An introduction to thermodynamic computing
for computer scientists

A single computing paradigm has dominated the first chapter of mainstream computing. And with good reason, classical architectures are great at running the kind of software that now scaffolds our world. This software implements deterministic functions; given the same input we expect an identical output. But the kind of software we'd like to run is changing. As programs become more intelligent, "learning" in real-time, the deterministic constraint relaxes. In fact, stochasticity becomes an asset. Our best models of physics are probabilistic, and so too are our best models of learning. In this note, we give some intuition for what these non-deterministic *thermo*dynamics are like, and how nature's computations might be leveraged to build programs that learn efficiently (and even tricked into performing linear algebra). We assume only a familiarity with calculus and probability.

1 Modeling a noisy world

Around 50 B.C., Lucretius posited the existence of atoms as an explanation for the movement of dust particles, their "tossing" revealed in a sunbeam.

> The sun's light and the rays, let in, pour down
> Across dark halls of houses: thou wilt see
> ...
> The ceaseless tossing of primordial seeds
> ...
> Which here are witnessed tumbling in the light:
> Namely, because such tumblings are a sign
> That motions also of the primal stuff
> Secret and viewless lurk beneath, behind.
> For thou wilt mark here many a speck, impelled
> By viewless blows, to change its little course,
> And beaten backwards to return again,
> Hither and thither in all directions round.
> ...
> Thus motion ascends from the primeval [atom]s on,
> And stage by stage emerges to our sense,
> Until those objects also move which we
> Can mark in sunbeams, though it not appears
> What blows do urge them.
>
> **(from "On the Nature of Things")**

Although the phenomenon Lucretius describes is best explained by air currents, this description accurately presages what we now call "Brownian motion", the mathematics of moving particles suspended in a gas or fluid.

1.1 Brownian motion as a stochastic process

Modeling the movement of microscopic particles is difficult using the physics we've developed to predict how macroscopic objects move. Computationally, it's untenable to keep track of the position of every relevant molecule in the fluid. Instead, let's take a somewhat opposite approach; we'll assume the particle's movement during each time-step is random. In particular, let's start by modeling the particle's movement as a random walk. Given independent and identically distributed variables $\xi_1, \xi_2, \dots$ with mean 0 and variance 1 (ξ_i represents the movement of the particle at time $t = i$), we can represent the trajectory of the particle as the sequence of positions:

$$0, \xi_1, (\xi_1 + \xi_2), (\xi_1 + \xi_2 + \xi_3), \dots$$

Intuitively, this single-point approximation is likely not accurate. And indeed the quality of our predictions under this approach will be directly related to the magnitude of samples we can acquire. Sampling from each ξ_i for varying lengths of trajectories, we define particle positions:

$$X_k = \xi_1 + \cdots + \xi_k$$

so the set

$$\begin{aligned}\{X_0, X_1, X_2, X_3, \dots\} = \{\ &\{0\},\\ &\{0 + \xi_1\},\\ &\{0 + \xi_1 + \xi_2\},\\ &\{0 + \xi_1 + \xi_2 + \xi_3\}, \dots\}\end{aligned}$$

defines a probability distribution over the particle's position at each time step. We call this family of random variables $\{X_0, X_1, \dots\}$ a **stochastic process**. By the central limit theorem, we know that for large enough n, the distribution of $\frac{1}{\sqrt{n}} X_n$ converges to the unit Normal distribution $\mathcal{N}[0, 1]$.[1] This tells us that if we pick a large time-step, we can model the particle's position at that time using a rescaled Gaussian.

1.2 Brownian motion as a Wiener process

Nature runs in continuous time, not in discrete steps. What happens in the continuum limit, or as the intervals become smaller and smaller? We can define the analogous **continuous time stochastic process** by interpolating over the real interval $I = [0, 1] \in \mathbb{R}$. Letting

$$X_k(t) = \sum_{1 \le i \le \lfloor kt \rfloor} \xi_i$$

[1] If the ξ_i are Gaussians, then $\frac{1}{\sqrt{n}} X_n$ is exactly $\mathcal{N}[0, 1]$ for any n.

where $t \in I$, we now have a new stochastic process $\{X_0(t), X_1(t), X_2(t), \dots\}$. Reinterpreting our central limit theorem result, we can say for large enough n, $\frac{1}{\sqrt{n}}X_n(t)$ approaches $\mathcal{N}[0,t]$. Donsker's theorem [7] extends this result by proving that as $n \to \infty$, the process $\frac{1}{\sqrt{n}}X_n(t)$ converges to a Wiener process. A "Wiener process" $\{W_t\}$ is a family of random variables W_t indexed by non-negative real numbers t with the following properties:

1. $W_0 = 0$
2. With probability 1, the function $t \to W_t$ is continuous in t
3. The process W_t has stationary[2], independent[3] increments
4. $W_{(t+s)} - W_s \sim \mathcal{N}[0,t]$ for $(0 \leq s \leq t)$

This model of Brownian motion is so widely accepted that many identify the terms "Brownian motion" and "Wiener process".

1.3 Brownian motion as the original Langevin equation

How do we reconcile these thermodynamic fluctuations with Newton's laws of motion? In 1908, Paul Langevin showed how to combine macroscopic and noisy microscopic forces to predict the time evolution of particles. The original Langevin equation describes the time evolution of the position of the Brownian particle. We represent the force acting on the particle using Stoke's law (a force proportional to the particle's velocity) and a noise term that represents collisions with other molecules. So Newton's second law ($F = ma$) becomes:

$$ma = -\lambda v + \eta(t) \tag{1}$$

where λ is a drag coefficient, $v = \frac{dx}{dt}$ is the velocity of the particle, $a = \frac{dv}{dt} = \frac{d^2x}{dt^2}$ is the acceleration, m is the mass, and $\eta(t)$ is exactly a Wiener process. This **stochastic differential equation** (SDE) captures the dynamics of the system as it evolves. We can generalize this equation by considering energy landscapes with both kinetic and potential energies. We simply add a term to (1):

$$ma = -\lambda v - \nabla_x U(x) + \eta(t) \tag{2}$$

where $U(x)$ is the potential energy of the system.

[2] "Stationary increments" guarantees that for any $0 < s, t < \infty$, the distribution of $W_{t+s} - W_s$ is the same as that of $W_t - W_0 = W_t$ [10]

[3] "Independent increments" guarantees that for every choice of $(0 \leq s_1 \leq t_1 \leq s_2 \leq t_2 \cdots \leq s_n \leq t_n \leq \infty)$ with $s_i, t_i \in \mathbb{R}_{\geq 0}$, the "increment" random variables $W_{t_1} - W_{s_1}$, $W_{t_2} - W_{s_2}, \dots, W_{t_n} - W_{s_n}$ are jointly independent [10]

1.4 Overdamped Langevin dynamics

Consider the case in which the inertia of the particle ma is outweighed by the damping constant $-\lambda$. In this case, our dynamics reduce to

$$\begin{aligned} ma &= -\lambda v - \nabla_x U(x) + \eta(t) \\ \lambda v \approx \lambda v + ma &= -\nabla_x U(x) + \eta(t) \\ v &\approx -\frac{1}{\lambda}\nabla_x U(x) + \frac{1}{\lambda}\eta(t) \end{aligned}$$

We'll refer to this formulation as **overdamped** dynamics.

1.5 Thermal equilibrium

It's difficult to determine much about an arbitrary SDE's equilibrium distribution. However, over long time-scales, the Langevin equation reduces to the Boltzmann distribution, the probability distribution function for particles in thermal equilibrium[4].

$$p(x) \propto \exp(-U(x)) \tag{3}$$

This distribution can be derived as the distribution that maximizes entropy subject to the following constraints:

1. $\sum_i p(x_i) = 1$
2. $\sum_i p(x_i)U(x_i) = \langle U \rangle$

These constraints guarantee that the probabilities sum to 1, and the system has fixed average energy (system is at equilibrium).

2 A noisy world, modeling

So far, we've derived the mathematics that we use to model physical systems driven by the fluctuation of microscopic particles. Now let's explore how we can use those same physical systems to compute mathematical quantities of interest. Our general strategy will be (1) Identify target mathematical quantity with a corresponding target probability distribution (2) Design a physical system whose equilibrium distribution matches our target probability distribution.

2.1 Solving Linear Systems

The problem of solving a system of linear equations can be stated succinctly. Given an invertible matrix $A \in \mathbb{R}^{d\times d}$ and nonzero $b \in \mathbb{R}^d$, find $x \in \mathbb{R}^d$ such that

$$Ax = b$$

[4]We leave out the factor $\frac{1}{k_B T}$ for simplification. For a full treatment and motivation, we recommend Landau and Lifshitz [11]

Without loss of generality, we may assume A is symmetric and positive definite (SPD). If not, then we may consider the system $A^\top Ax = A^\top b$ whose solution $x = A^{-1}b$ is also the solution of $Ax = b$. How can we use the physical world to solve this system? Suppose we have a physical system with potential energy function

$$U(x) = \frac{1}{2}x^\top Ax - b^\top x$$

where $A \in SPD_d(\mathbb{R})$ ($d \times d$, with values in $\mathbb{R}$). This is common; any nonlinear potential energy looks like a harmonic oscillator[5] when expanded around a local minimum. Suppose further, that we allow this device to come to thermal equilibrium with the environment. As we saw in 3, at equilibrium the Boltzmann distribution describes the probability the oscillators have a given spatial coordinate x. I.e.

$$p(x) \propto \exp(-U(x))$$

Plugging in and expanding, we can see that this corresponds to a multivariate Gaussian distribution.

$$\begin{aligned}
e^{-U(x)} &= e^{-(\frac{1}{2}x^\top Ax - b^\top x)} \\
&= e^{-(\frac{1}{2}(A^{-1}b-x)^\top A(A^{-1}b-x) - \frac{1}{2}b^\top A^{-1}b)} \\
&= e^{-\frac{1}{2}(A^{-1}b-x)^\top A(A^{-1}b-x) + \frac{1}{2}b^\top A^{-1}b} \\
&= e^{-\frac{1}{2}(A^{-1}b-x)^\top A(A^{-1}b-x)} e^{\frac{1}{2}b^\top A^{-1}b} \\
&= e^{\frac{1}{2}b^\top A^{-1}b} e^{-\frac{1}{2}(A^{-1}b-x)^\top A(A^{-1}b-x)} \\
&= \frac{1}{\sqrt{(2\pi)^k|\Sigma|}} e^{-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)} \quad (\text{where } \Sigma = A^{-1}, x = A^{-1}b)
\end{aligned}$$

We use the generalized Gaussian integral (with respect to b) to arrive at this final line. This tells us that at thermal equilibrium, the spatial coordinate x is a Gaussian random variable, whose mean is equivalent to the solution of linear systems.

$$x \sim \mathcal{N}[A^{-1}b, A^{-1}]$$

So given the appropriate hardware [1] (conveniently parameterizes a physical system), we can spawn a physical system with the correct potential energy, allow that system to come to equilibrium, and then sample its state to get an approximation to the solution of our linear system. We can use similar protocols to compute other linear algebra primitives, such as the determinant or inverse of an (SPD) matrix.

2.2 Higher Order Optimization

In a supervised learning setting, we're given data $\{x_n, y_n\}_{n=0}^N$, a hypothesis class of models $\mathcal{F}$ and our goal is to find a function $f_\theta \in \mathcal{F}$ that maps input features

[5] For example, a simple mass-spring system

to their correct labels $f_\theta : x_n \mapsto \hat{y}_n = y_n$. In order to compare models, we often define a loss metric $l(\theta)$ that we want to minimize. Given such a metric, we can compute how it changes (at a point), as we change model parameters. Then we can nudge the model parameters in the direction that most sharply decreases our metric. In practice, this usually amounts to gradient descent:

$$\theta_{t+1} = \theta_t - \alpha \nabla l(\theta_t) \tag{4}$$

However, the gradient is a linear approximation to our objective. It doesn't take into account the fact that nudging the parameters the same amount in different directions may have a different magnitude of impact on the objective. In order to take into account the curvature of the loss landscape, we need second-order methods. There are many such methods[6], which have been shown to outperform gradient descent. They usually take the shape of adding a "pre-conditioning" matrix to the update[7]:

$$\theta_{t+1} = \theta_t - \alpha P^{-1}(\theta_t) \nabla l(\theta_t) \tag{5}$$

Where $P(\theta_t)$ contains information from the Hessian, (square matrix of second-order partial derivatives) of the loss, or an approximation thereof. However, even computing approximations of the Hessian is expensive on digital hardware. If instead we design a physical system whose potential energy is exactly our loss, we can simply allow the system to come to equilibrium (where low-energy states are more likely) and then read off the state as our update to the weight parameters. Practically, the steps are similar to those in 2.1, only we align the equilibrium distribution with the expensive pre-conditioner matrix and gradient product $P(\theta_t)\nabla l(\theta_t)$ [5] [6].

Quadratic approximations to the loss landscape are still only reliable locally. This means we still need to make many small updates in order to effectively descend the loss landscape. If future hardware allowed for higher order non-linearities, then we could potentially map a larger region of the loss landscape with high fidelity, making optimization much more efficient.

2.3 Bayesian Inference

As described in 2.2, we often turn model selection into an optimization problem by choosing a loss function, and attempting to update the parameters of our model to minimize it. Another often mathematically equivalent perspective considers our candidate model to be a probability distribution $f(y|x, \theta)$, and understands the goal of model selection as finding the distribution that maximizes the likelihood of the observed data. Seen from such a perspective, we can use many tools from the "probabilistic" tool kit in order to find such a distribution.

A major category of such tools comes from Bayesian statistics. This theory proposes that instead of modeling probabilities purely based on frequencies, we

[6]For example, Newton's Method, Natural Gradient Descent [3], K-FAC [13], Shampoo [8]
[7]For more background, we like these notes [9]

should begin with some initial (prior) beliefs, and then update our beliefs based on observations. Bayes theorem formalizes this:

$$\mathrm{P(H|D)} = \frac{\mathrm{P(D|H)P(H)}}{\mathrm{P(D)}} \tag{6}$$

where D is the data we've observed, and H is our hypothesis. In this case, we're modeling our beliefs about the parameters ($\mathrm{H} = \theta$). Our prior P(H) accounts for our initial beliefs about what we think are likely good model parameters (maybe from past experience), and as we observe data, we update our beliefs in conjunction with the likelihood of the data, given our current hypothesis about the data generating function P(D|H). The "posterior" distribution (LHS) represents our belief (distribution) after observing the data. However, computing the posterior distribution is often difficult, since P(D) [8] often amounts to computing a very large or undefined integral.

In order to get around this, we've developed methods that allow us to sample from unknown (posterior) distributions. Imagine first, a simple procedure where we throw darts uniformly about a rectangle that totally subsumes the target two-dimensional distribution. As long as we know the outline of the target distribution, after many throws we can simply discard any darts that fall outside the outline, and the collection of darts left will be a good approximation to our target distribution.

This is a kind of Monte Carlo method that uses rejection sampling. We can use the numerator of (6) as our "outline", since we know our posterior is simply a scaled version of this term. And instead of throwing darts uniformly, we can develop smarter strategies to achieve a good approximation faster. Suppose by chance, we throw a dart inside the lines. We should use that information to throw another dart close by. "Close by" could mean sampled from a Gaussian centered at the position of our last dart.[9] This is the well-known Metropolis-Hastings algorithm.

Even better, if we knew in which direction the density distribution was increasing around our point, we could choose a point that takes a step in this direction. We measure this using the score function[10] $\nabla_x \log p(x)$. In order to make sure we still explore lower density areas, we add some noise back in.[11] Our protocol for throwing darts then looks like: (1) Initialize x_0 by sampling from an arbitrary prior distribution $x_0 \sim \pi(x)$, (2) Iterate:

$$x_{i+1} \leftarrow x_i - \epsilon \nabla \log p(x) + \sqrt{2\epsilon} z_i$$

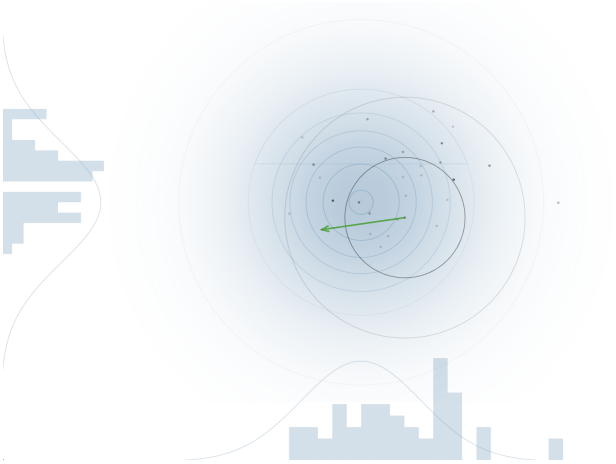
where $z_i \sim \mathcal{N}(0, I)$. Looking familiar yet? Let's suppose we conveniently have an overdamped physical system with potential energy function given by

[8] Often called the "evidence"

[9] Other "transitions" can be designed; the important thing is that our Markov transition preserves the target distribution. We recommend Betancourt [4] for a thorough introduction.

[10] For more background on score functions, we like these notes [12]

[11] This is an oversimplification, see [4] for formal introduction.



Visualization of Random Metropolis-Hastings, thanks to the fantastic website:
`https://chi-feng.github.io/mcmc-demo/app.html`

$U(x) = \log p(x)$. Then, the dynamics of the system are modeled by:

$$v = -\frac{1}{\lambda}\nabla_x U(x) + \frac{1}{\lambda}\eta(t)$$
$$v = -\frac{1}{\lambda}\nabla_x \log p(x) + \frac{1}{\lambda}\eta(t)$$

Intuitively, our certainty increases as we observe more data. And indeed, the language of information theory allows us to conclude that the entropy of a Bayesian posterior is, on average, less than or equal to the entropy of the prior. Physically realized, Bayesian inference can be seen quite literally as an "entropy pump", requiring work in order to reduce the entropy of a system while dissipating heat to its environment [2].

References

[1] M. Aifer, K. Donatella, M. H. Gordon, S. Duffield, T. Ahle, D. Simpson, G. E. Crooks, and P. J. Coles. Thermodynamic linear algebra, 2024.

[2] M. Aifer, S. Duffield, K. Donatella, D. Melanson, P. Klett, Z. Belateche, G. Crooks, A. J. Martinez, and P. J. Coles. Thermodynamic bayesian inference, 2024.

[3] S.-i. Amari. Neural learning in structured parameter spaces - natural riemannian gradient. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996.

[4] M. Betancourt. A conceptual introduction to hamiltonian monte carlo, 2018.

[5] K. Donatella, S. Duffield, M. Aifer, D. Melanson, G. Crooks, and P. J. Coles. Thermodynamic natural gradient descent, 2024.

[6] K. Donatella, S. Duffield, D. Melanson, M. Aifer, P. Klett, R. Salegame, Z. Belateche, G. Crooks, A. J. Martinez, and P. J. Coles. Scalable thermodynamic second-order optimization, 2025.

[7] M. Donsker. An invariance principle for certain probability limit theorems. *Memoirs of the American Mathematical Society*, (6), 1951.

[8] V. Gupta, T. Koren, and Y. Singer. Shampoo: Preconditioned stochastic tensor optimization, 2018.

[9] M. Johnson. Natural gradients and k-fac.

[10] S. Lalley and P. Mykland, 2012.

[11] E. Lifshitz and L. Landau. *Statistical Physics*. Pergamon Press, 1959.

[12] E. J. Ma. A pedagogical introduction to score models.

[13] J. Martens and R. Grosse. Optimizing neural networks with kronecker-factored approximate curvature, 2020.