Article

Machine Learning is just Statistical Mechanics with Better Marketing

by Vaibhav Kalvakota

Introduction

Both machine learning and statistical mechanics work with the complexity of high-dimensional spaces[1], emergent properties, and stochastic dynamics. Now of course, statistical mechanics studies the macroscopic behavior of systems composed of many interacting components using results from probability and statistics to bridge the gap between microscopic details and observable phenomena. In fact, it does this *so well* that most of the intuition to the workings of the macroscopic world are built on it from the fine-grained details.

Machine learning uses probabilistic tools to learn patterns in large datasets, optimizing over parameter spaces to make predictions or discover underlying structures, to carry out a particular task. There are many aspects of statistical mechanics, such as Boltzmann distributions, free energy minimization, and stochastic processes that lie deeply within the tools and algorithms that drive ML, including stochastic gradient descent, regularization, and probabilistic modeling.

(There are actually many other overlaps and applications of pure mathematics in machine learning as well, which we will not discuss.)

Boltzmann

One of the most fundamental correlations between the two subjects – of machine learning and statistical mechanics – is the obvious use of a number of information theoretic components in machine learning. When we seek to define a loss function $J(\theta)$ (where θ are the parameters of the model), we usually define it to be something like the *cross entropy loss function*,

$$J_{CE}(P,\, Q,\, \theta) = -\mathbb{E}_P\left(\log Q\right), \tag{II.1}$$

Vaibhav Kalvakota is a student in theoretical CS and studies the intersection of physics and ML. His interests include supervised learning, theoretical computer science and theoretical physics. Kalvakota's corresponding address is vaibhavkalvakota@gmail.com.

[1]By this, I want you to imagine a large parameter space and you want to find the optimal configuration in machine learning.

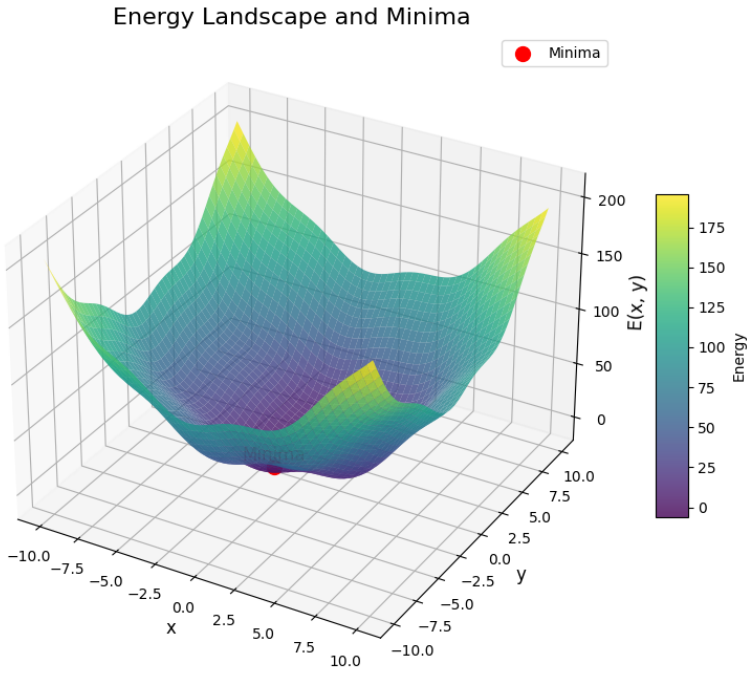

Fig: The minima is the little red dot in the landscape of different loss configurations. By Vaibhav Kalvakota

where P is the *true* distribution and Q is the *selected* probability distribution. $\mathbb{E}$ denotes the expectation value and this just becomes the usual form of $-P \log Q$. A loss function measures how "off" the model is from the actual probability distribution. Yes, yes, there's some measure theory stuff that happens when talking about this quantity rigorously. This is essentially measuring the Kullback-Leiblar divergence, but at the heart of it, all you need to worry about is that it tells you the "loss" of an observed distribution. Now what these loss functions also tell you is *how bad the other choices are*.

Imagine a huge parameter space. The problem in machine learning is to find the optimal configuration through this parameter space so that you have the least loss – best fit for the predictions – in the model. In the context of supervised learning[2], this is simply just making better predictions, and might involve some more tweaking with regularization/dropouts/etc.

[2]That is, when you are providing the outputs for the model to train on instead of just trying to find features in the inputs.

Figure-1 shows you the minima of the loss functions: the optimal state, so to speak, and this is *really hard* both computationally as well as theoretically sometimes, to reach. For most of the heavy machine learning models we use, the choice is that of *stochastic gradient descent* – a fancy way of using the extremum derivative rule for arbitrary "learning steps". This brings down the computational costs by a significant margin over normal gradient flow where you have to estimate with smaller strides. The price of using SGD over batch gradient descent is that you never really converge onto the minima; you just get really really close to it. As physicists, we obviously never complain about doing stuff like this.

In generative machine learning, more specifically a branch called ICA – **Independent Components Analysis**, you end up with a heavy reliance on *probabilistic models*. Simply put, given a probability distribution, you want to find the loss function minima as usual, but the way you do so isn't by wildly sampling[3] over the parameter space or training with more and more data; you instead view a *Boltzmannian approach* to machine learning. The slogan here is: *loss functions are no longer just minimas, they are energy functions, and you seek to minimize the energy.*

In a Boltzmann distribution, for some input x, you have an associated energy $E(x)$ and a probability given by

$$p(x) = \frac{\exp\left(-\beta E\right)}{Z} \ , \tag{11.2}$$

where Z is a normalization term. Of course, the physics people must have said *who are you kidding Vai that is a partition function, just call it that*, and yes it is a partition function[4] but bear with me for a second, will you?

The objective here is to minimize the energy loss function $E(x)$, but not with ordinary gradient descent. Instead, we want to use a *flow* $\mathcal{F}$ so that the action of $\mathcal{F}$ on the energy space[5] $\mathbf{E}$ is to make higher values of the energy function larger and the lower values smaller. So, you no longer just minimize by walking along the function, you stretch it out, by something like $\dot{x}(t) = -\nabla E(x)$, something like an inverse curvature flow.

This is something that has been done for quite a while in machine learning now, and Yann LeCun[6] has a very good paper on this. In working with energy loss functions instead of ordinary loss functions, we have to restrict to certain kinds of loss functions, but the most used loss function, the cross-entropy loss (11.2) is inherently a natural energy function.

[3] By this I mean taking steps in the parameter space to minimize the loss function.

[4] This field of research is called *canonical ensemble learning* where we seek to define partition functions and minimize the energy function.

[5] That is, a parametric space w.r.t the energy function.

[6] https://yann.lecun.com/exdb/publis/pdf/lecun-06.pdf

Attention Is All You Have

A natural sort of continuation from the above discussion is to talk about the partition function Z that I didn't want to specifically talk about. I still don't, because there are subtleties around actually computing it. Spoiler: you don't really consider it a "computable" quantity, but that is fine.

When you work with Transformer neural networks, you usually work with the softmax function by taking in the raw logits z_i and producing a weighted attention weight p_i:

$$\text{softmax}(z) = \frac{\exp z_i}{\sum \exp z_i} \ . \tag{11.3}$$

Look carefully at the denominator term. It is technically just a partition function, and this would generate "moments" akin to correlators like $\langle x_1,\ x_2,\ x_3 \ldots x_n \rangle$.

The first moment is the Helmholtz free energy $F = -\log Z$, which is a very interesting term. But there are more terms that also appear in working with these models, the most important of them being the *entropy* S. This is the usual Shannon entropy, but there are many useful things that come out of these two terms.

The entropy

$$S = -p_i \ \log p_i \tag{11.4}$$

calculates the uncertainty of the model. And LLMs are all about making that trade-off between very deterministic responses and overly diverse responses. This is a very simple task indeed; for even something as well trained as GPT-4o mini, you can end up with responses that will be too deterministic. The most natural way an LLM generates an output is by *greedy sampling*, where it simply picks the tokens with the highest post-softmax'd weights. However, the issue with this is that for out-of-distribution scenarios, the responses will be – trash. So models typically use *temperature* to make more diversified generations. There is a trade-off between strictly deterministic and highly diverse outputs that models require, and it was suggested early this year that there be dynamic temperature sampling using entropy. Which is really *really* interesting, and more recently, @xjdr started Entropix, which is (last I remember) a Llama 3.1 model with entropy sampling. I am, in fact, working on related things with entropy sampling for attention sparsity that could potentially make the complexity order of these models less than $O(n^2)$ or $O(n\sqrt{n})$ as is usually expected from adaptive sparsity. You could, alternatively, use the free energy, which technically captures more information than the entropy. In fact, sampling with free energy would work on two sides: one, it would focus on the lower energy functions subspace of $\mathbf{E}$ and could potentially sample a larger subset, and two, it would update the usual uncertainty metrics like entropy and variance-entropy. However, I am unaware of any models that use this yet.

Let me illustrate two machine-learning-in-physics topics that I think are really interesting. This is of course going to be somewhat more technical than what I talked above. [7]

Calabi-Yau Machine Learning

This is one of the most ambitious projects I have seen in quite some time in theoretical physics that has numerical calculations. Calabi-Yau manifolds are Ricci-flat objects[8] in string theory that have many applications in dimensional compactifications of extra dimensions.

Without going into too much technicalities, due to a high level of sophistication as well as my own lack of competence to phrase the technicalities in a readable format, the basic idea is just that Calabi-Yau manifolds are closed Kahler manifolds (these are complex manifolds with a hermitian metric). More specifically, a Calabi-Yau manifold is a compact Kahler manifold so that equivalently, (1) the first Chern class[9] vanishes, and (2) there exists a g that is Ricci flat. Another way of stating this is that there is a non-vanishing holomorphic n-form or has holonomy[10] in the special unitary group SU(n). These manifolds generally appear in string theory when working with dimensional compactifications, as in taking a description of a 10D $N = 1$ SUSY theory with a low-energy limit on a $D = 4$ manifold, with the remaining 6 dimensions reduced onto a Calabi-Yau manifold. There are certain topological invariants called *Hodge numbers*, which for some level of vague intuition, are related to the Euler characteristic of the manifold.

At a high level, we basically just want to be able to calculate these Hodge numbers. You can just think of this process as taking a list of pre-calculated complete-intersection Calabi-Yau manifolds data with the Hodge numbers and train a neural network using supervised learning and predict Hodge numbers simply, by taking a huge corpus of these Calabi-Yau data, it is possible to train a neural network that predicts the Hodge numbers very effectively. There are other Calabi-Yau calculations where neural networks help, see for instance a recent paper by Manki et al[11]. See also this paper[12] which has some excellent discussions on the string theory arena of machine learning.

Discrete Theory Space 101

The above discussion of CY manifold machine learning was straightforward to speculate on. However, in many cases, there are no *discrete* theory spaces for a particular constraint problem.

[7] To be fair, all I did was talk about energy based models.

[8] That is, the contracted Ricci tensor is zero, while the Riemann tensor need not be zero.

[9] https://en.wikipedia.org/wiki/Chern Class

[10] See for instance, https://ncatlab.org/nlab/show/holonomy

[11] Cristofero S. Fraser-Taliente, Thomas R. Harvey, and Manki Kim. 2024. Not So Flat Metrics. 11, https://arxiv.org/abs/2411.00962

[12] Andrei Constantin. 2022. Intelligent Explo- rations of the String Theory Landscape

My example of this would be the Wheeler-DeWitt equation,

$$H\Psi[g,\Phi] = 0\ , \tag{11.5}$$

where g is the metric and Φ are the matter fields on the manifold (M, g). States of this constraint are hard to solve unless you impose specific conditions such as asymptotic bulk limits, restriction to isometry groups, etc. and compose the Hilbert space of perturbative canonical quantum gravity when solved around perturbations. One could ask if there is a machine learning optimization task that could help us solve constraint (11.5). This is a technical problem but just consider the following. Take the decomposition of WDW states $\Psi[g,\Phi]$ into two "branches", Z^+ and Z^-. This happens because the Hamiltonian constraint is quadratic in nature, and usually there is one dominant branch. In any case, the collection of Z generate the so-called "theory space" and have a universal counterterm $S[g,\Phi]$ from holographic renormalization that applies to the entire theory space. This is good.

However, if you want to model neural networks that predict these counterterms, unlike the prediction of Hodge numbers, you end up with terms that are not "discrete", in the sense that numerically, it does not make sense to have a collection of universal counterterms that define a particular theory space simply because there aren't a discrete subset of these to begin with. You could seek to define other things that could be more numerically discrete, so that for specific cases you have specific values and then try to predict the values for the term in other cases. This goes on to just illustrate a level of obstruction for what can be computed and what cannot be, even if fundamentally they are just purely numeric coefficients.

In Phenomenology

Often in physics, we have to calculate physical couplings for theories. These are quantities that tell us a lot about the theory and the concretely observable properties of the theory as well. In particle collider experiments such as at the LHC (CMS or ATLAS), by computing the phenomenology of particle collisions, you can gain a lot of insight into the interaction between, say, partons. As a way of illustrating the role of machine learning in such phenomenological calculations, when working with QCD, it is very important to work with *parton distribution functions*, which are measured in partonic interaction experiments and give information about the cross-sections and interactions. The usual way of calculating or checking these distribution functions are either too complex or too time-taking, due to which large-scale computations become less feasible. In this paper[13], the collaborators trained a neural network to calculate the log-likelihood χ^2 from parton experiments at LHC. In fact, machine learning finds several roles in hadronic physics and pheno/experiment research, such as in working with meson production, heavy-ion collisions, and even beyond standard model particle interactions.

[13] DianYu Liu, ChuanLe Sun, and Jun Gao. 2022. Machine learning of log-likelihood functions in global analysis of parton distributions. JHEP, 08:088

Theory vs Experiment

Here I would like to draw a comparison that I think is very important. In high energy physics theory, there is an inherent distinction between theory and experiment. When I say that there is holography in an evaporating black hole spacetime, I typically mean that there are some concrete observables, but these aren't observables that you can actually calculate in real life as a part of an experiment.

For that matter, one of the last theory-meets-experiment timelines we had was closed around the time the CP violation was observed, because it was an observable phenomena. We do not have a way to calculate the amount of radiation collected from an evaporating black hole in anti-de Sitter space or the entropy of an island in AdS/CFT. This is the case with a lot of hep-th. The case with machine learning is the polar opposite; there are significantly good resources to actually check a theory. This has been known from the time we had RNNs and LSTMs, which still hold up nearly as well as some of the smaller-end base model Transformer architectures like BERT. In such cases, the line between theory and "experiment" must not be drawn, and the mathematical (often referred to as "pedantic") aspects that comprise machine learning should not be discarded. Since, after all, machine learning is essentially statistical mechanics with better marketing.

However, there is an important distinction between physics and the *applied* physics aspects of machine learning. By this, I mean that there are many things like Ising models, diffusion models, Langevin dynamics, phase transitions, etc. that are used in machine learning that arise from statistical mechanics. However, this does not count as doing *physics*. I say this because the 2024 Nobel Prize in physics was controversial and many (hep-th) academics questioned the principles on which this was given. However, if there is stringent applied physics being used in things like quantum computing or machine learning, this lies within the domain of "physics" in general.

Conclusion

In summary, there are a lot of interesting stuff to work on in machine learning and physics, and the overlaps between the two fields[14], and you should go check out the arXivs for hep-th and cs-LG (or stat-ML and cs-AI). And the so-called "this part of Twitter" (tpot) will illustrate the necessity of this article.

[14] There's also stuff with algebraic geometry in *singular learning theory*, which has some really exciting mathematical prospects. See *Sumio Watanabe. 2009. Algebraic Geometry and Statistical Learning Theory. Cambridge Monographs on ACM. Cambridge Press.*