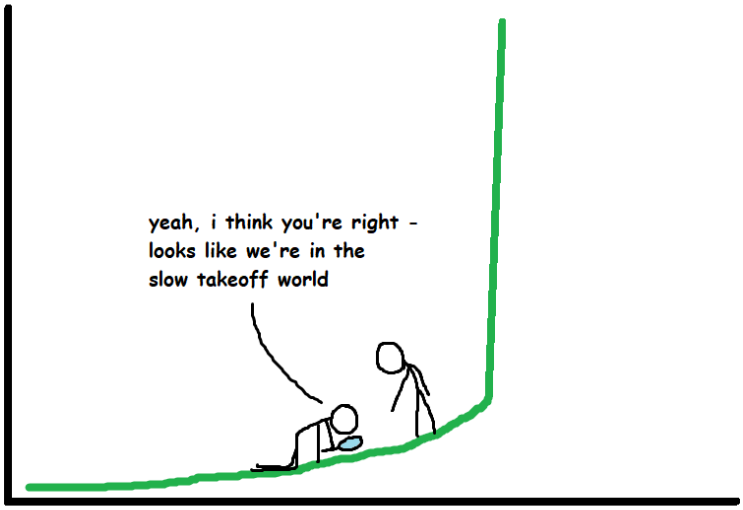
Detecting Phase Transitions

by **Jesse Hoogland**

Summary

- **Our aim was to develop tools that could detect phase transitions** (parts of training in which the model quickly learns a particular subtask) purely from weights.
- **We ended up blocked by finding suitable datasets** in which to study phase transitions.
    - We attempted several techniques to control and induce transitions: "graduating the data" and studying bounded polynomials of varying difficulty, but these all ran into problems.
    - We also looked at well-known tasks with transitions (grokking) and learning without transitions (MNIST & CIFAR-10).
- **We hereby lay the seeds for (future) phase detectors**.
    - You can find a GitHub repo with the (ongoing) work here.
    - Notebooks: (graduated) MNIST, bounded polynomials, CIFAR-10.

Motivation



Sudden changes are dangerous.

Phase transitions are dangerous. Sudden, discontinuous changes in capabilities are a major concern in AI safety. Whether you call these "sharp left turns," "phase transitions," or "intelligence explosions," the danger is that capabilities could outpace alignment, thereby magnifying the negative consequences of initially small differences in values.

Detecting phase transitions. This makes detecting, interpreting, and anticipating phase transitions a potentially high-impact intervention. In particular, we envision the development of "phase probes" that indicate the presence of a phase transition. These would ideally be low-cost and easy-to-integrate into the training pipeline.

Phase transitions leave structural traces. In this project, we studied whether we could detect phase transitions based on purely structural (=weight-based) probes. This was inspired by strong evidence that suggests that "cognitive phase transitions" leave traces in structure.

1. **Mechanistic evidence**: Nanda et al. (2023) demonstrate continuous progress measures that underlie the apparently discontinuous qualitative changes of "grokking" with modular arithmetic tasks.
2. **Singular learning theory (SLT)**: SLT suggests that singularities in the loss landscape determine the overall phase. Coupled with the fact that the singularities contain information about computational structure (see, e.g., Waring 2021), discontinuous changes in computation would follow from discontinuous changes in singularities.

3. **Divergences are detectable**: In SLT, phase transitions in learning are the same kinds of phase transitions as in physics: they involve divergences in the derivatives of the free energy. The presence of a divergence suggests the existence of macroscopically measurable signals.

Spectroscopy of singularities. Taking inspiration from the role of scanning tunnelling microscopes in solid state physics, the hope for SLT is to build devices that probe the divergences behind these phase transitions by investigating suitable observables over the loss landscape.

Aims for this hackathon. During the hackathon, we set up and studied several toy datasets to investigate phase transitions and lay the foundations for developing further phase probes.

Controlled Transitions

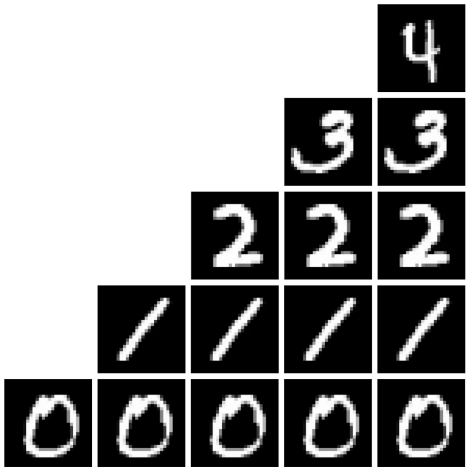
Controlled transitions. To study the structural traces of phase transitions, we needed a controlled environment in which we knew ahead of time which transitions were present and when they would occur. Then, we could test our structural probes and verify that they matched the known transitions on specific subtasks.

This turned out to be more difficult than we anticipated.

It's hard to control phase transitions. Most of our effort ended up going towards devising a dataset with controllable phase transitions. This ended up blocking much of the interpretability work we wanted to do. *Such is the way of hackathons.*

Graduated Training

Our first idea was "*graduated training.*"

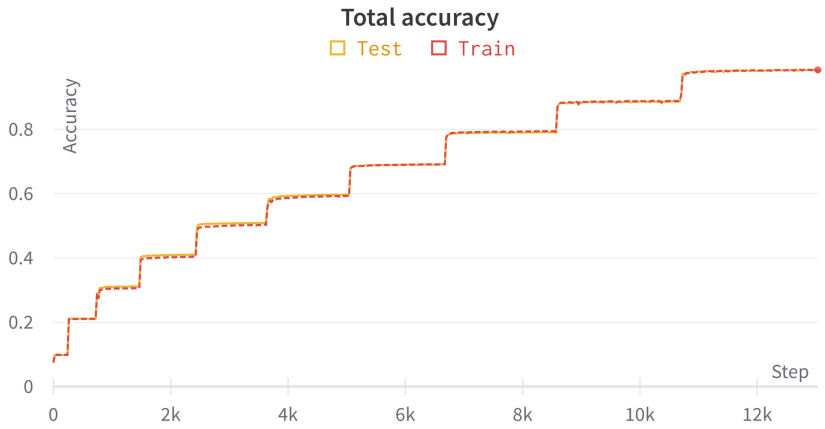


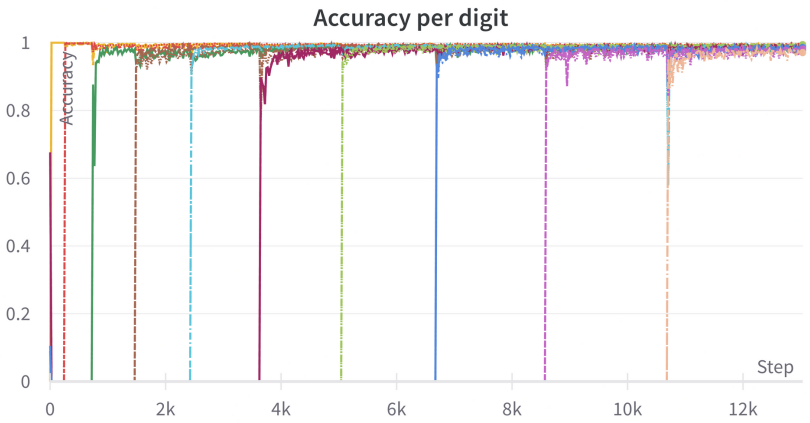
In graduated training, we introduce one task/class at a time.

Graduated training. In graduated training, one introduces one subtask at a time to the model. For example, in classification, the model is trained on a subset of the data belonging to just one label during the first X training steps. During the next X training steps, we expand the data to include another class label, and so on. After KX steps (where K is the number of labels), the model has seen all of the data.

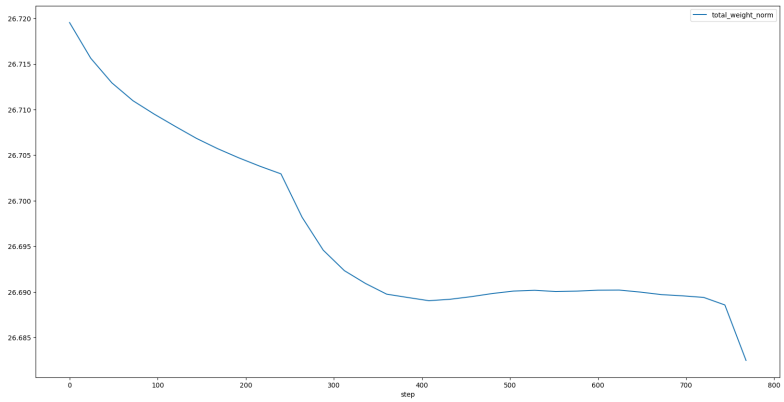
We applied this procedure to two datasets: MNIST and CIFAR.

MNIST



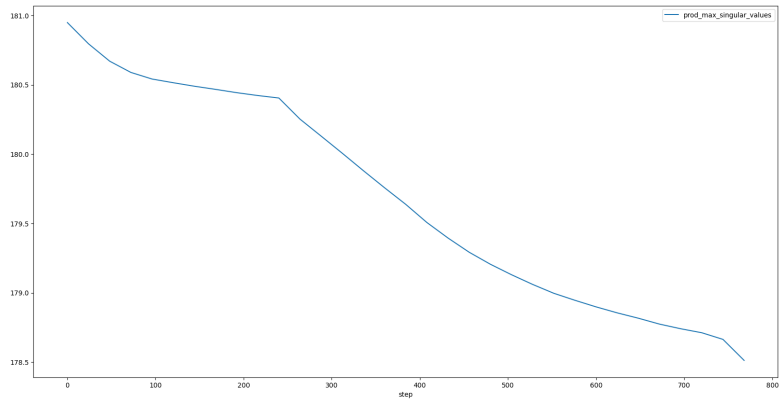


Weight norm. Plotting the weight norm shows curves like the following.



Weight norm over the first two transitions. The inflection points occur right at the moments that we introduce new data.

Maximal singular values. We calculated the maximum singular value for each layer. We then calculated the product of the singular values across all layers and plotted them as follows:



Same data, but now plotting the **product of the maximum singular values over each layer**.

Limitations of graduated training. The main problem with this approach is that introducing a new class is a confounder for learning a new task. *It's not obvious whether the structural changes are due to the former or the latter.*

What we're after, then, is proof of concept. The most successful observables we find here are candidates to investigate further in follow-up research on phase transitions in more realistic settings.

In addition, this procedure may change what the function learns. At the start, it's simpler to learn the function "map everything to class k" then it is to learn what class k "actually" is. It's not until you introduce the second class that the function actually has to learn anything meaningful. This makes it misleading to talk about the first X steps as consisting of a phase transition in which the model "learns" class k.

So these techniques are currently unable to tell us whether the transition involves learning the "correct" behavior — only that something has changed.

Bounded (Legendre) polynomials

Bounded Polynomials. Our second thought for a dataset with controllable phase transitions was to fit polynomials of various degrees. The idea was that higher degrees would be more difficult to learn and take more time.

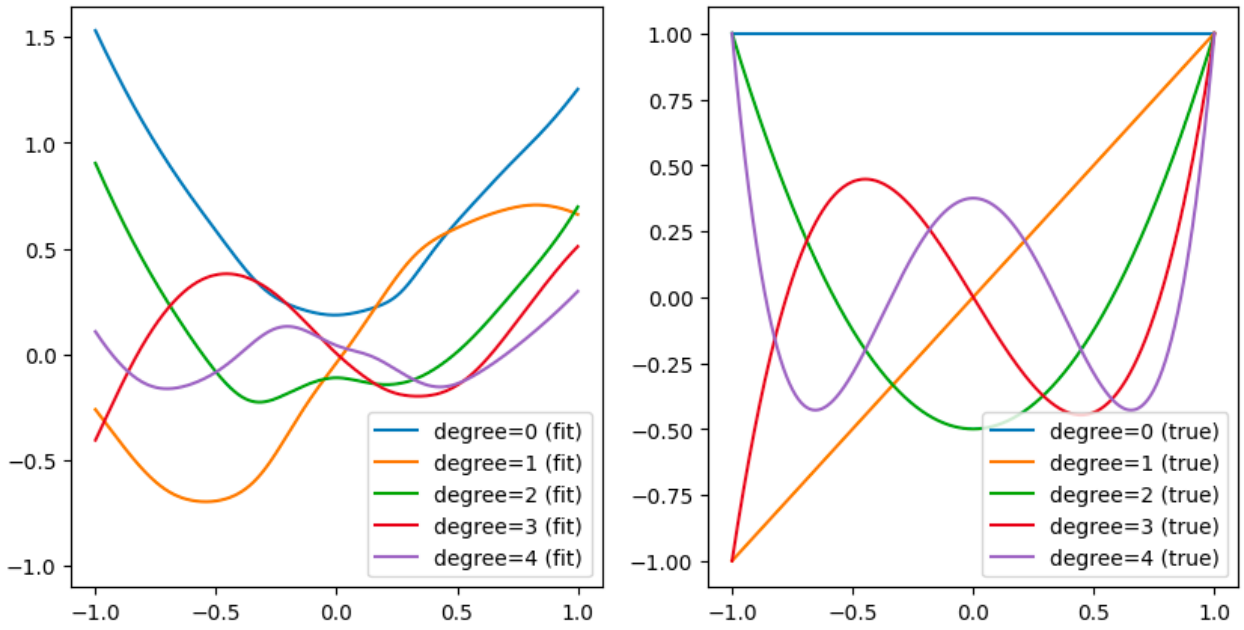
In particular, for a given maximum degree, d, we generated d-dimensional input vectors. For each degree $0 \leq i \leq d$, the generated input vector has i-th component within the range $-1 \leq x \leq 1$ and all other components set to zero. We then

applied a positional encoding to allow the model to determine the appropriate polynomial to compute when $x = 0$.

(We tried simpler input-outputs, $[i, x] \to y$, but the model has a hard time learning this.)

The corresponding output was a vector of the same shape with the corresponding output value for the Legendre polynomial of degree i in the i-th component.

Limitations of bounded polynomials. This intuition seems to have been partially confirmed, but we faced barriers in actually getting the network to learn the behavior. By the time we got it working, the hackathon was nearly over. *Eh.*



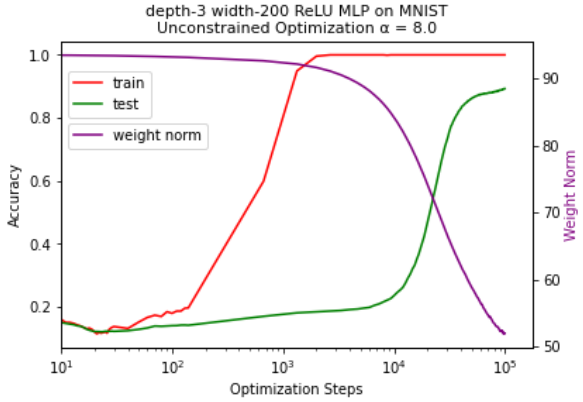
We haven't quite gotten the models to learn this task well enough yet to put it to good use.

Grokking

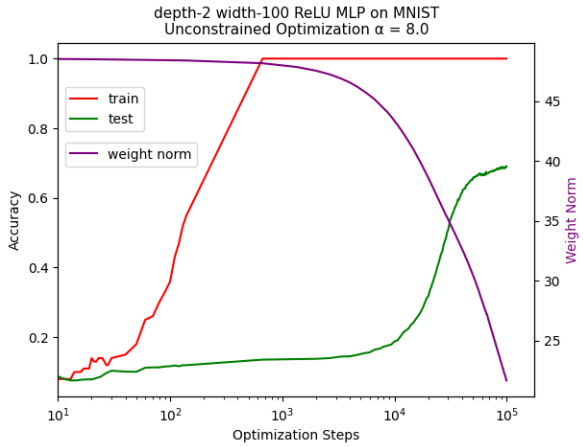
We also looked at an existing dataset in which the phase transition was already known, the modular arithmetic dataset behind grokking.

(Relative) Weight norm. Based on the Omnigrok explanation of grokking given by Liu et al. 2022, we looked at the role of weight norm in grokking. it appears that we can see grokking appear in a model that is learning the MNIST dataset. It turns out that an observable that appears to give a phase transition is the weight norm $||w||_2$.

In the paper, the authors scale the initial weights by a factor $\alpha \in [0.1, 10]$. For example, in the authors' notebook, when $\alpha = 8$ we get the following graph:

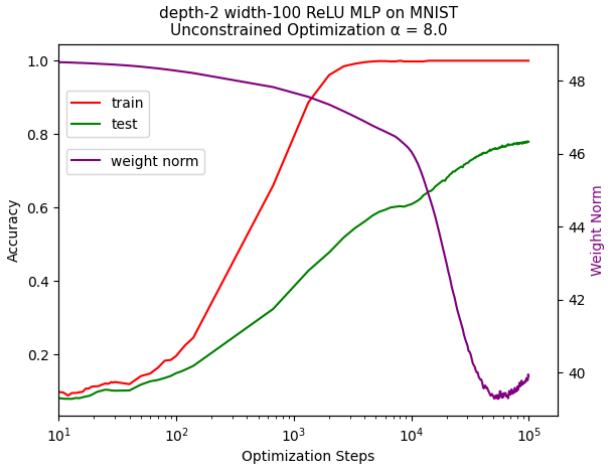


Lowering the depth and width of the model (with the goal of getting a shorter run time), we still observed an acute transition in the accuracy and the weight norm:



With depth 2 layers and 100 width and training points reducing from 1000 to 100, this took 15 minutes to train on a free colab GPU.

While it didn't get very high test accuracy, we can still see an increase in the performance of the model associated with a drastic change in the weight norm. Here's one more:

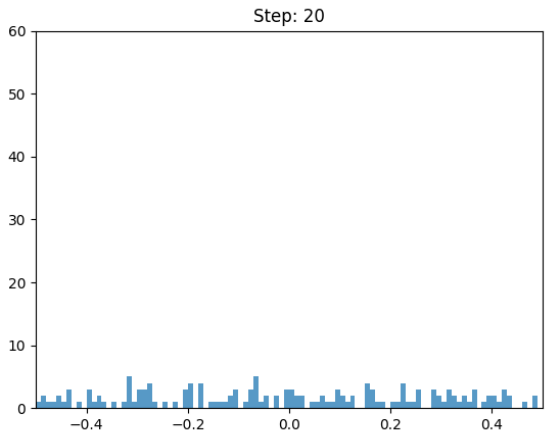


With 1,000 training points, this took 1 hour to run.

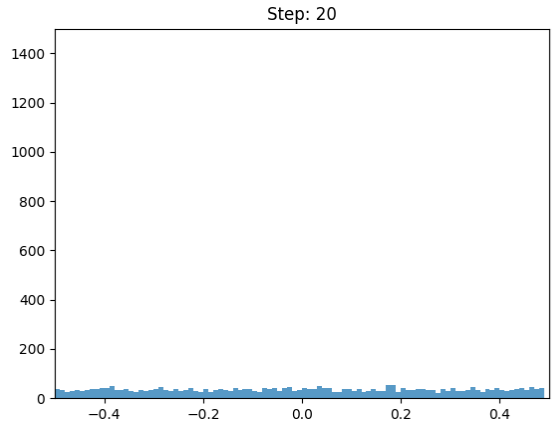
So not really worth the extra run time. Although perhaps the key takeaway from this paper and the above might be that the weight norm is a key observable that appears to encode some kind of phase transition.

CIFAR-10

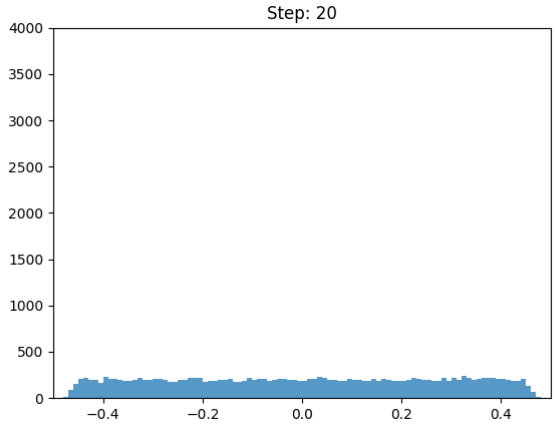
Weights. Below you'll find animations of the distribution of weight values over the course of training (for a simple 4-layer model).



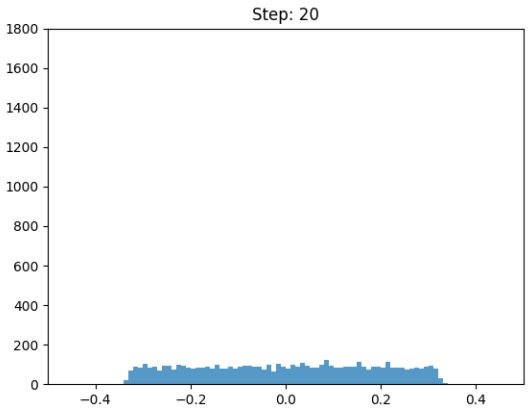
First convolutional layer.



Second convolutional layer.



Third convolutional layer.

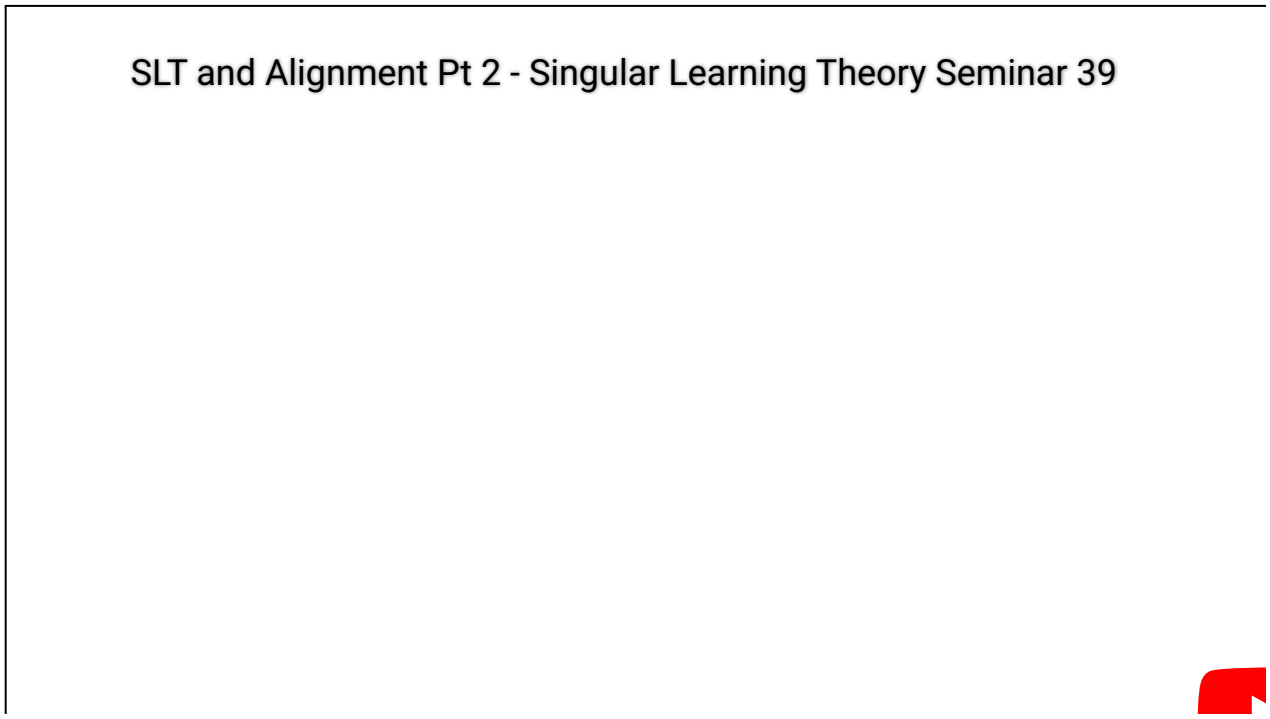


Fully connected layer.

Appendix

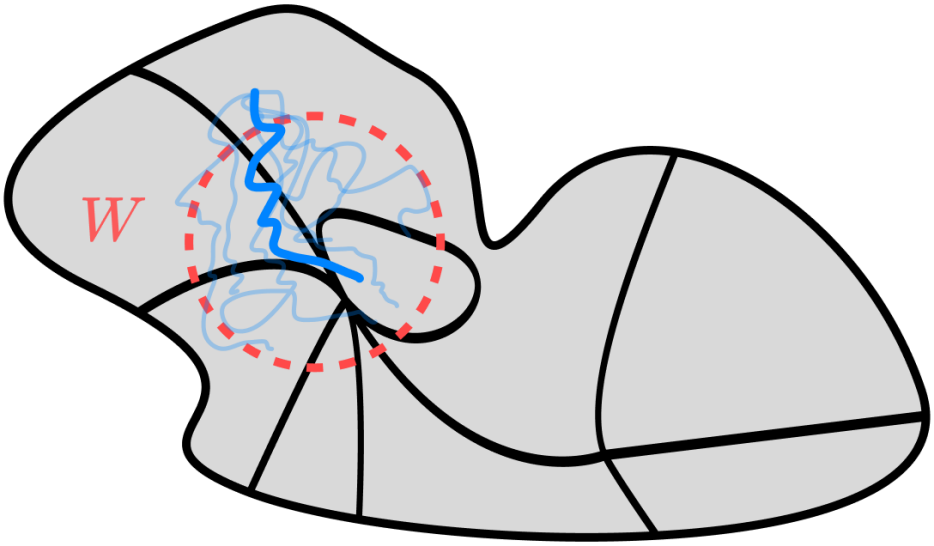
Integration phase detectors in training

What might phase detectors actually look like when applied to the training process?



Sampling the free energy. The way we might estimate the free energy (and any other observables) could look something like the following:

1. Take a checkpoint.
2. Generate several perturbations (e.g., apply Gaussian noise to the weights).
    a. With many dimensions, there might be more efficient ways to generate these perturbations, such as applying noise only in the k-most important dimensions (as decided by, e.g., a PCA over recent gradients).
3. Evolve the perturbations for several timesteps on the same batches.
4. Estimate your observable over the model and its perturbations.
5. Look for discrete changes in the observable between different checkpoints.



We can sample the local geometry by evolving a few models from similar starting configurations over several timesteps.

Review of phase transitions

But what kinds of "phase transitions." There are many competing notions of phase transitions that show up in the literature. Below we've reviewed several of the more common and salient examples. As in physics, t the term "phase transition" can refer to discontinuous changes that result from continuous changes in *any* controllable dial (usually either training steps or parameter count).

For the purpose of this project, we've restricted our experimental attention to *transitions over training time*.

Transitions in subtasks

LLM "emergence." The kind of phase transition we are most familiar with (and interested in) is the kind associated with LLMs as one increases scale. As LLMs increase in size, their overall performance improves steadily, but their performance on specific subtasks (e.g., coding, poetry, theory of mind, physics-problem-solving) can jump suddenly.

Quantization Hypothesis. Michaud et al. (2023) put forth an explanation for this phenomenon in terms of what they call the "Quantization Hypothesis." According to this hypothesis, the dataset would consist of a finite number of discrete subtasks ("quanta"), whose frequencies (=how often they show up in any given sample) follow a power-law distribution. This predicts both (1) the observed power-law relations between dataset size, parameter count, training time, and loss, *and* (2) sudden discrete changes on particular subtasks.

Induction heads. One of the better-understood subtask transitions is the formation of induction heads (Olsson 2022). In their words, "[t]ransformer language models undergo a “phase change” early in training, during which induction heads form and simultaneously in-context learning improves dramatically."

Toy Models of Superposition. Phase transitions are not necessarily transitions in terms of training time or parameter count. Elhage et al. (2022) demonstrate a transition between different regimes of superposition that occurs as one changes the relative importance and sparsity of different features. In follow-up work, Henighan et al. (2023) demonstrate a similar transition between a memorizing (data-points-in-superposition) and generalizing (features-in-superposition) regime.

Grokking. Power et al. (2020) demonstrated that transformers trained on small modular arithmetic tasks could display "grokking" in which the model learns to generalize long after it learns to memorize the training data. Liu et al. (2022) put forth an explanation in terms of the initial weight norm being too large, which puts the model in a regime of weight space with many memorizing solutions nearby. A slow process of weight decay eventually pushes the model towards generalizing solutions. Nanda et al. (2023) demonstrate that structurally the model goes through three transitions: memorizing, circuit formation, and clean-up.

More general transitions

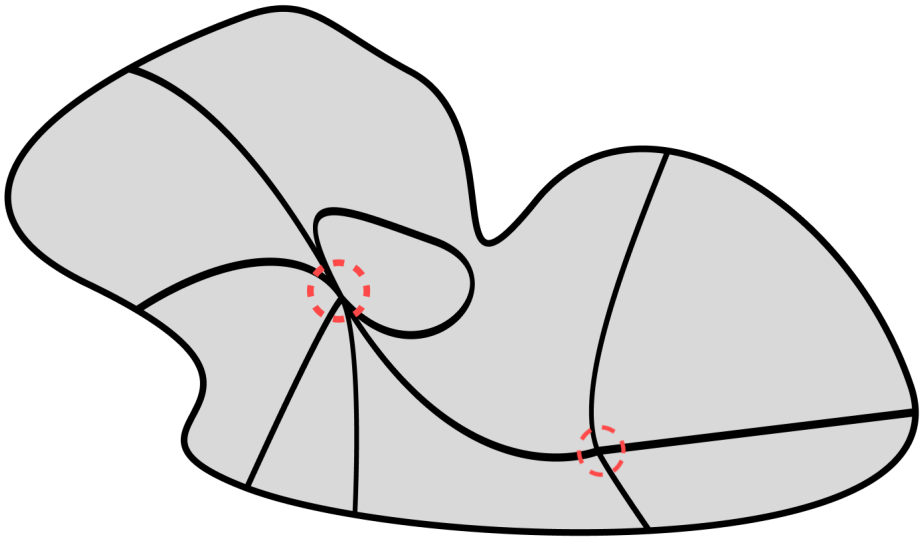
Singular learning theory (SLT). In physics, formal phase transitions involve discontinuities in the free energy or its derivatives. In learning theory, one thing we might study is discontinuities in the *restricted* free energy, which is the probability (in bits) associated with some subregion, W, of weight space,

$$F_t(W) = -\log \int_W p_t(w)\, \mathrm{d}w.$$

Here the probability $p_t(w)$ is the probability at timestep t of your optimizer ending up at w (under different choices of initialization/batch ordering).

What SLT tells us is that as the model gets closer to minima of its loss function (this is perhaps equivalent to the limit $t \to \infty$), these integrals become dominated by singularities in W. The same holds for the expected values of any observables, $\mathcal{O}$, based on this formula:

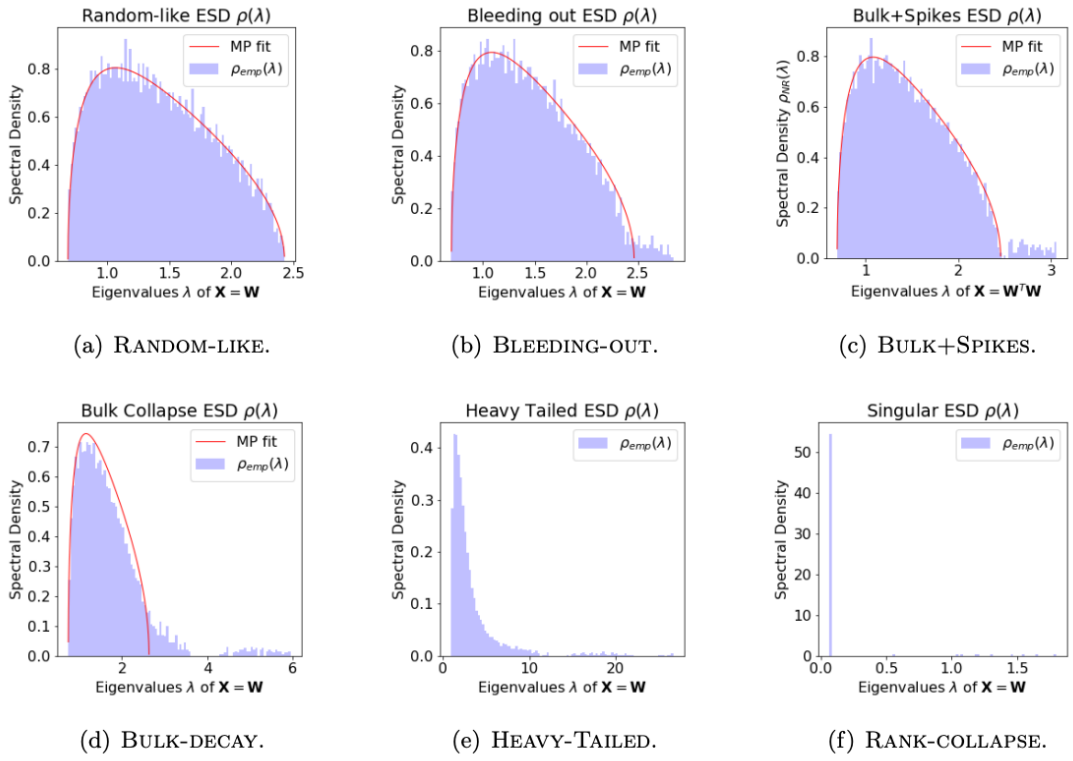
$$\mathbb{E}_t[\mathcal{O}|W] = \int_W \mathcal{O}(w)\, p_t(w)\, \mathrm{d}w$$



Weight-space. The lines running through this blob represent the set of minimum loss points. Phase transitions *might* correspond to changes in the singularities of the local minimum loss set (indicated by red circles).

In this picture, phase transitions during training time would correspond to discrete changes in the local singular geometry of the loss landscape — singularities that come into or out of view as you move the region W around weight space.

5+1 phases. Martin and Mahoney (2018) show that the eigenvalue spectra of hidden-layer weight matrices appear to go through five phases over the course of training, which correspond to the random matrix universality class best describing the observed spectrum. This inspired the WeightWatcher library.



(a) RANDOM-LIKE. (b) BLEEDING-OUT. (c) BULK+SPIKES.

(d) BULK-DECAY. (e) HEAVY-TAILED. (f) RANK-COLLAPSE.

Figure 14 from Martin and Mahoney (2018). Displays the distribution of singular values at different points during training.

Interpolation threshold in double descent. A well-studied transition is the interpolation threshold in double descent: the first point at which the model is able to achieve perfect training loss. Baldassi et al. (2022) demonstrate a second phase transition that follows the interpolation point, characterized by "the discontinuous appearance of a different kind of “atypical” structures [with] wide regions of the weight space that are particularly solution-dense and have good generalization properties."

Jamming transition. The interpolation transition has also been linked to the "jamming transition" of spin-glass physics, as explored by, for example, Geiger et al. (2020).

VERSION HISTORY