Physical Learning Algorithms for Physical Networks

J. M. Schwarz

June 22, 2026

1 What is physical learning?

As I compose these *informal* notes from Santa Barbara, CA, while helping coordinate the KITP Brainless26 Workshop, I will assume that each of you has had some experience interacting with a Large Language Model, such as Claude. Indeed, we have all witnessed the Artificial Intelligence (AI) boom and now must grapple with how to effectively use AI to become better theoretical physicists. And if AI will help us solve humanity's problems, then perhaps we should understand how AI really works.

There is an ever-growing collection of physicists, including myself, realizing that if we are to understand how AI works then we must understand how the simplest systems, be it living or nonliving, learn. And by "simplest" in living systems, we do *not* mean the brain since brains developed quite a bit later in the evolution of life. In fact, one could argue that *all* biological systems are learning systems, all biological systems are physical systems, and so there must be a direct connection between physics and learning, hence, physical learning. It is, therefore, somewhat obvious why physicists are becoming increasingly interested in physical learning, beyond, of course, the initial brain-inspired work of Hopfield (and many others) [1].

If "all" biological systems learn, let's define a biological system as one with many interacting components (a network) and that is capable of modifying/updating, to greater or lesser extents, its own components and/or the interactions between components. A paradigmatic biological system example is a single cell. There are many components to, say, a eukaryotic cell, in terms of its cytoskeleton, its nucleus, its mitochondria, etc. And, for example, the cytoskeleton can change into different morphologies (more branched, less branched) as the cell interacts with its surroundings, i.e., it can modify itself. To some people's surprise, single cells *learn*. There is experimental evidence for this [2, 3, 4, 5]. For example, they habituate to harmless signals from the environment, meaning they learn to ignore harmless signals [4].

Before further rambling, let's define learning in the context of a task with inputs, outputs, and some measure of performance. Given input signals from the environment, learning is the modification of components/interactions in a system to produce an output that improves performance of not just some current task but also future-related tasks. Obviously, this learning construction does not distinguish between a living system and a nonliving system. However, while for living systems modification is typically self-induced, nonliving systems may need a little more help. In the habituation example above, the output is not responding

to the input and the learning is, for example, habituating to related harmless signals more quickly.

Now that we have some idea about learning, as opposed to working with more complex systems such a biological cell with its numerous interacting subsystems in terms of mechanical, metabolic, chemical, etc., let us turn to simpler interacting systems, namely flow networks, resistor networks, and spring networks. While many of us have thought about such systems prior to coming to Boulder, Eleni's notes make the linear response of all three systems clear. Thank you, Eleni!

To probe questions about learning in such networks, we need to elevate them to a system that can be modified. What do I mean by that? For any network with fixed architecture, modifications will be changes to interaction strengths, such as the resistance of each resistor. These interaction strengths act much like updatable weights in a neural network. For any network without fixed architecture, rewiring of the resistor network, is also a possible modification (but one that conventional AI platforms do not seem to consider). In biology, again, such modifications are done by the system itself (no supervisor required), while for nonliving systems the modifications will, for the most part, be done by a computer or by a person, i.e., a supervisor. Some sort of physical learning algorithm will be needed to quantify the amount of modification. We will now focus on several such algorithms with the first two learning class meets focusing on networks of fixed architecture (so no rewiring) but modifiable resistance, for example, and the third learning class meet will address a system in which the architecture is restructured.

Please note that some additional related references on physical learning will be provided later on in the notes. The vast majority of the following was done in collaboration with a number of people, including V. Anisetti, B. Scellier, A. Kandala, T. Zhang, S. Ameen, K. Kim, O. Pfister, J. Paulsen, N. Pashine, M. Stern, I. Klich, A. Amanuel, W. Guo, and K. Zhang. Also notice that there are no figures in these notes. There definitely will be figures during the class meet, so please do not think of these notes as a replacement for the class meet, but as a supplement. Lastly, each of the following three sections has a notation unto itself. Apologies.

2 Learning Class Meet I: Multi-mechanism Learning with fixed architecture weighted Laplacian networks

2.1 Driving question

Let's cut right to the chase and ask:

How can a physical network learn a task using gradient descent?

2.2 Set-up of the weighted Laplacian network

To begin to answer our question, consider a weighted graph with N nodes labeled $1, \ldots, N$. Each edge connecting nodes x and y has a symmetric weight $w_{xy} = w_{yx}$ and $w_{xx} = 0$ (no self-loop), i.e., there is a set of weights. It is these weights that will be modified

with the network architecture fixed. Note that x and y are node indices and not spatial indices, fortunately or unfortunately. The weights, for example, represent conductances, if we are considering a resistor network. Then, let v_x denote the voltage at node x. Let us choose M input pairs of nodes by which applied input electric currents flow into and out of the network. Different applied input currents for different input node pairs are possible. The input currents form a list of numbers, or $\vec{I} \in \mathbb{R}^n$. Current conservation at each node x (using Ohm's law) gives

$$\sum_{y \, d.c. \, x} w_{xy} \left(v_x - v_y \right) = I_x, \tag{1}$$

where *d.c.* denotes the nodes y directly connected to node x. This equation can be written in matrix form as

$$\hat{L}\vec{v} = \vec{I}, \tag{2}$$

as the voltages at each node form a list of numbers as well. Here, $\hat{L}$ is the weighted graph Laplacian:

$$L_{xx} = \sum_{y \, d.c. \, x} w_{xy}, \qquad L_{xy} = -w_{xy} \; (x \neq y), \tag{3}$$

so $\hat{L}$ is symmetric. For all nodes not directly connected, the weighted graph Laplacian contribution is zero.

For the sake of argument, we will also consider an identical twin resistor network in which the architecture and weights are the same. In this identical twin resistor network, we define pairs of output nodes in which feedback currents are injected into and flow out of the identical twin network. The feedback currents are, again, a list of numbers denoted by $\vec{E} \in \mathbb{R}^n$. When feedback currents are injected into the network, the response is governed by the same weighted Laplacian operator, or

$$\hat{L}\vec{u} = \vec{E}, \tag{4}$$

with feedback voltages at each node denoted by u_x.

If you are scratching your head about the introduction of an identical twin network, you have every right to be. Certainly biology is not set up that way. So instead, you are welcome to also consider one *multi-plexed* network where the response to the input currents is a resistor network. However, these resistors also act as conduits/tubes such that the response to the feedback currents are measured in terms of pressures. Given this multi-plexed network, the weights of the feedforward network with its input currents and the weights of the feedback network with its feedback currents should take on the same values, which may or may not be easy to do given that the weights represent different quantities. We can discuss this further in person. The important takeaway is that u_x and v_x at each node must be *distinguishable*.

Finally, while voltage and pressure are scalar quantities, displacement is a vector quantity. To quantify spring networks, one can group each spatial component together to form a block weighted Laplacian.

2.3 A Task to Learn

There are many different tasks one can assign either the identical twin networks or the multi-plexed network to learn, such as classification. To navigate the task, a cost function

C is defined as

$$C = \frac{1}{2}\sum_{o.p.}(v(x,y) - v_d(x,y))^2, \tag{5}$$

where *o.p.* denotes output pairs and voltage drops are defined via

$$v(x,y) = v_x - v_y, \qquad u(x,y) = u_x - u_y. \tag{6}$$

Achieving the task becomes synonymous with minimization of the cost function. Gradient descent is a means by which to achieve this minimization.

To be more specific, the first extremely, extremely simple AI task one can try is Iris classification [6]. This is a very old data set consisting of only numbers (and not images of, say, handwritten digits from the MNIST data set or images of cars, airplanes, cats, frogs, and six other types of creatures or objects, etc. from the CIFAR-10 dataset. There are 150 irises of three different types (50 of each). There are four inputs consisting of petal width and length and sepal width and length. The data set is split in half; half for training and half for testing so that we can test for generalization (as opposed to memorization).

Finally, as we stated above the learning involves a system either getting modified (via a supervisor) or modifying itself (no supervisor/unsupervised). The trainable/updatable degrees of freedom are the resistances/the tube diameters/the spring constants of the network(s). The physical degrees of freedom are the voltages/pressures/strains.

2.4 Multi-mechanism (Multi-modal) Learning Overview

Now that we have a system with many interacting components that can be modified and a task, let us turn to what is called the Multi-mechanism Learning (MmL) algorithm to minimize C [7]. Here is its pseudocode tailored to the Iris classification task.
Step 1: First give input currents for the four features and determine the voltage (v_x) at output node pairs. There will be 3 pairs of output nodes.
Step 2: Determine the feedback currents at the output nodes in the identical twin network via

$$E_x = -\eta\frac{\partial C}{\partial v_x}, \tag{7}$$

where $\eta > 0$ is a scalar that sets the feedback amplitude.
Step 3: Apply the feedback currents to the output nodes in the identical twin network and compute the voltage (u_x) at the output node pairs.
Step 4: The following learning rule updates each conductance *locally* as

$$\Delta w_{xy} = -\alpha\, u(x,y)\, v(x,y), \tag{8}$$

with $\alpha > 0$ a learning-rate factor.
Step 5: Steps 1-4 are iterated until a dimensionless version of the cost function C decreases below some threshold value that is much less than one.
Note that though there may also be some pre-processing of the inputs and post-processing of the outputs that we can discuss in class.

2.5 Motivating the form of the update rule (after the fact)

Let's demonstrate that this particular update rule performs gradient descent on C, i.e.,

$$\Delta w_{xy} = -\alpha\eta \frac{\partial C}{\partial w_{xy}}. \tag{9}$$

Because $\vec{v}$ depends on the weights through Eq. (2), the chain rule gives for each edge weight w_{xy},

$$\frac{\partial C}{\partial w_{xy}} = \left(\frac{\partial C}{\partial \vec{v}}\right)^{\top} \frac{\partial \vec{v}}{\partial w_{xy}}. \tag{10}$$

Multiplying both sides by $-\eta$ and using Eq. (7), we arrive at

$$-\eta \frac{\partial C}{\partial w_{xy}} = \vec{E}^{\top} \frac{\partial \vec{v}}{\partial w_{xy}}. \tag{11}$$

Given Eq. (4), i.e., we have

$$-\eta \frac{\partial C}{\partial w_{xy}} = (\vec{L}\vec{u})^{\top} \frac{\partial \vec{v}}{\partial w_{xy}} = \vec{u}^{\top} \hat{L}^{\top} \frac{\partial \vec{v}}{\partial w_{xy}} = \vec{u}^{\top} \hat{L} \frac{\partial \vec{v}}{\partial w_{xy}} \tag{12}$$

since $\hat{L}^{\top} = \hat{L}$.

Next, differentiate the feedforward response equation (Eq. (2)) with respect to w_{xy}, or

$$\frac{\partial \hat{L}}{\partial w_{xy}} \vec{v} + \hat{L} \frac{\partial \vec{v}}{\partial w_{xy}} = \vec{0} \tag{13}$$

to yield

$$\hat{L} \frac{\partial \vec{v}}{\partial w_{xy}} = -\frac{\partial \hat{L}}{\partial w_{xy}} \vec{v}. \tag{14}$$

Substituting Eq. (14) into Eq. (12) leads to

$$-\eta \frac{\partial C}{\partial w_{xy}} = \vec{u}^{\top} \left(-\frac{\partial \hat{L}}{\partial w_{xy}} \vec{v}\right) = -\vec{u}^{\top} \frac{\partial \hat{L}}{\partial w_{xy}} \vec{v}, \tag{15}$$

or

$$\eta \frac{\partial C}{\partial w_{xy}} = \vec{u}^{\top} \frac{\partial \hat{L}}{\partial w_{xy}} \vec{v}. \tag{16}$$

From the definition of the weighted Laplacian (Eq. (3)), changing w_{xy} affects only four entries of $\hat{L}$:

$$\frac{\partial L_{xx}}{\partial w_{xy}} = 1, \quad \frac{\partial L_{yy}}{\partial w_{xy}} = 1, \quad \frac{\partial L_{xy}}{\partial w_{xy}} = -1, \quad \frac{\partial L_{yx}}{\partial w_{xy}} = -1, \tag{17}$$

and all other entries are zero. Therefore,

$$\begin{aligned} \vec{u}^{\top} \frac{\partial \hat{L}}{\partial w_{xy}} \vec{v} &= u_x v_x + u_y v_y - u_x v_y - u_y v_x \\ &= (u_x - u_y)(v_x - v_y) \\ &= u(x,y)\, v(x,y). \end{aligned} \tag{18}$$

Combining Eqs. (16) and (18) leads to

$$\eta \frac{\partial C}{\partial w_{xy}} = u(x,y)\, v(x,y). \tag{19}$$

Finally,

$$-\alpha\eta \frac{\partial C}{\partial w_{xy}} = -\alpha\, u(x,y)\, v(x,y) \tag{20}$$

matches the proposed update rule given in Eq. (8). Hence,

$$\Delta w_{xy} = -\alpha\eta\, \frac{\partial C}{\partial w_{xy}}, \tag{21}$$

and each step of the algorithm performs one step of gradient descent with effective step size $\alpha\eta$.

2.6 Performance, comments on MmL and related algorithms

To see how well MmL does with learning the Iris classification, please see Ref. [7]. For this simple linear physical network, the best testing accuracy is approximately 93%, or an A student, but definitely not an A+ student. Our physical network is linear. Nonlinearity is encoded in the weight update rule via a product of two responses, which is the simplest nonlinearity one can construct. There are ways to improve the performance.

This algorithm is one of distributed learning in the sense that the update rule is a local one. The update uses only edge-local quantities $u(x,y)$ and $v(x,y)$. This property is rather different for the update rule used in backpropagation, whose update rule is nonlocal (as we discussed in Learning Class Meet II with a very simple example). Moreover, the construction relies on $\hat{L}$ being symmetric (undirected network, $w_{xy} = w_{yx}$). In the multi-plexed network, one needs multiple mechanisms/modes of information transmission, or response, such that both the feedforward field v and the feedback field u can be identified.

Finally, the inspiration for MmL was provided by a related algorithm known as Equilibrium Propagation developed back in 2017 by Scellier and Bengio [8, 9]. Equilibrium Propagation (EP) compares the free equilibrium state of a physical system with a nearby weakly nudged equilibrium state. In the small-nudging limit, the resulting dynamics can be shown to follow the gradient flow of a cost function. Thus, although the physical system itself never explicitly computes gradients, the measured physical contrasts are mathematically equivalent to gradient descent. It should be noted that Coupled Learning (CL) developed in 2021 by Stern, Hexner, Rocks, and Liu closely resembles EP by comparing nearby constrained physical states [10]. Please see also work from one of our amazing organizers, Xiaoming, on very related learning in spring networks [11].

Recent theoretical work has shown that the updates of CL are not, in general, exactly, but approximately, gradient descent [12]. This distinction suggests that physical learning algorithms need not be viewed solely through the lens of gradient descent. Rather, the underlying physical contrasts themselves are the primary objects from which learning rules emerge. MmL can also be viewed as a contrastive learning algorithm as we shall see in the next learning class meet.

A quick check on the explanatory power (or lack thereof) of these notes: Rather than a multi-plexed network or identical twin networks, please think about constructing another version of MmL involving a single network but using time as a way to distinguish the feedforward and feedback responses.

3 Learning Class Meet II: Perturbative Contrastive Physical Learning

3.1 Driving Question

With Multi-mechanism Learning, physical processes on a physical network can transpire, if you will, to produce parameter updates equivalent to gradient descent. Of course, a supervisor is needed to compute the amount of modification and enact the update. However, we are working towards autonomous physical learning networks with other physical processes performing the multiplication in the update rule. In fact, please see Ref. [13] where there is a discussion of training a particular type of circuit to perform the multiplication. Please recall that the MmL algorithm never explicitly computed the gradient of the cost function. Rather, the gradient emerged implicitly through the comparison of two physical responses. This observation raises a more fundamental question:

Can measurable physical contrasts themselves be the primary objects from which learning algorithms should be constructed?

In conventional machine learning, optimization with gradient descent is used. It assumes that gradients can either be computed analytically. Physical systems, however, can be rather complex and so we do not even know the governing physics equations. And yet, physical systems reveal information through their *response to perturbations*. A small change in an applied force, boundary condition, or internal/hidden parameter may produce a measurable change in the observable state of the system. Physics is largely built upon this simple observation: perturb a system and measure how it responds. *The central idea of Perturbative Contrastive Physical Learning (PCPL) is to elevate this physics principle into a learning principle* [13]. Rather than computing derivatives symbolically, we compare nearby physical states generated by controlled perturbations. These measurable contrasts provide sufficient information to guide parameter updates. Learning, therefore, becomes a process of probing, measuring, and remodeling/modifying a physical system according to its own accessible response. This viewpoint generalizes MmL, EP, CL, and *in situ backpropagation*. Such algorithms rely upon the same fundamental principle: useful learning information is contained in measurable differences between nearby physical states. PCPL places this contrast principle at the center of the learning process. In what follows for the rest of this section, you are welcome to also check out Ref. [13] for more information.

3.2 Physical response maps

Let's go beyond a weighted Laplacian network and consider a physical system described by a response map

$$z = G(y, \theta), \tag{22}$$

where y denotes some encoded input, θ represents the modifiable/updatable physical parameters (formerly known as w_{xy} in the previous learning class meet), z denotes the measured physical response. The encoding of the raw data may itself involve a fixed preprocessing stage $y = F(x)$, where x denotes the original data. Please note that x and y denote different quantities from the last learning class meet. Perhaps I should have labelled in the encoded input apple and the output orange since they are relatively easy to draw. Apologies. Also note that unless otherwise stated, y and z and x should be interpreted as vectors of inputs (which were input currents I in the previous section) and measured outputs whenever the physical system has multiple inputs or outputs. Moreover, θ also forms a list. In other words, the notation has been streamlined. Finally, throughout this section of the notes, we assume that the encoding map F remains fixed, while learning modifies only the response function G. *Importantly, G is not required to be linear.*

As discussed in Learning Class Meet I, the objective of learning is to determine an update

$$\theta \to \theta + \Delta\theta \tag{23}$$

(formerly known as w in streamlined notation) that improves the performance on a prescribed task. The parameter update is inferred from measurable contrasts between nearby physical responses,

$$\Delta\theta \sim M(z_1, z_2), \tag{24}$$

where M denotes a measurement operator that converts observable contrasts into parameter updates. The measurement operator depends upon the physical implementation. It may involve differences between equilibrium states, differences between oscillatory responses, measured displacements, optical quadratures, or other experimentally accessible observables. The important point is that M acts on measurable physical quantities rather than symbolic derivatives. So we ask, which nearby physical states should be compared? This simple question naturally gives rise to two distinct learning geometries.

3.3 Two learning geometries

Here are two complementary modes of PCPL, namely

Mode A. The perturbed response is compared with the system's own free response. Learning therefore probes the local sensitivity of the system and can be interpreted as measuring a Jacobian action.

Mode B. The perturbed response is compared with an externally specified target. Learning therefore solves an inverse problem whose geometry is determined by the physical response of the system itself.

3.3.1 Mode A: Learning Through Local Response

Linear response theory occupies a central role throughout physics. Mode A adopts precisely this philosophy for learning. To be more specific, the system is first allowed to evolve without any perturbation, producing the *free response*,

$$z_f = G(y, \theta). \tag{25}$$

Next, the system is perturbed slightly, $\theta \to \theta + \delta\theta$, yielding the perturbed response

$$z_p = G(y, \theta + \delta\theta). \tag{26}$$

The measurable contrast is, therefore,

$$\Delta z = z_p - z_f. \tag{27}$$

No gradients have been computed. No adjoint equations have been solved. Only two physical measurements have been performed. Provided the perturbation remains sufficiently small, then

$$G(y, \theta + \delta\theta) = G(y, \theta) + J_\theta \delta\theta + O(\delta\theta^2), \tag{28}$$

where

$$J_\theta = \frac{\partial G}{\partial \theta} \tag{29}$$

is the Jacobian of the response. Finally,

$$\Delta z \approx J_\theta \delta\theta. \tag{30}$$

This equation is the central result of Mode A. The measured contrast is nothing more than the Jacobian acting upon the imposed perturbation. The measured contrast must be converted into a parameter update using the measurement operator defined above with $z_1 = z_f$ and $z_2 = z_p$, which extracts the information required for learning. Formally, the update becomes

$$\Delta\theta = \eta M(z_p, z_f), \tag{31}$$

where η denotes the learning rate. Notice that the target itself does not appear explicitly during the perturbation step. Instead, the desired task enters implicitly through the construction of the measurement operator M.

To help make Mode A more concrete, let's recast MmL as Mode A learning. Specifically, the two physical states being compared are

$$z_f \equiv v, \qquad z_p \equiv u, \tag{32}$$

where one construction is identical to the prior construction in Section 2 (with stream-lined notation). The measurement operator is simply

$$[M(z_p, z_f)]_{xy} = -u(x, y)v(x, y). \tag{33}$$

Consequently,

$$\Delta w_{xy} = \eta [M(z_p, z_f)]_{xy}. \tag{34}$$

Indeed, MmL measures two nearby physical responses and combines them through a local measurement operator. Gradient descent emerges only after mathematical analysis of the physical measurements. Here, the perturbed response is the feedback current applied to the identical twin network, for example. And although Mode A does not explicitly compute gradients during learning, the construction of an effective measurement map is itself generally a nontrivial task. In the case of Multi-mechanism Learning (Section 2), the measurement map was designed and subsequently shown to produce exact gradient descent. Thus, gradients entered the theoretical development of the algorithm, even though the physical system executing the algorithm never computes them explicitly.

A quick check on the explanatory power (or lack thereof) of these notes: Please construct the pseudocode for Mode A.

3.3.2 Mode B: Learning Through Inversion

Suppose we already know the response that we would like the physical system to produce. Rather than probing its sensitivity, Mode B solves an inverse problem. Learning proceeds by determining which parameter changes most effectively reduce the discrepancy between the current physical response and a desired target. As before, the measured instantaneous response is z, while the target response is z^*. The instantaneous error e is, therefore,

$$e = z^* - z. \tag{35}$$

Our objective is to determine a parameter update that reduces this error. To begin to achieve this objective, let's again assume that the parameter update remains sufficiently small, so that $G(y, \theta + \delta\theta) = G(y, \theta) + J_\theta \delta\theta + O(\delta\theta^2)$, then $z^* \approx z + J_\theta \delta\theta$. Once J_θ has been estimated, we seek a generally different parameter change whose linearized response reduces the output error. Requiring the updated response to approach the target gives

$$e \approx J_\theta \Delta\theta. \tag{36}$$

In general, this equation cannot be solved exactly. So instead, we seek the parameter update $\Delta\theta$ that minimizes the regularized objective

$$\min_{\Delta\theta} \left\| J\,\Delta\theta - e \right\|^2 + \lambda \left\| \Delta\theta \right\|^2, \tag{37}$$

where the second term $\lambda \left\| \Delta\theta \right\|^2$ penalizes large parameter updates and the scalar $\lambda > 0$ controls the strength of regularization. Taking the derivative with respect to $\Delta\theta$ and setting it to zero yields

$$\begin{aligned} &\frac{\partial}{\partial\,\Delta\theta}\left[\left\| J\,\Delta\theta - e \right\|^2 + \lambda \left\| \Delta\theta \right\|^2 \right] \\ &= 2J^\top (J\,\Delta\theta - e) + 2\lambda\,\Delta\theta = 0, \end{aligned} \tag{38}$$

which gives $(J^\top J + \lambda I)\,\Delta\theta = J^\top e$. The closed-form solution is, thus, the regularized pseudoinverse step

$$\Delta\theta = \left(J^\top J + \lambda I \right)^{-1} J^\top e. \tag{39}$$

Note that this is the Newton-Gauss method for optimization with a Tikhonov regularization, which is a standard method for those solving inverse problems. Please search for references. Here, we have shown that linear response theory from physics naturally leads to such a method for physical learning. Also note that if the Jacobian cannot be computed analytically, particularly when we do not even know the physics governing the system, then one can approximate entries in the Jacobian with finite differences.

The appearance of $J_\theta^T J_\theta$ has an important geometric interpretation. Gradient descent assumes that all directions in parameter space are equally important. Consequently, parameter updates follow the Euclidean geometry of the parameter space. Mode B behaves differently since directions in parameter space that strongly influence the measured response naturally receive larger corrections than directions to which the system is relatively insensitive. Learning, therefore, follows the intrinsic geometry of the physical system rather than the Euclidean geometry of the parameters.

A quick check on the explanatory power (or lack thereof) of these notes: Please construct the pseudocode for Mode B.

To illustrate Mode B in a super simple setting, consider a resistor network with two trainable conductances, $\theta_1 = g_1$ and $\theta_2 = g_2$, connect two input voltages, v_1 and v_2, to a single output node of voltage v_3. The output node is connected to ground through a fixed conductance g_3 (since we are now using the good ole physics notation for conductance). Applying Kirchhoff's current law at the output node gives

$$g_1(v_3 - v_1) + g_2(v_3 - v_2) + g_3 v_3 = 0. \tag{40}$$

Solving for the output voltage yields

$$v_3 = \frac{g_1 v_1 + g_2 v_2}{g_1 + g_2 + g_3}, \tag{41}$$

To train on multiple input examples, consider for the input voltages $v_1^{(1)} = 1$, $v_2^{(1)} = 0$ and $v_1^{(2)} = 0$, $v_2^{(2)} = 1$ with the respective desired outputs $v_*^{(1)} = \frac{1}{2}$ and $v_*^{(2)} = \frac{1}{4}$. The output vector is initially therefore

$$v_3(g) = \begin{pmatrix} \dfrac{g_1}{g_1 + g_2 + g_3} \\ \dfrac{g_2}{g_1 + g_2 + g_3} \end{pmatrix}, \tag{42}$$

and the error vector becomes $e(g) = v_3(g) - v_*$ (again using stream-lined notation). Defining $S = g_1 + g_2 + g_3$ and differentiating the output voltage with respect to either conductance gives the compact expression

$$\frac{\partial v_3}{\partial g_j} = \frac{v_j - v_3}{S}, \tag{43}$$

with $j = 1, 2$. Evaluating these derivatives for each training example produces the Jacobian

$$J = \begin{pmatrix} \dfrac{\partial v_3^{(1)}}{\partial g_1} & \dfrac{\partial v_3^{(1)}}{\partial g_2} \\ \dfrac{\partial v_3^{(2)}}{\partial g_1} & \dfrac{\partial v_3^{(2)}}{\partial g_2} \end{pmatrix}. \tag{44}$$

Assume all three conductances are set to unity, then $S = 3$ and $v_3^{(1)} = v_3^{(2)} = \frac{1}{3}$, and

$$J = \frac{1}{9}\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}. \tag{45}$$

The initial error is

$$e = \begin{pmatrix} -\frac{1}{6} \\ \frac{1}{12} \end{pmatrix}. \tag{46}$$

Using the update rule for Mode B without any regularization leads to

$$\Delta g = \begin{pmatrix} \frac{3}{4} \\ 0 \end{pmatrix}. \tag{47}$$

Thus,

$$g_1^{\text{new}} = \frac{7}{4}, \tag{48}$$

$$g_2^{\text{new}} = 1. \tag{49}$$

Substituting these updated/"new" conductances into the circuit equations gives

$$v_{\text{new}} = \begin{pmatrix} \frac{7}{15} \\ \frac{4}{15} \end{pmatrix} \approx \begin{pmatrix} 0.467 \\ 0.267 \end{pmatrix}, \tag{50}$$

which is substantially closer to the desired target. Of course, the target is not reached exactly because the Gauss–Newton step solves only the linearized problem. Repeating the procedure with the updated conductances would produce successive approximations to the nonlinear optimum.

This very simple training example serves only as a pedagogical example. However, it readily demonstrates when learning is not effective, such as when $v_3 \approx v_1$, for example. On the other hand, imagine being handed a *mysterious* resistor network. The governing physics may be unknown, or it may be perfectly understood but too complicated to differentiate analytically. Regardless, Mode B requires only the ability to probe the system by introducing small parameter perturbations and measuring the resulting changes in the outputs. From these measurements, we estimate the local Jacobian and compute the same Gauss–Newton update as before.

More importantly, because the desired task has been specified explicitly from the outset, the estimated Jacobian does more than guide parameter updates. It characterizes the local learnability of the physical system. When training succeeds, the Jacobian identifies the parameter combinations responsible for improving performance. When training stalls, it can reveal whether the limitation arises from weak parameter sensitivities, poor conditioning, or intrinsic constraints of the physical system. In this sense, Mode B is simultaneously an optimization algorithm and a tool for probing what a physical system can and cannot learn.

Although both Mode A and Mode B seek to reduce the output error and both may ultimately succeed or fail depending on the capabilities of the underlying physical system, they differ fundamentally in how the learning task is represented. In Mode A, the learning objective is encoded implicitly within the physics through paired system states. The material itself generates a local learning signal without explicitly constructing either a Jacobian or a global model of its input–output behavior. In contrast, Mode B begins with an explicit specification of the desired task, estimates the parameter-to-output map through controlled perturbations, and computes a parameter update. Viewed together, the two modes provide complementary perspectives on physical learning: one demonstrates how learning can emerge from the material itself, while the other systematically explores what that material is capable of learning.

A quick check on the explanatory power (or lack thereof) of these notes: Please list some systems that may be more amenable to either Mode A or Mode B.

4 Learning Class Meet III: Can sand learn?

4.1 Driving question

Perhaps the above question caught your attention (or perhaps not). Why would someone (such as myself) even pose such a question? Perhaps it is because I have moved on from Santa Barbara, CA, with its *sandy* beaches, to camping out amongst the Redwoods and the Great Basin National Park, with its wonderfully starry night sky (since it is in the middle of nowhere for mainland USA), and now camping in desert terrain, with, well *more sand*. Sand, or more generally, interacting particle systems, have been known to exhibit interesting properties too numerous to discuss here, other than stating it can jam (or become rigid as Ileana has discussed), it can flow [14], it can store memories [15]. Probably something that is less known is that sand can act as a *finite-state machine* in certain conditions [16].

A finite-state machine (FSM) is made up of a finite set of states, imagine a set of M spins each with two states, a transition table between the sets of states, and an initialization. Start the FSM and let it run. One of the simplest example of a FSM is the turnstile (according to Wikipedia, (yes, Wikipedia)). A turnstile acts as a finite-state machine because it occupies one of a finite number of states (locked or unlocked) and transitions between these states only in response to specific inputs, such as inserting a coin or pushing the gate. At any given time, the behavior of the turnstile is completely determined by its current state and the received input. The transition rules are fixed, so the same input always produces the same response when applied to the same state.

How can a class of interacting particle systems act as a FSM you ask? Both experimentally and numerically, researchers have found that an interacting particle system (such as colloids at an oil-water interface), in response to cyclic shear exhibits localized particle rearrangements known as hysterons [16]. They are hysterons in the sense that the shear amplitude to trigger the particle rearrangement in one direction is different from the shear amplitude to trigger the particle rearrangement to go back to the "initial" particle configuration. So there are two states, of the localized particle rearrangement, and there is hysteresis in going from one state to the other. Interestingly, or perhaps, obviously, there are inter-

actions between these as mediated by the other particles not participating in the localized rearrangements.

This interacting hysteron system can be mapped to a FSM [17]. We can sketch out a quick example in class. The point of raising the FSM construction is to emphasize that "sand" can perform nontrivial tasks, just as a turnstile has its own task to not allow people through the gate if they do not pay, then perhaps we can train sand to perform nontrivial tasks as well. If the interactions between the hysterons can be updated in some way, then we perhaps can indeed train the system. Only now, as opposed to Learning Class Meet I, where training a fixed architecture system involved updating weights in terms of conductances, here, let's not update interaction strengths between particles. Together, let's take a leap. Let's try to train a system that does not have a fixed architecture and let's use the changing of the architecture as the training task itself. Again, we will not change any interaction strengths or particle sizes in the particle packing. The updatable parameters are the positions of the particles themselves, or what some would call the physical degrees of freedom.

With this background in mind, let's reframe the question opening this section as:

Can one train a rigid, but reconfigurable, particle packing to learn associative tasks?

4.2 Mechanical training for associative tasks

We will look at a few slides to illustrate training "sand" for a particular range of associative tasks, associative meaning an output is associated with an input [18], just as Pavlov trained his dogs to salivate at the sound of the bell because the dogs associated the bell with dinner [19]. For those who have heard of protein allostery, there is work on modifying a spring network, which is a placeholder for a protein, such that the strain on an output edge is associated with a particular strain on an input edge [20, 21, 22]. Let's turn to some slides to see how a particle packing, with the particles interacting via the Lennard-Jones potential, responds to be trained for an associative task. Hope you like the slides! For even more details, please see Ref. [18].

4.3 Toy model for particle packing training: History-Modifying Mechanical Finite-State Machines

Now that we have seen our best performance to date is that of a B- student, we (Tao Zhang and I and others) are continuing to play with the training algorithm to try to obtain better performance. On the other hand, let's see if we can construct a toy model to better understand fundamental limits on this type of training. Why could we not do better than a B-student? Moreover, unlike the previous two learning class meets, here, we do not yet know the explicit update rule, so perhaps a toy model invoking an update rule may be useful, so that we can begin to determine what types of learning rules are possible.

The idea here is to view a mechanically trained material, such as "sand", as an effective *History-Modifying Mechanical Finite-State Machine* (HM-MFSM). The HM-MFSM is not intended as a microscopic description of the material, but rather as a coarse-grained description of transitions between metastable mechanical states. We define the HM-MFSM

as

$$\mathcal{M}(\mathcal{H}) = (Q(\mathcal{H}), \Sigma, \delta_{\mathcal{H}}), \tag{51}$$

where $Q(\mathcal{H})$ denotes the set of metastable mechanical states, Σ denotes the family of loading protocols,$\delta_{\mathcal{H}}$ represents the history-dependent transition law, and $\mathcal{H}$ denotes the loading history. The defining feature of the HM-MFSM is

$$\delta = \delta(\mathcal{H}), \tag{52}$$

and, more generally,

$$Q = Q(\mathcal{H}). \tag{53}$$

Thus, both the available metastable states and the transition law may evolve under repeated loading. Unlike conventional FSMs, the transition function is not fixed. And unlike MmL or PCPL, there is no explicit optimization, nor is any Hebbian update rule assumed, which has helped drive neural network learning. In fact, the HM-MFSM framework does not assume any particular local learning rule. Instead, driving history modifies the energy landscape, one determines how metastable states evolve, and the HM-MFSM records the resulting transition structure.

Since the HM-MFSM framework provides only a coarse-grained description of how history modifies transitions between metastable mechanical states, to make more progress, we can attempt to specify the microscopic origin of the evolving transition law. Let's do so by considering a simple quasi-static toy model motivated by the idea that repeated cooperative rearrangements modify the local mechanical environment, thereby changing future rearrangements. The purpose of the following calculation is to illustrate one possible mechanism by which associative behavior can emerge from history-dependent evolution of the energy landscape.

Consider two localized rearrangement motifs, R_1 and R_2, each of which may either activate or remain inactive during a loading cycle. Let's introduce binary variables $s_i \in \{0, 1\}$, where $s_i = 1$ indicates that motif R_i rearranges. Unlike a neural-network model, we do not assume that the rearrangements directly modify pairwise couplings between rearrangements. Instead, repeated cooperative rearrangements slowly modify a structural variable, such as residual stress, or another coarse-grained measure of the evolving mechanical environment. We denote this slow variable by $m_{12} \geq 0$. Following each loading cycle,

$$m_{12}^{(n+1)} = (1 - \beta) m_{12}^{(n)} + \gamma s_1^{(n)} s_2^{(n)}, \tag{54}$$

where γ measures the structural change produced by a cooperative rearrangement and β represents slow relaxation or forgetting. Thus, isolated rearrangements do not significantly modify the local structure, whereas simultaneous rearrangements produce persistent structural evolution.

Suppose motif R_2 possesses a quasi-static activation threshold θ_2^0. History modifies this threshold according to

$$\theta_2^{(n)} = \theta_2^0 - \xi m_{12}^{(n)}, \tag{55}$$

where $\xi > 0$ quantifies how structural evolution changes local stability. Given an applied loading amplitude Γ, the activation rule is

$$s_2^{(n)} = \Theta\left(\Gamma - \theta_2^{(n)}\right). \tag{56}$$

In other words, the structural memory may indeed modify the local energy landscape associated with motif R_2. More precisely, whenever neighboring motifs rearrange cooperatively, i.e. $s_1 = s_2 = 1$, the structural memory changes. For repeated cooperative events,

$$m_{12}^{(N)} = \frac{\gamma}{\beta}\left[1 - (1-\beta)^N\right]. \tag{57}$$

As training proceeds, m_{12} changes the geometry of the local energy landscape, thereby shifting $\Gamma_c(m_{12})$. Consequently, a loading protocol that initially fails to activate motif R_2 may eventually trigger it after sufficient structural evolution, i.e., should $m_{12}^{(N)} > \frac{\theta_2^0 - \Gamma}{\xi}$

While the structural evolution of m_{12} may be gradual, the transition law can evolve discontinuously given particle packing rearrangements. Before the rearrangement, coarse-grained coordinate q describing progress along the collective motion associated with a localized rearrangement lies near one local minimum of the energy landscape. As loading increases, the minimum eventually loses stability, and the system rapidly evolves to another nearby minimum corresponding to a new particle configuration. Since rearrangements occur when a metastable state loses stability, we approximate the energy near such an instability by the simplest polynomial that can describe the disappearance of a local minimum, or

$$E(q) \simeq \frac{q^4}{4} + \frac{a}{2}q^2 + cq. \tag{58}$$

The disappearance of the local minimum is governed by

$$\frac{\partial E}{\partial q} = 0, \qquad \frac{\partial^2 E}{\partial q^2} = 0, \tag{59}$$

which determines the critical loading amplitude $\Gamma_c(m_{12})$ (assuming that c depends on m_{12}). You are perhaps familiar with such analysis with the van der Waals gas in terms of identifying the critical point in the water phase diagram. In catastrophe theory, this local behavior is described by the fold catastrophe (given the two parameters a and c). The activation threshold therefore emerges from the geometry of the evolving energy landscape rather than being imposed phenomenologically. The learning mechanism can be summarized as rearrangements $\rightarrow$ structural memory $\rightarrow$ evolving energy landscape $\rightarrow$ new metastable state $\rightarrow$ modified FSM.

A bound on learning. Since $m_{12}^{(\infty)} = \frac{\gamma}{\beta}$, the structural memory cannot grow indefinitely. Consequently, $\Gamma_c(m_{12})$ can only move a finite amount. The model therefore predicts a finite upper bound on learning performance arising from finite landscape plasticity rather than imperfect optimization. In particular, the largest threshold shift is $\xi\frac{\gamma}{\beta}$ and so motif R_2 being learned is possible if $\theta_2^0 - \Gamma < \xi\frac{\gamma}{\beta}$. Should this condition not be satisfied, then no amount of repeated training can trigger R_2 given R_1. Finding such bounds will be important in terms of bounds on learning. Perhaps this toy model gets at the heart of why our best student was a B- student.

Before leaving this section behind, this toy model raises numerous open questions: Can history-dependent transition graphs be learned systematically? Can one classify different physical learning universality classes? Can statistical mechanics provide a theory of learning in evolving energy landscapes?

5 Concluding Remarks

We have seen how physical systems ranging from fixed architecture resistor networks (and trainable parameters) to particle packings can learn different tasks. The emerging field of physical learning is indeed taking shape with new algorithms being uncovered to be applied to nonliving systems to give rise to new types of learning platforms that will ultimately hopefully complement conventional neural network platforms. I envision that perhaps depending on the classes of problems one would like to solve, one platform may be more handy than another bearing in mind the different trade-offs, including power consumption.

Viewing biological systems through this learning lens will ultimately be paramount to understanding how biology actually works. When studying a biological system beyond its immediate response to some perturbation, we should not be pre-occupied with developing biophysical models with fixed parameters. Rather, we should consider the system to be a learning system and develop teaching protocols to figure out how these systems learn. If you are curious about applying contrastive learning techniques to a cell packing in which cell rearrangements and weight updates are involved please check out a pre-print by Shabeeb Ameen, Tao Zhang, and myself [23].

Since the field of physical learning is relatively young, so to speak, there is much to do and even simple results can make an impact. Here are two review articles on the subject for your perusal [24, 25]. Moreover, the history dependence discussed explicitly in the previous section cries out for a statistical mechanics of learning in which history is part and parcel to the framework. Please stay tuned!

References

[1] John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities." In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[2] Aurèle Boussard et al. "Adaptive behaviour and learning in slime moulds: the role of oscillations". In: *Philosophical Transactions of the Royal Society B* 376.1820 (2021), p. 20190757.

[3] Samuel J Gershman et al. "Reconsidering the evidence for learning in single cells". In: *Elife* 10 (2021), e61907.

[4] Deepa Rajan et al. "Single-cell analysis of habituation in Stentor coeruleus". In: *Current Biology* 33.2 (2023), pp. 241–251.

[5] Lisa Schick et al. "Decision-making in light-trapped slime molds involves active mechanical processes". In: *PRX Life* (Apr. 2026), pp. –. DOI: 10.1103/rv7g-d9kx. URL: https://link.aps.org/doi/10.1103/rv7g-d9kx.

[6] R. A. Fisher. "The use of multiple measurements in taxonomic problems". In: *Annals of Eugenics* 7.2 (1936), pp. 179–188. DOI: 10.1111/j.1469-1809.1936.tb02137.x.

[7] Vidyesh Rao Anisetti, Benjamin Scellier, and J. M. Schwarz. "Learning by non-interfering feedback chemical signaling in physical networks". In: *Physical Review Research* 5.2 (2023), p. 023024.

[8] Benjamin Scellier and Yoshua Bengio. "Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation". In: *Frontiers in Computational Neuroscience* Volume 11 - 2017 (2017). ISSN: 1662-5188. DOI: 10.3389/fncom.2017.00024.

[9] Benjamin Scellier. "A deep learning theory for neural networks grounded in physics". In: *arXiv preprint arXiv:2103.09985* (2021).

[10] Menachem Stern et al. "Supervised learning in physical networks: From machine learning to learning machines". In: *Physical Review X* 11.2 (2021), p. 021045.

[11] Shuaifeng Li and Xiaoming Mao. "Training all-mechanical neural networks for task learning through in situ backpropagation". In: *Nature Communications* 15.1 (2024), p. 10528. DOI: 10.1038/s41467-024-54849-z.

[12] Joshua A McGinnis, Adam G Kline, and Yoichiro Mori. "A Conservation Law for Equilibrium Propagation and Coupled Learning". In: *Available at SSRN 7029694* (2026).

[13] Kyungeun Kim et al. "Perturbative Contrastive Physical Learning". In: *arXiv preprint arXiv:2606.09756* (2026).

[14] Andrea J Liu and Sidney R Nagel. "Jamming is not just cool any more". In: *Nature* 396.6706 (1998), pp. 21–22.

[15] Nathan C Keim et al. "Memory formation in matter". In: *Reviews of Modern Physics* 91.3 (2019), p. 035002.

[16] Nathan C Keim and Joseph D Paulsen. "Multiperiodic orbits from interacting soft spots in cyclically sheared amorphous solids". In: *Science Advances* 7.33 (2021), eabg7685.

[17] Martin van Hecke. "Profusion of transition pathways for interacting hysterons". In: *Physical Review E* 104.5 (2021), p. 054608.

[18] Wenjing Guo et al. "Learning Associations in Reconfigurable Particle Packings via Local Cyclic Driving". In: *arXiv preprint arXiv:2603.14534* (2026).

[19] Ivan Pavlov. "The work of the digestive glands". In: *Scientific and Medical Knowledge Production, 1796-1918*. Routledge, 2023, pp. 157–173.

[20] Le Yan et al. "Architecture and coevolution of allosteric materials". In: *Proceedings of the National Academy of Sciences* 114.10 (2017), pp. 2526–2531.

[21] Filip Jagodzinski, Jeanne Hardy, and Ileana Streinu. "Using rigidity analysis to probe mutation-induced structural changes in proteins". In: *Journal of bioinformatics and computational biology* 10.03 (2012), p. 1242010.

[22] Jason W Rocks et al. "Designing allostery-inspired response in mechanical networks". In: *Proceedings of the National Academy of Sciences* 114.10 (2017), pp. 2520–2525.

[23] Shabeeb Ameen, Tao Zhang, and JM Schwarz. "Training cell stress patterns in 3D cellular packings". In: *arXiv preprint arXiv:2604.25439* (2026).

[24] Menachem Stern and Arvind Murugan. "Learning without neurons in physical systems". In: *Annual Review of Condensed Matter Physics* 14.1 (2023), pp. 417–441.

[25] Ali Momeni et al. “Training of physical neural networks”. In: *Nature* 645.8079 (2025), pp. 53–61.