial^2 f/\partial J_{\mathrm{ij}} \partial J_{\mathrm{i'j'}}$ for a Newton conjugate-gradient descent method. After maximizing likelihood for all spins, the maximum likelihood parameters may not satisfy the symmetry $J_{\mathrm{ij}} = J_{\mathrm{ji}}$. We impose the symmetry by insisting that

$$J'_{\mathrm{ij}} = (J_{\mathrm{ij}} + J_{\mathrm{ji}})/2. \tag{29}$$

Pseudolikelihood is extremely fast and often surprisingly accurate. Each calculation of the gradient is order $\mathcal{O}(RN^2)$ and Hessian $\mathcal{O}(RN^3)$, which must be done for all N. With analytic forms for the gradient and Hessian, the conjugate-gradient descent method tends to converge quickly.

Pseudolikelihood for the Ising model is implemented in `Pseudo`.

Minimum Probability Flow

Minimum probability flow involves analytically approximating how the probability distribution *changes* as we modify the *configurations* [27, 28]. In the methods so far mentioned, the approach has been to maximize the objective (the likelihood function) by immediately taking the derivative with respect to the parameters. With MPF, we first posit a set of dynamics that will lead the data distribution to equilibrate to that of the model. When these distributions are equivalent, then there is no "probability flow" between them. This technique is closely related to score matching, where we instead have a continuous state space and can directly take the derivative with respect to the states without specifying dynamics [13].

First note that Monte Carlo dynamics (satisfying ergodicity and detailed balance) would lead to equilibration to the stationary distribution. One such transition matrix suggested in Ref [28] is

$$\dot{p}_{\mathrm{s}} = \sum_{\mathrm{s}'\neq \mathrm{s}} \Gamma_{\mathrm{ss}'} p_{\mathrm{s}'} - \sum_{\mathrm{s}'\neq \mathrm{s}} \Gamma_{\mathrm{s}'\mathrm{s}} p_{\mathrm{s}} \tag{30}$$

$$\Gamma_{\mathrm{ss}'} = g_{\mathrm{ss}'} \exp\left[\frac{1}{2}\left(E_{\mathrm{s}'} - E_{\mathrm{s}}\right)\right] \tag{31}$$

with transition probabilities $\Gamma_{\mathrm{ss}'}$ from state s$'$ to state s. The connectivity matrix $g_{\mathrm{ss}'}$ specifies whether there is edge between states s and s$'$ such that probability can flow between them. By choosing a sparse $g_{\mathrm{ss}'}$ while not breaking ergodicity, we can drastically reduce the computational cost of computing this matrix.

Imagine that we start with the distribution over the states as given by the data and run the Monte Carlo dynamics. When data and model distributions are different, probability will flow between them and indicate that the parameters must be changed. By minimizing a derivative of the Kullback-Leibler divergence, we measure how the difference between the model and the states in the data $\mathcal{D}$ changes when the dynamics are run for an infinitesimal amount of time.

$$L(\{\lambda_{\mathrm{k}}\}) \equiv \partial_t D_{\mathrm{KL}}(p^{(0)}||p^{(t)}(\{\lambda_{\mathrm{k}}\})) = \sum_{\mathrm{s}\notin\mathcal{D}} \dot{p}_{\mathrm{s}}(\lambda_{\mathrm{k}}) \tag{32}$$

The idea is that this derivative is also minimized with optimal parameters: the MPF algorithm looks for a minimum of the objective function L.

For the Ising model, each evaluation of the objective function where $\Gamma_{\mathrm{ss}'}$ connects each data state with G neighbors has runtime $\mathcal{O}(RGN^2)$. In a large fully connected system, $G \sim 2^N$ would be prohibitively large so a sparse choice is necessary.

MPF is implemented in the `MPF` class.

Regularized mean-field method

One attractively simple and efficient approach uses a regularized version of mean-field theory. In the inverse Ising problem, mean-field theory is equivalent to treating each binary individual as instead having a continuously varying state (corresponding to its mean value). The inverse problem then turns into simply inverting the correlation matrix C [9]:

$$J_{\mathrm{ij}}^{\mathrm{mean-field}} = -\frac{(C^{-1})_{\mathrm{ij}}}{\sqrt{p_{\mathrm{i}}(1-p_{\mathrm{i}})p_{\mathrm{j}}(1-p_{\mathrm{j}})}}, \tag{33}$$

where

$$C_{\mathrm{ij}} = \frac{p_{\mathrm{ij}} - p_{\mathrm{i}}p_{\mathrm{j}}}{\sqrt{p_{\mathrm{i}}(1-p_{\mathrm{i}})p_{\mathrm{j}}(1-p_{\mathrm{j}})}}, \tag{34}$$

and where p_{i} corresponds to the frequency of individual i being in the active (+1) state and p_{ij} is the frequency of the pair i and j being simultanously in the active state.

A simple regularization scheme in this case is to discourage large values in the interaction matrix J_{ij}. This corresponds to putting more weight on solutions that are closer to the case with no interactions (independent individuals). A particularly convenient form adds the following term, quadratic in J_{ij}, to the negative log-likelihood:

$$\gamma \sum_{\mathrm{i}} \sum_{\mathrm{i}<\mathrm{j}} J_{\mathrm{ij}}^2 p_{\mathrm{i}}(1-p_{\mathrm{i}})p_{\mathrm{j}}(1-p_{\mathrm{j}}). \tag{35}$$

In this case, the regularized version of the mean-field solution in (33) can be solved analytically, with the slowest computational step coming from the inversion of the correlation matrix. For details, see Refs. [2, 11].

The idea is then to vary the regularization strength γ to move between the non-interacting case ($\gamma \to \infty$) and the naively calculated mean-field solution (33) ($\gamma \to 0$). While there is no guarantee that varying this one parameter will produce solutions that are good enough to "fit within error bars," this approach has been successful in at least one case of fitting social interactions [11].

The inversion of the correlation matrix is relatively fast, bounded by $\mathcal{O}(N^3)$. Finding the optimal γ, involves Monte Carlo sampling from the model distribution, which has computational cost similar to MCH. It is, however, much more efficient because we are only optimizing a single parameter.

This is implemented in `RegularizedMeanField`.

Cluster expansion

Adaptive cluster expansion [2, 8, 9] iteratively calculates terms in the cluster expansion of the entropy S:

$$S - S_0 = \sum_{\Gamma} \Delta S_{\Gamma}, \tag{36}$$

where the sum is over clusters Γ and in the exact case includes all $2^N - 1$ possible nonempty subsets of individuals in the system. In the simplest version of the expansion, one expands around $S_0 = 0$. In some cases it can be more advantageous to expand around the independent individual solution or one of the mean-field solutions described in the previous section [2].

The inverse Ising problem is solved independently on each of the clusters, which can be done exactly when the clusters are small. These results are used to construct a full interaction matrix J_{ij}. The expansion starts with small clusters and expands to use larger clusters, neglecting any clusters whose contribution ΔS_Γ to the entropy falls below a threshold. To find the best solution that does not overfit, the threshold is initially set at a large value and then lowered, gradually including more clusters in the expansion, until samples from the resulting J_{ij} fit the desired statistics of the data sufficiently well.

The runtime will depend on the size of clusters included in the expansion. If the expansion is truncated at clusters of size n, the worst-case runtime would be $\mathcal{O}\left(\binom{N}{n}2^n\right)$. The point is that S can often be accurately estimated even when $n \ll N$.

The adaptive cluster expansion method is implemented in the `ClusterExpansion` class.

THE FINITE SAMPLE PROBLEM

A fundamental problem in model inference is that uncertainty coming from the finiteness of data translates into uncertainties in parameters. In the exposition of the maxent formulation derived from Eq 4, there is no ambiguity in the model parameters because they are fully specified by the constraints calculated from data. In reality, these constraints are typically estimated from a finite sample, and they are noisy. The straightforward answer to this problem is to take more data—in a pairwise maxent problem, we might insist that we have enough samples to well-constrain the correlation between every pair of individuals. But it is not always possible to take enough data. For instance, in a social system in which we are trying to measure stable social structure that lasts on the order of months, there are only a finite number of social interactions that occur over those months, which may not be enough to tightly constrain parameters. When we fit exactly to constraints measured from a small amount of data, we run the danger of overfitting and poor generalization.

When we find the parameters that minimize the Kullback-Leilber divergence between the model and the data distributions, we are maximizing the likelihood of the data. Sample size influences curvature of the likelihood function around the maximum-likelihood peak. Specifically, small sample size leads to a smaller curvature, implying a broad peak and larger uncertainty in parameters. We can estimate this uncertainty from the data: Assuming that the data is independently and identically distributed, the errors are given by the standard error of the mean of a binomial distribution $p_{\mathrm{ij\ldots k}} = p(s_{\mathrm{i}} = s_{\mathrm{j}} = \ldots = s_{\mathrm{k}} = 1)$ with R data points.

$$\delta_{\mathrm{ij\ldots k}} = \sqrt{(1-p_{\mathrm{ij\ldots k}})p_{\mathrm{ij\ldots k}}/R}. \tag{37}$$

In algorithms that iteratively bring the model closer to the data, we can stop once we have gotten close enough, where close enough is given by the expected fluctuations in the data distribution.

An alternative approach is to regularize the problem as in the regularized mean field or cluster expansion algorithms. Here, we restrict the search space in some principled way so that more complicated solutions are disallowed. We then check that the regularized solutions fit the data within expected statistical fluctuations. If not, a more lax regularization can be used to allow more complicated solutions that are able to fit the remaining signal in the data. In a Bayesian formulation, this approach is equivalent to including a prior distribution over parameters that more heavily weights simpler models.

MODEL VALIDATION

Besides just fitting the model while accounting for the noisiness of the data, how do we know if we need a more complicated model? To answer this question, we need to quantify how well the model fits and the corresponding tradeoff between the complexity and fit of the model.

One way to formalize this is to quantify the tradeoff between a good description of the data with the cost of describing the model. In the Bayesian formulation, we would write this as maximizing the posterior probability of the data given the parameters θ,

$$\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}|\theta) + \log p(\theta) + \mathrm{const} \tag{38}$$

and the tradeoff manifests as a competition between the likelihood $p(\mathcal{D}|\theta)$ and prior $p(\theta)$. For example, in the regularized mean-field method, the quadratic prior on the couplings favors sparse solutions over more complex solutions that might capture the data better but fail to generalize. This picture is formalized by quantities like the Akaike information criterion (AIC) or the Bayesian information criterion (BIC) [17].

Although likelihood tells us how well the models are doing relative to one another, it does not tell us in detail how the model is fitting the data. Basic checks would be to compare against other correlations in the data. This could include higher-order correlations than those used to fit the model or other features that are relevant to the question at hand. For instance, an important coarse-grained feature in macaque conflict is the distribution of fight sizes, which is well reproduced by fitting only pairwise correlations [11]. The most strict check would be to compare the entire model probability distribution with that of the data [18], but this is only feasible when the data set is reasonably large.

A summary of how well the model captures the distribution across the entire probability distribution is the multi-information [22, 25]. If we take a maxent model and add further constraints, the models can be ordered in terms of entropy $S_1 > S_2 > ...S_{\rm m} > ... > S_{\rm data}$, where the minimum entropy the most constrained model could have is equal to the data. To measure how much correlation in the data our model has captured, we can calculate the multi-information $I_{\rm m} = S_1 - S_{\rm m}$, the amount of correlation captured by the model relative to the independent model. The fraction of multi-information captured is $F = I_{\rm m}/I_{\rm data}$. This is a measure of how much of the correlation in the data is captured by the model.[1]

These tools to quantify model fit then inform the important choice of which constraints to impose. Typically, the approach is to constrain the lowest order interactions that are sufficient to produce collective behavior. The intuition from physics is from the observation that many physical systems are very well described by pairwise interactions [3]. From the model fitting perspective, however, we might choose to constrain other parts of the probability distribution. Ref [12] explores choosing correlations to constrain depending on whether they are signicantly large. Ref [20] discusses how a pairwise model becomes an increasingly effective description of a system with higher order interactions as the system gets larger.

SOFTWARE ARCHITECTURE

The package is divided into three principal modules containing the algorithms for solving the inverse maxent problem (`solvers.py`), the Monte Carlo Markov Chain (MCMC) sampling algorithms (`samplers.py`), and supporting "utility" functions for the remaining modules (`utils.py`) as shown in Fig. 2. Besides the `utils.py` module, the package is organized around classes that correspond to different algorithms. This class-based structure ensures that the state of the solver or sampler, including the data it was fit to and the current guess for the parameters, are all contained within the instance of the algorithm class. As a result, the current state of work can be saved and moved between workstations using the Python package `dill`.

For the solvers, the different algorithms available are accessible from the `coniii.solvers` module as listed in Fig. 2. These algorithm classes are all derived from a base `Solver` class as shown in Fig. 2. In the accompanying release version of the package, v1.0.0, the method `Solver.solve` serves as the interface for solving the inverse maxent problem. Once a solution has been found,

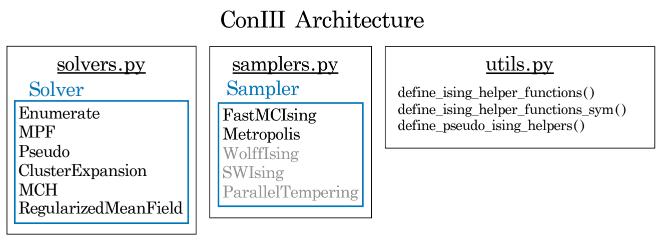

FIG. 2. ConIII architecture. The principal modules are `solvers.py`, `samplers.py`, and `utils.py`. The module `solvers.py` contains classes based on a the super class `Solver` that each implement a different algorithm for solving the relevant inverse maxent problem accessible through the method `Solver.solve()`. The `samplers.py` module contains two algorithms for Monte Carlo Markov Chain sampling and will support other samplers in future versions (gray font) including the Wolff sampling, Swendsen-Wang sampling, and parallel tempering. The `utils.py` module contains supporting functions for the other modules such as the few examples listed. ConIII's highly modularized structure ensures that contributed algorithms can be appended independently of existing code.

samples can be taken from the model distribution by calling `Solver.generate_samples()` that wraps calls to the `Solver.sampler` instance of a sampling class. For the Ising model, this is typically an instance of `FastMCIsing` from the `coniii.solvers` module. The only other available class is `Metropolis`, but other sampling algorithms listed in the `samplers.py` box in Fig. 2 will be released with later versions of this package.

To keep the solution algorithms generic, they require definition of the maxent model upon instantiation through the definition of the keyword argument `calc_observables`.

REUSE

To contribute either an algorithm for the inverse maxent problem or a sampling technique, we suggest following the template for the classes described in the base `Solver` and `Sampler` classes. New algorithms should be pushed to the Github repository along with an example of an example solution that can be included in the usage guide Jupyter notebook.

QUALITY CONTROL

For checks of basic functionality, the package is released with test modules that can be run with the package pytest.

For a test of the algorithms, the most direct one is to generate a system where the parameters are known, sam-

[1] See Refs [5] and [18] for details and further references on how to estimate information quantities.

ple from the system to generate a data set, and to run the inverse solution to make sure that the found parameters and correlations match. With a finite sample, exact correspondence to the correct parameters is not expected. Furthermore, most of the algorithms only return an approximate solution. For others, the fidelity of the found parameters to the original ones will depend on whether or not the approximation is valid. The Jupyter notebook released with the software provides examples for using the algorithms included in ConIII for a random system of five spins. The included examples show how well the different algorithm might be expected to do in this case. We recommend that that user run this notebook to check that the algorithms when run on their workstation converge closely the resulting solutions that are already included in the provided figures in the notebook.

CONCLUSION

Maxent provides a coherent framework for modeling a diverse body of biological and social systems. The approach is structured to consider easily issues of model selection, embodying a canonical approach for the scientific process: we start with the simplest models and the fewest constraints possible and then increase the complexity of the model until it makes good predictions. In principle, we could add as many constraints as would allow us to better fit the data, but complex models are more computationally expensive and often do not generalize well. Maxent therefore stops before adding all possible constraints, leaving the model otherwise as structureless as possible. In this sense, the maxent approach is a principled method of statistical inference that is relevant beyond its roots in statistical physics. We have built ConIII as an accessible software package in the hope that it will inspire those unfamiliar with maxent approaches to experiment and apply this technique to their own research questions.

[1] E. Aurell and M. Ekeberg. Inverse Ising Inference Using All the Data. *Phys Rev Lett*, 108(9):090201, 2012.

[2] J. Barton and S. Cocco. Ising models for neural activity inferred via selective cluster expansion: structural and coding properties. *J Stat Mech Theory Exp*, 2013(03): P03002, March 2013. ISSN 1742-5468. doi:10.1088/1742-5468/2013/03/P03002.

[3] W. Bialek and R. Ranganathan. Rediscovering the power of pairwise interactions. *arXiv*, pages 1–8, 2007.

[4] W. Bialek, A. Cavagna, I. Giardina, T. Mora, E. Silvestri, M. Viale, and A. M. Walczak. Statistical mechanics for natural flocks of birds. *PNAS*, 109(13):4786–4791, 2012.

[5] W. S. Bialek. *Biophysics: Searching for Principles.* Princeton University Press, 2012.

[6] T. Broderick, M. Dudik, G. Tkačik, R. E. Schapire, and W. Bialek. Faster solutions of the inverse pairwise Ising problem. *arXiv*, pages 1–8, 2007.

[7] C. Castellano, S. Fortunato, and V. Loreto. Statistical physics of social dynamics. *Rev Mod Phys*, 81(2):591–646, 2009.

[8] S. Cocco and R. Monasson. Adaptive Cluster Expansion for Inferring Boltzmann Machines with Noisy Data. *Phys Rev Lett*, 106(9):090601, 2011.

[9] S. Cocco and R. Monasson. Adaptive cluster expansion for the inverse Ising problem: convergence, algorithm and tests. *Journal of Statistical Physics*, pages 1–57, 2012.

[10] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, Hoboken, 2nd edition, 2006.

[11] B. C. Daniels, D. C. Krakauer, and J. C. Flack. Control of finite critical behaviour in a small-scale social system. *Nat Comms*, 8:14301–8, 2017.

[12] E. Ganmor, R. Segev, and E. Schneidman. Sparse low-order interaction network underlies a highly correlated and learnable neural population code. *PNAS*, 108(23): 9679–9684, 2011.

[13] A. Hyvärinen. Some extensions of score matching. *Comput Stat Data Anal*, 51(5):2499–2512, 2007.

[14] E. Ising. *Beitrag zur Theorie des Ferromagnetismus.* PhD thesis, 1924.

[15] E. T. Jaynes. Information Theory and Statistical Mechanics. *Phys Rev*, 106(4):620, 1957.

[16] E. T. Jaynes. *Probability Theory: The Logic of Science.* Cambridge University Press, 2003.

[17] S. Konishi and G. Kitagawa. Information Criteria and Statistical Modeling. Springer New York, New York, NY, 2008.

[18] E. D. Lee, C. P. Broedersz, and W. Bialek. Statistical Mechanics of the US Supreme Court. *J Stat Phys*, pages 1–27, 2015.

[19] D. J. C. MacKay. Information Theory, Inference and Learning Algorithms, 2005.

[20] L. Merchan and I. Nemenman. On the Sufficiency of Pairwise Interactions in Maximum Entropy Models of Networks. *J Stat Phys*, 162(5):1294–1308, 2016.

[21] T. Mora, A. M. Walczak, W. Bialek, and J. Curtis G Callan. Maximum entropy models for antibody diversity. *PNAS*, 107(12):5405–5410, 2010.

[22] I. Nemenman. Information theory, multivariate dependence, and genetic network inference. *arXiv*, 2004.

[23] M. E. J. Newman and G. T. Barkema. *Monte Carlo Methods in Statistical Physics.* Clarendon Press, 1999.

[24] H. C. Nguyen, R. Zecchina, and J. Berg. Inverse statistical problems: from the inverse Ising problem to data science. *arXiv*, 2017.

[25] E. Schneidman, M. J. Berry, RS II, and W. Bialek. Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, 2006.

[26] C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[27] J. Sohl-Dickstein, P. Battaglino, and M. R. DeWeese. Minimum Probability Flow Learning. *arXiv*, 2009.

[28] J. Sohl-Dickstein, P. B. Battaglino, and M. R. DeWeese. New Method for Parameter Estimation in Probabilistic Models: Minimum Probability Flow. *Phys Rev Lett*, 107 (22):220601, 2011.

[29] G. J. Stephens and W. Bialek. Statistical mechanics of letters in words. *Phys Rev E*, 81(6):066119, 2010.

[30] G. Tkačik, E. Schneidman, M. J. Berry II, and W. Bialek. Ising models for networks of real neurons. *arXiv*, 2006.