Convenient Interface to Inverse Ising (CONIII): A package for solving maximum entropy models

Edward D Lee, Bryan C Daniels
[1]*Department of Physics, 132A Clark Hall, Cornell University, Ithaca NY 14850,* [2]*ASU*

CONIII is a Python project for providing a simple interface to various algorithms for solving maximum entropy models. We describe the maximum entropy problem and give an overview of various algorithms that have been proposed to solve it that are implemented as part of CONIII. We then show examples of models that have been solved on various data sets using CONIII.

Caution: Work in progress. Errors rampant.

INTRODUCTION

Many biological and social systems are characterized by collective behavior whether it is the correlated pattern of neuron firing [**?**], protein diversity in the immune system, conflict participation in monkeys, flocking in birds [**?**], statistics of letters in words, or consensus voting in the US Supreme Court [**?**]. Statistical physics is a natural approach to probing such systems precisely because they are collective [**?**]. Recently, with the advent of numerical, analytic, and computation tools, it has become possible to solve for the statistical physics model that corresponds to a particular system, an approach known as the inverse problem. This is in contrast with the typical problem in statistical physics where one postulates 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 Ising model has been of particular interest because of its simplicity and generality. A variety of algorithms have been proposed to solve this problem including mean-field approximation, etc., but different approaches are disparately available on separate code bases in different coding languages, which makes comparison difficult and pedagogy more complicated. CONIII (Convenient Interface to Inverse Ising) is a Python project intended to provide a centralized resource for the inverse Ising problem and provide a base for the addition of more maximum entropy problems in the future. The modular structure of CONIII was inspired by the machine learning package scikit-learn, where each algorithm is implemented in a class. With CONIII, it is possible to solve the inverse Ising problem with a variety of algorithms in just a few lines of code.

We hope to provide a (soft) introduction to the analytic, numerical, and computational techniques used to solve these problems to make them accessible to students especially at the graduate level in physics and beyond (and even advanced undergraduates). With the goal of furthering general understanding, we aim not to be rigorous but to be accurate and comprehensible, pointing to the relevant literature whenever possible. Although we strive to be as clear and explicit as possible across this guide, a good background in mathematics or physics will be extremely helpful. In some places, we have left out steps in the derivations, and we encourage the reader to work out the missing steps.

WHAT IS MAXIMUM ENTROPY?

Shannon introduced the concept of information entropy in his seminal paper about communication over a noisy channel [**?**]. Information entropy is the unique measure of uncertainty that a followed from a three basic postulates. According to Shannon, information entropy, hereon just "entropy", over the probability distribution of possible configurations s of a model is

$$S[p] = -\sum_{s\in\mathcal{S}} p_s \log p_s \tag{1}$$

These configurations could correspond to a particular firing pattern in neurons or a particular configuration of 4 letters in a word.

Given the set of configurations $\mathcal{S}$, entropy is maximized when the probability distribution is uniform $p_s = p_{s'}$, and this is a unique maximum. The uniform distribution is maximally unstructured because there is no structure to exploit to make any predictions. Another way to put this is to say that we are maximally uncertain about what the next observation from this model is (and thus the unfortunate name "information" entropy). If there were a slight bias for any state, we could do better than randomly choosing a state s.

The idea behind the maximum entropy, or maxent, approach is to build a model that is consistent with the data but otherwise as structureless as possible [**?** **?**]. Since entropy is a measure of uncertainty, we can build a contrained maximization problem. From the data, we choose some important constraints indexed by k,

$$\langle f_k(s)\rangle = \sum_{s\in\mathcal{S}} p_s f_k(s) \tag{2}$$

With K constraints, the maximum entropy problem can be solved by the method of Langrangian multipliers

$$\mathcal{L}[p] = -\sum_s p_s \log p_s - \sum_k^K \lambda_k \sum_s p_s f_k(s) \tag{3}$$

$$F = S - \langle E \rangle \tag{4}$$

$$E = -\sum_{\mathrm{k}}^{K} \lambda_{\mathrm{k}} f_{\mathrm{k}}(s) \tag{5}$$

The notation draws out the equivalence between maximizing entropy and minimizing the Helmholtz free energy where by specifying the constraints on the maximization we fix the form of the Hamiltonian E (See Appendix). This formulation makes clear the fundamental connection that statistical mechanics is an inference procedure using the maximum entropy principle [**?**].

The resulting model is a Boltzmann distribution over states

$$p(s) = e^{-E(s)} \Big/ Z \tag{6}$$

with normalization, the partition function,

$$Z = \sum_\sigma e^{-E(s)} \tag{7}$$

otherwise known as the exponential family of models outside of the physics literature. If we fix the average energy of the system, we get the microcanonical ensemble (See Appendix). which is equivalent to the axiom that all microstates have equal probability (that is the maxent distribution).

Ising model

The Ising model is a statistical physics model of magnetism. In its simplest form, it consists of a set of sites σ_{i}, or spins, with 2 possible orientations (up and down), coupled to an external magnetic field h and coupled to each other with couplings J [**?**]. The strength of the magnetic field determines the tendency of each of the spins to orient in a particular direction and the couplings J determine whether the spins tend to point together ($J > 0$) or against each other ($J < 0$). Typically, neighbors are defined as spins that interact with one another given by some underlying lattice structure as in Figure 1.

More generally, we could imagine that each of the spins has a particular bias given by an indexed local field h_{i} as well as particular interactions with neighbors J_{ij}. The corresponding Hamiltonian is

$$E = -\sum_{\langle \mathrm{ij} \rangle} J_{\mathrm{ij}} \sigma_{\mathrm{i}} \sigma_{\mathrm{j}} - \sum_{\mathrm{i}=1}^{N} h_{\mathrm{i}} \sigma_{\mathrm{i}} \tag{8}$$

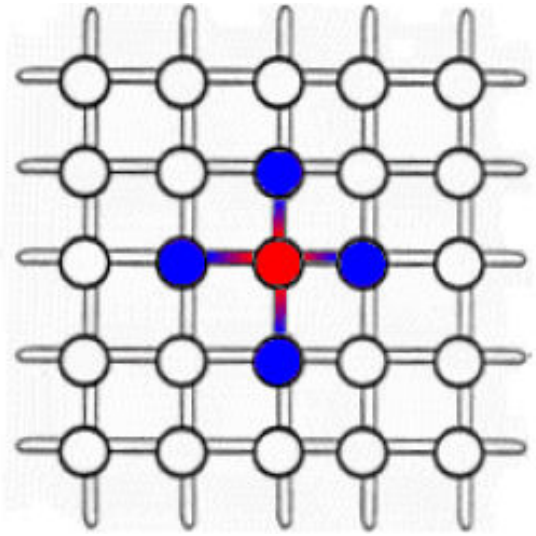

FIG. 1. Example of an Ising model on a lattice.

and as mentioned before states with lower energy are more likely so spin tend to orient along the direction of the local field $h_{\mathrm{i}}^{\mathrm{local}} = \sum_{\mathrm{j}} J_{\mathrm{ij}} \sigma_{\mathrm{j}} + h_{\mathrm{i}}$.

The Ising model corresponds to a pairwise maximum entropy model. Fixing the the magnetizations and pairwise correlations to those observed in the data

$$\langle \sigma_{\mathrm{i}} \rangle_{\mathrm{data}} = \langle \sigma_{\mathrm{i}} \rangle \tag{9}$$

$$\langle \sigma_{\mathrm{i}} \sigma_{\mathrm{j}} \rangle_{\mathrm{data}} = \langle \sigma_{\mathrm{i}} \sigma_{\mathrm{j}} \rangle \tag{10}$$

Constructing the Langrangian from Eq ??

$$\mathcal{L}[p] = -\sum_s p(s) \log p(s) + \sum_{\langle \mathrm{ij} \rangle} J_{\mathrm{ij}} \langle \sigma_{\mathrm{i}} \sigma_{\mathrm{j}} \rangle + \sum_{\mathrm{i}=1}^{N} h_{\mathrm{i}} \langle \sigma_{\mathrm{i}} \rangle \tag{11}$$

$$\frac{\partial \mathcal{L}[p]}{\partial p(s)} = -\log p(s) - 1 + \sum_{\langle \mathrm{ij} \rangle} J_{\mathrm{ij}} \sigma_{\mathrm{i}} \sigma_{\mathrm{j}} + \sum_{\mathrm{i}=1}^{N} h_{\mathrm{i}} \sigma_{\mathrm{i}} \tag{12}$$

$$\log p(s) = -1 + \sum_{\langle \mathrm{ij} \rangle} J_{\mathrm{ij}} \sigma_{\mathrm{i}} \sigma_{\mathrm{j}} + \sum_{\mathrm{i}=1}^{N} h_{\mathrm{i}} \sigma_{\mathrm{i}} \tag{13}$$

$$p(s) = e^{-E(s)} \Big/ Z \tag{14}$$

where to enforce normalization of the probability distribution $\sum_s p(s) = 1$,

$$Z = \sum_s e^{-E(s)} \tag{15}$$

Thus, the resulting model is exactly the Ising model mentioned earlier.

Finding the parameters that match the constraints is equivalent to minimizing the Kullback-Leibler divergence

between the model and the data [**?**]

$$D_{KL}(p_{\rm data}||p_{\rm ME}) = \sum_s p_{\rm data} \log\left(\frac{p_{\rm data}(s)}{p_{\rm ME}(s)}\right) \tag{16}$$

$$\frac{\partial D}{\partial \lambda_k} = \sum_s p_{\rm data}(s) \frac{\partial(-E - \log Z)}{\partial \lambda_k} \tag{17}$$

$$0 = \langle f_k \rangle_{\rm data} - \langle f_k \rangle_{\rm ME} \tag{18}$$

[Of course, we also have to show that the problem is convex.] In other words, the parameters of the maximum entropy model are the ones that minimize the information theoretic "distance" to the distribution of the data. Note that these parameters are given by the data and so there is search for the best parameters in the conventional sense.

When the nonlinear equations are coupled, this cannot be solved analytically in general. Below we discuss some analytical approximations and numerical techniques for approaching the inverse problem.

The primary difficulty for solving probabilistic models is calculating the normalization because of the size of the state space. Most of the methods below involve avoiding that problem in the first place (Monte Carlo sampling, MCH) or approximations to drastically reduce the state space (pseudolikelihood, adaptive cluster expansion).

EXAMPLES

Supreme Court. Neurons. Monkeys. Genes.

EXACT ENUMERATION

One approach is to write the equations down and feed them into a standard optimization algorithm. This approach does not scale well because the number of terms in the partition function grows exponentially with system size. For relatively small systems $n \leq 15$ on a typical desktop computer, however, this approach is feasible.

This is implemented in the `Exact` class.

MEAN-FIELD METHODS

[I'm imagining that we neatly tie together the different techniques. As far as I know, they're all different approaches and no one's done a real good job relating one technique to another. It's easy enough to go read a bunch of different papers about each method, but it's not so clear why one is preferable over another.]

Analytic approximations simplify the equations to an extent where they are tractable.

MONTE CARLO METHODS

Perhaps the most straightforwards and most expensive route to solving these equations is to use Monte Carlo Markov Chain sampling to approximate the distribution and guess how to adjust the parameters appropriately. We will not explain MCMC methods in detail because of the plethora of useful texts [**?**]. [For those unfamiliar with MCMC sampling, we give a brief introduction in Appendix A.]

The simplest MCMC algorithm is to generate a random sample to estimate the constraints $\langle f_{\rm k} \rangle$, and then to implement a learning rule to adjust the parameters. This can be combined with a variety of stochastic gradient descent methods to reduce the number of sampling steps.

Monte Carlo histogram

Since the sampling step is expensive, could we reuse a sample for more than one gradient descent step? This is the idea behind Monte Carlo histogram [**?**]. Instead of sampling every time we change the parameters, we can predict how the distribution will change if we modify the parameters slightly. Given that we have a sample with probability distribution p_s generated with parameters $\lambda_{\rm k}$, we would like to estimate the distribution p'_s from parameters $\lambda'_{\rm k} = \lambda_{\rm k} + \Delta\lambda_{\rm k}$.

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

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

To estimate the average,

$$\sum_s p'_s f_k(s) = \frac{Z}{Z'} \sum_s p_s e^{\sum_k \Delta\lambda_k f_k(s)} f_k(s) \tag{21}$$

But we have a sampled approximation to p.

$$\langle f_k \rangle' = \frac{Z}{Z'} \left\langle e^{\sum_k \Delta\lambda_k f_k(s)} f_k(s) \right\rangle_{\rm sample} \tag{22}$$

Likewise, the ratio of the partition function can be estimated

$$\frac{Z}{Z'} \approx 1 \Big/ \left\langle e^{\sum_k \Delta\lambda_k f_k(s)} \right\rangle_{\rm sample} \tag{23}$$

The main difficulty with MCH is choosing a learning rule for how to update the Lagrangian multipliers $\{\lambda_k\}$ at each iteration while being careful to stay within the bounds of a reasonable extrapolation. One suggestion is to update the parameters with some inertia

$$\Delta\lambda_k(t+1) = \Delta\lambda_k(t) + \eta\Delta\lambda_k(t-1) \tag{24}$$

$$\Delta\lambda_k(t) = \epsilon\left(\langle f_k \rangle' - \langle f_k \rangle\right) \tag{25}$$

This has the correct fixed points. One suggestion is to shrink ϵ exponentially with the number of iterations. In the code, there are more parameters $\Delta\lambda_{\max}$ and $\Delta\lambda_{\text{norm}} \sum_k \sqrt{\Delta\lambda_{\text{k}}^2}$ that restrict. Another heuristic is to begin with estimates on a small data set and increase the size of the data set as we refine our estimates of the parameters.

Typically, one must check in on the progress of the algorithm to tune the various parameters. If the extrapolation steps or the gradient steps are too large, the algorithm will fail to converge (or may even start diverging).

MCH is implemented in the `MCH` class.

PSEUDOLIKELIHOOD

An analytic approximation that reduces the computational complexity of the problem.

MINIMUM PROBABILITY FLOW

Minimum probability flow is an efficient method using an analytic approximation to write an easily computable objective function [**?**]. Instead of estimating the probability distribution, the overarching idea is to analytically approximate how the probability distribution *changes* as we modify the *configurations*.

As before, we start with minimizing the Kullback-Leibler divergence, but instead of taking the derivative with respect to the parameters, we would like to take the derivative with respect to the states [**?**]. If the state space is discrete, we can instead postulate a set of dynamics for how the probability evolves (more on this later), and take the derivative with respect to time. The idea is that the change in the probability distribution relative to the data goes to zero when we have converged to the data. Thus, we can now take a derivative with respect to the parameters.

As is explained [**?**], one set of dynamics (they must converge to the equilibrium distribution?) is simply Monte Carlo where the master equation gives

$$\dot{p}_{\text{i}} = \sum_{j \neq i} \Gamma_{\text{ij}} p_{\text{j}} - \sum_{\text{j} \neq \text{i}} \Gamma_{\text{ji}} p_{\text{j}} \tag{26}$$

and Γ_{ij} is the transition probability from state j to state i.

So we are looking for how the distance between data and model changes when we evolve the dynamics of the model

$$\partial_t D_{KL}(p^{(0)} || p^{(t)}(\{\lambda_k\})) = \sum_{\text{i} \notin \mathcal{D}} \dot{p}_{\text{i}}(\lambda_k) \tag{27}$$

$$K(\{\lambda_{\text{k}}\}) = \sum_{\text{i} \notin \mathcal{D}} \dot{p}_{\text{i}}(\lambda_k) \tag{28}$$

[Typically with large systems, numerical precision errors can become a problem because of the exponential form of $p(\lambda_{\text{k}})$. By taking the logarithm of the objective K can be crucial to an accurate implementation.]

MPF is implemented in the `MPF` class.

CLUSTER EXPANSIONS

Implemented in the `ClusterExpansion` class.

BETHE/KIKUCHI FREE ENERGY/CAVITY METHODS

How to validate a maxent model

The hope of maxent is that by fixing some aspects of the probability distribution, we can forget about the rest. In this sense, maxent is not a typical fitting method where the optimal parameters are sought such that the model is as good as possible. Instead, we specify which aspects of the system we think are important and the resulting "parameters" come from the maximum entropy principle.

Thus, one way to test the model is to consider how well we do in predicting the rest of the distribution. If the rest of the distribution is not well fit, certainly we must go back and reconsider which constraints to impose. On the other hand, finding that the rest of the probability distribution is fit is all good and well, but it does not necessarily mean we have found the right model [cite much discussion here].

WHY NOT MAXIMUM ENTROPY?

The microcanonical ensemble and maximum entropy

The conventional textbook in statistical mechanics first introduces the concept of entropy as a way of counting the phase volume available to the system at a given energy

$$S(E) = k_B \log \Omega(E) \tag{29}$$

where k_B is Boltzmann's constant and Ω the number of states between E and $E + \delta E$. Temperature is defined as

$$\frac{1}{T} = \frac{\partial S}{\partial E} \tag{30}$$

begins with the concept of a small system coupled to a heat bath. In this limit, we can linearly expand thermodynamic quantities about the energy of the bath

$E_{\text{bath}} = E - E_s$

$$S_{\text{bath}}(E - E_s) \approx S_{\text{bath}}(E) - E_s \left.\frac{\partial S}{\partial E_s}\right|_{E-E_s} \tag{31}$$

$$= S_{\text{bath}}(E) - \frac{E_s}{T} \tag{32}$$

$$e^{S_{\text{bath}}(E-E_s)/k_B} = e^{S_{\text{bath}}(E)/k_B} e^{-E_s/k_B T} \tag{33}$$

Since the entropy is proportional to the density of states at a particular energy E_s, Eq ?? corresponds to

$$p(E_s) = e^{-E_s/k_B T}/Z \tag{34}$$

With partition function Z. In other words, the Gibbs measure takes exponential form. Eq ?? can be rewritten in terms of free energy

$$\log p(E_s) = -E_s/k_B T - \log(Z) \tag{35}$$

Averaging both sides over all energy configurations and rearranging

$$-k_B T \log Z = \sum_s p(E_s) E_s + k_B T \sum_s p(E_s) \log p(E_s) \tag{36}$$

Remembering the fundamental postulate of statistical mechanics that all states are equally likely

$$-k_B T \log Z = \langle E_s \rangle - TS \tag{37}$$

$$F = \langle E_s \rangle - TS \tag{38}$$

This equation tells us how the Helmholtz free energy is related to the internal energy and the entropy of the system.

When we say that free energy is minimized for a system with Hamiltonian E_s at equilibrium, we are equivalently saying that entropy is maximized. Entropy maximization is the crux of statistical physics models.

Notes on Monte Carlo Markov Chain sampling

The key points behind MCMC sampling is ergodicity and detailed balance. Ergodicity just means that we can get from one state to another in a finite number of steps, and this ensures that we don't get stuck in a few states. Detailed balance is a sufficient but not necessary condition (stronger than needed) for ensuring that the equilibrium distribution matches the distribution that we seek. There are some specialized algorithms that don't satisfy detailed balance but do produce the desired distribution.

To check whether an algorithm works, one must prove that both these conditions are satisfied. Ergodicity is usually trivial. Typically, we write down the condition for detailed balance for any two states a and b,

$$p(a|b)p(b) = p(b|a)p(a) \tag{39}$$

$$p(b)/p(a) = p(b|a)/p(a|b) \tag{40}$$

$$e^{-(E_b - E_a)} = \tag{41}$$

A reasonable way to do this would be to take $p(b|a) = e^{-(E_b - E_a)}$ and $p(a|b) = 1$ when $E_b > E_a$.

$$p_a\, T(a \to b) A(a \to b) = p_b\, T(b \to a) A(b \to a) \tag{42}$$

$$p_a/p_b = T(b \to a) A(b \to a)/T(a \to b) A(a \to b) \tag{43}$$

where T is the transition probability and A is the acceptance probability. For a Boltzmann type model, this means that this ratio must be equal to $\exp(-\beta(E_a - E_b))$.

Swendsen-Wang

As an example, we consider the Swendsen-Wang cluster algorithm. In this algorithm, the first step is to form bonds between like spins (to grow clusters) then we flip to any configuration permitted by the current bond structure which is uniform. If you work through the calculations (GNB V pg 81), you will find that the probability of transitioning between states depend on the bonds that could have formed between like spins but didn't, and this is equal to the energy difference between the states.

Wolff

Another example is the Wolff algorithm (see GNB V pg 83) which is like the SW algorithm except that only a single cluster is build and flipped at a time with probability 1, i.e. the acceptance probability is 1 in the original algorithm. We start with any random site as the initial spin then we build a cluster spreading out from that spin to its nearest neighbors where the choose the probability of choosing a neighbor to be $1 - \exp(-2J_{ij})$.

The random fields can be accounted for in the acceptance probabilities (so the probability of a cluster flipping is no longer 1). Working through the math, you will find that the cluster growth accounts for the ratio of the energies that come from the couplings but to account for the fields, the probability of a particular orientation of the clusters has to be

$$p(\Sigma_n) \propto \exp\left(\sum_i h_i \sigma_i\right) \tag{44}$$

which we can easily simulate using something like the Metropolis algorithm. Obviously, this will increase the

correlation time between samples because we will not always accept a cluster flip. This can be especially problematic when the clusters become large and have similar fields, but as long as the sum of the distribution of local fields for spins in a cluster is symmetric about 0, this will not be too slow. Certainly, it is faster than any local flip algorithm.

A simple example that I worked through is with 4 spins and the transition probability between two different states. Here, we can easily enumerate all possible ways of transitioning between these two states using the Wolff algorithm.

Replica exchange Monte Carlo (REMC)

REMC simulates multiple replicas of the system at different temperatures simulaneously and allow states to be exchanged between them. The idea is that an energy barrier may be very difficult to cross below a certain critical temperature, but there may be a trajectory allowed by diffusing through higher temperatures to cross the boundary.

Instead of considering the ensemble of a single system, we consider the ensemble of multiple independent systems. Thus, the joint probability distribution on the extended state space is

$$p(\sigma, \beta_n). \tag{45}$$

where $\beta_n = 1/T_n$ is the inverse temperature.

The simplest possible assumption for the shape of this distribution would be that the probability of occupying any particular temperature β_n is uniform $p(n) = 1/N_T$. It turns out that this is optimal according to some criterion (see citation in Kerler and Rehberg). Note that this is not the same as just simulating a set of replicas for which the partition funtion would be

$$Z = \prod_m \sum_\sigma \exp\left[-\beta_m H(\sigma)\right] \tag{46}$$

This is for the obvious reason that the average energy is a function of T so a simulation of Eq 46 would spend much more time exploring lower temperatures than higher temperatures. This makes it difficult to have the system use higher energy states to cross energy barriers.

So instead of Eq 46, we can include a term g_n to compensate

$$Z = \prod_m \sum_\sigma \exp\left[-\beta_m H(\sigma) + g_m\right] \tag{47}$$

How do we find g_n?

Under the assumption that the probability $p(n)$ is a constant,

$$p(n) = e^{g_n} \sum_\sigma \exp\left[-\beta_n H(\sigma)\right] / Z \tag{48}$$

$$g_n = \log[p(n)\, Z] + \tilde{Z}_n \tag{49}$$

$$g_n \to \tilde{Z}_n \tag{50}$$

(remembering that constants don't matter in the relation between energy and probability) where

$$\tilde{Z}_n = \sum_\sigma \exp\left[-\beta_n H(\sigma)\right] \tag{51}$$

One way to find the g_n that satisfy this condition is to come up with an iterative algorithm that converges to a fixed point. One suggestion is to estimate g_n by reweighting the distributions from adjacent replicas.

Now, it remains to find a set of β_n that is optimal for simulating relaxing into equilibrium quickly. One suggestion to make the algorithm spend an equal amount of time at each temperature. Kerler and Rehberg suggest a method.