Intro to Chaotic Bits

Kyle Lee

August 2024

1 Introduction

Only part of the OPUS lab group is working on chaotic bits. This applies to you if you plan to work on chaotic bits.

Chaotic bits (c-bits) are an alternative (and experimental) implementation of the p-bit, first described by Hideyuki Suzuki's 2013 paper "Chaotic Boltzmann Machines". "Chaotic" typically describes systems that are extremely sensitive to initial conditions and evolve so unpredictably so as to appear random. A prime example of this is the weather. The physical laws governing the weather may be deterministic and depend on initial conditions, but the weather evolves so quickly and unpredictably that there appears to be randomness involved.

Here is the p-bit update rule, which should be familiar by now:

$$m_i = \text{sgn}[\tanh(I_i/T) - \text{rand}_u(-1, 1)] \tag{1}$$

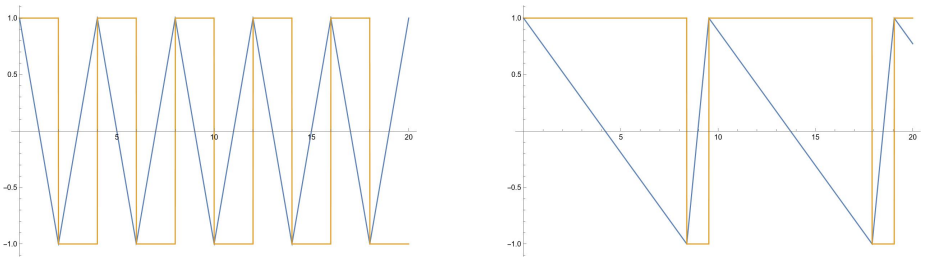
In the case of c-bits, we replace Eq. 1 with a deterministic update rule:

$$\begin{aligned} \frac{dx_i}{dt} &= -m_i + \tanh(\beta I_i) \\ x_i = +1 &\implies \text{set } m_i \text{ to } +1 \\ x_i = -1 &\implies \text{set } m_i \text{ to } -1 \end{aligned} \tag{2}$$

where β is inverse temperature and m_i is the current state/spin of the c-bit. A single c-bit can be thought of as a billiard x_i and a latch m_i. The billiard ball x_i is bouncing between the boundaries +1 and -1. This billiard has a tunable slope dx_i/dt (which is its velocity). The slope is dependent on the input to the c-bit, I_i. When the billiard hits a boundary at +1 or -1, the state of the c-bit m_i is latched to that value, and the billiard changes direction.

This is shown graphically in Fig. 1. We set the temperature to a constant value, $T = 1$. Fig. 1a shows a c-bit under constant $I_i = 0$. The billiard's upward velocity is the same magnitude as its downward velocity. The c-bit bounces between the +1 and -1 boundaries with a slope of magnitude 1, and whenever the billiard x_i hits a boundary, the state of the c-bit m_i is latched to that value, and the billiard reverses its direction.

Fig. 1b shows what happens when we increase the input $I_i = 1$. Now, the billiard quickly goes from -1 to +1, and slowly goes from +1 to -1. The



(a) $I_i = 0$ (b) $I_i = 1$

Figure 1: Visualization of c-bits with constant input I_i and constant temperature $T = 1$. m_i is shown in yellow, and x_i is shown in blue.

latching action of the c-bit state m_i causes the c-bit to most frequently occupy the $m_i = +1$ state. The same logic works in the other direction: if I were to decrease the input I_i in the negative direction and input $I_i = -1$, then the c-bit would occupy the $m_i = -1$ state most frequently.

In order to understand how we came up with our c-bit definition (Eq. 2), let's consider a single probabilistic bit under constant input I_i. The probability of being in the +1 state divided by the probability of being in the -1 state is as follows:

$$\frac{p(m_i = +1)}{p(m_i = -1)} = \exp\left(2\beta I_i\right) \tag{3}$$

We defined c-bits such that, under constant I_i, continuous time sampling at a sufficiently high rate will yield the same ratio. In other words, the lifetime of a c-bit in the +1 state, $\tau(m_i = +1)$, divided by the lifetime of a c-bit in the -1 state, $\tau(m_i = -1)$, is similar to the amount of time a p-bit spends in the +1 state divided by the amount of time a p-bit spends in the -1 state. More formally:

$$\frac{\tau(m_i = +1)}{\tau(m_i = -1)} = \frac{\left|\dfrac{dx_i}{dt} \quad \text{when } m_i = -1\right|}{\left|\dfrac{dx_i}{dt} \quad \text{when } m_i = +1\right|} = \exp\left(2\beta I_i\right) \tag{4}$$

The key point here is that c-bits are deterministically updated according to Eq. 2, as opposed to stochastically updated p-bits. Furthermore, a single c-bit under constant input is really a periodic square wave, which does not seem particularly interesting or useful. The magic happens when we couple c-bits together in large networks. We use the exact same synapse function as we do for p-bits:

$$I_i = \sum_j J_{ij} m_j + h_i \tag{5}$$

and we solve a system of coupled ordinary differential equations (ODEs) (Eq. 2) in *continuous time*. This is different from the way we simulate p-bits in software using Gibbs sampling. For p-bits, we sequentially update each bit in the system, and we call this one Monte Carlo sweep, at which point we sample the state of the system. For c-bits, there is no discrete notion of time. It is just a system that is dynamically changing its state as the time t increases. I could choose to sample the spin configuration any time I want. A simple sampling method would be to sample at natural numbers $t = \{1, 2, 3, ...\}$.

There is little justification for c-bits properly functioning as Ising machines and sampling from the Boltzmann distribution. Existing results show that c-bit networks seem to mimic the behavior of p-bit networks for sampling and optimization, but it is still not clear that c-bits can be a "drop-in replacement" for p-bits. We are also experimenting with injecting deterministic c-bits with explicit noise in order to improve their performance in optimization.

There are many challenges and open questions as well. This is an example of a stiff system of ODEs, which have rapidly changing dynamics and are notoriously difficult to evaluate and simulate in software. ODE solvers use discretization. When we use ODE solvers, whether in MATLAB, Mathematica, or even self programmed, we are introducing rounding errors due to double precision as well as errors from the solver itself being unable to perfectly solve the system. Furthermore, we typically randomize the state of a c-bit system at the start of a simulation, which is an additional source of explicit randomness.

Due to these factors, we suspect that we may be surreptitiously enjoying the benefits of hidden sources of randomness. While c-bits are purportedly deterministic, in our actual implementation, that may not quite be the case. If this were implemented in hardware, we may end up benefiting from sources of thermal noise and other noise.

2 Recommended papers

1. "Noise-augmented Chaotic Ising Machines for Combinatorial Optimization and Sampling", Kyle Lee, Shuvro Chowdhury, and Kerem Camsari, 2024

2. "Chaotic Boltzmann Machines", Hideyuki Suzuki, Scientific Reports, 2013