Feature-aware pruning in MLPs

Tanishq Kumar
Harvard College
tkumar@college.harvard.edu

Abstract

Recent work in feature learning theory theory of neural networks [Jacot et al., 2018, Chizat et al., 2019] has identified regimes in which neural networks behave like kernel methods. Since kernel methods are generalized linear models, they are not capable of adapting their internal representations over the course of learning (what is called "feature learning.") In particular, form any network f_θ, we can define a kernel (called the NTK) K_θ, such that inference with that network is equivalent to doing kernel ridge regression (KRR) with K_θ. As we train the network to $f_{\theta'}$, the associated kernel $K_{\theta'}$ also evolves. Here, we study the lottery ticket phenomenon through this lens of kernel theory and feature learning. In particular, our main question is: do lottery tickets learn the same features over the course of sparsification as their parent networks? Does making lottery tickets behave more like linear models improve, or worsen, their performance at high sparsities, and why? In seeking a simple setting in which to replicate results and reason about network dynamics, we use a toy model of an MLP in a student-teacher task, then show that our insights apply out-of-the box to a more complicated task like MNIST classification.

1 Setup

Our project studies feature learning in lottery tickets using recently developed tools from feature learning theory. In particular, we are interested in how the representations change over the course of iterative magnitude pruning to discover lottery tickets over the course of training.

1.1 Introduction: The Lottery Ticket Hypothesis

There are three levels at which one can prune a neural network.

1. At initialization [Lee et al., 2018, Wang et al., 2020, Tanaka et al., 2020]
2. During training [Frankle and Carbin, 2018]
3. After training, before inference [Han et al., 2015]

In this paper, we're focused on one particular method for pruning at level (2) of abstraction: finding lottery tickets. In [Frankle and Carbin, 2018], an algorithm for pruning called "Iterative Magnitude Pruning with Rewind." In it, we repeat the following: initialize a neural network and store its initial random state, train it on some data, prune the lowest magnitude weights, and, crucially, rewind the nonpruned weights to their initial values. Alternatives to this algorithm, which we'll refer to as LTH as now on, are rewinding to random weights, which we'll refer to as "Random rewind" and doing IMP without rewinding at all, which we'll refer to as "No rewind." Below, we replicate the LTH paper and compare the three methods. This is an MLP trained on MNIST and pruned to a high sparsity level. We see LTH is the only network whose performance does not degrade rapidly at very high sparsity levels.

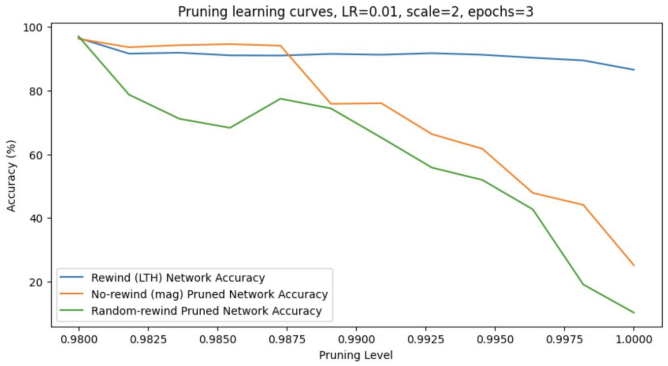

Figure 1: Lottery tickets (blue) are trainable to high sparsity. Rewind is necessary in IMP.

1.2 Related work: Linear vs Nonlinear Training Regimes

In the lazy training regime, a network $f(\boldsymbol{w}, \boldsymbol{x})$ of inputs $\boldsymbol{x}$ and parameters $\boldsymbol{w}$ obey the approximation $\mathrm{f}(\boldsymbol{w}, \boldsymbol{x}) \approx f(\boldsymbol{w}_0, \boldsymbol{x}) + \nabla_{\boldsymbol{w}} f(\boldsymbol{w}, \boldsymbol{x})|_{\boldsymbol{w}_0} \cdot (\boldsymbol{w} - \boldsymbol{w}_0)$. While this approximation is still capable of learning nonlinear functions of the input $\boldsymbol{x}$, it is a *linear model* in the trainable parameters $\boldsymbol{w}$. It can thus be recast as a kernel method with the neural tangent kernel K

$$K(\boldsymbol{x}, \boldsymbol{x}') = \nabla_{\boldsymbol{w}} f(\boldsymbol{w}, \boldsymbol{x})|_{\boldsymbol{w}_0} \cdot \nabla_{\boldsymbol{w}} f(\boldsymbol{w}, \boldsymbol{x}')|_{\boldsymbol{w}_0} \tag{1}$$

evaluated at initialization. This holds for any linearized model trained with GD on any loss, though some loss functions (e.g. cross-entropy) cause non-linear models to eventually deviate from their linearization [Lee et al., 2019]. Deep neural networks can approach this linearization in many ways:

- Large width: in commonly used parameterization and initialization schemes of neural networks, increasing the network width improves the quality of this approximation Jacot et al. [2018], Lee et al. [2019].

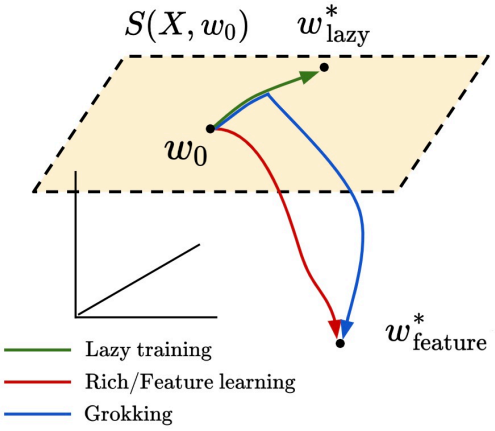

Figure 2: Mechanics of lazy learning, taken from Kumar et al. [2023]. This depicts movement of weights during training from the initialized weights w_0 in weight space, depicted as $\mathbb{R}^3$ in this image. Lazy training means linearizing the model movement through parameter space so it is restricted to a subsapce.

- Large initial weights: increasing the size of the initial weights also causes the network to behave closer to linearized dynamics [Chizat et al., 2019]. *We argue that this explains why Liu et al. [2022] and Varma et al. [2023] observed an association between grokking and large initial weight norm.*
- Label rescaling: scaling y by factor α^{-1} can induce lazy training [Geiger et al., 2020].
- Output rescaling: multiplying the network output logits by a large scale factor α can also induce lazy training [Chizat et al., 2019].

Here, we primarily use the last option, where the $\alpha \to \infty$ limit is one where equation 1.2 becomes exact and the network behaves at inference-time like a linear model (in the case of MSE, this linear model is kernel regression with the initial NTK).

2 Experiments and Results

2.1 Toy Model: Student-Teacher Task

Setup

Our toy model is a student-teacher setting with both networks using a one-hidden layer MLP architecture. This is a common setting on ML theory, and its behavior is well understood, eg in Mei et al. [2018], Gerbelot et al. [2022], Bordelon et al. [2020]. Both networks take the form

$$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{\alpha}{N} \sum_{i=1}^{N} \phi\left(\boldsymbol{w}_i \cdot \boldsymbol{x}\right), \quad \phi(h) = h + \frac{\epsilon}{2} h^2$$

where we note that this architecture (with the second layer, readout fixed) is called a "committee machine," and is a special case of a two layer MLP. It is studied in Saad and Solla [1995]. The key parameters of interest here are α, ϵ.

The value of α controls the scale of the output, and consequently the speed of feature learning. The value of ϵ alters the task structure and network structure: $\epsilon \to \infty$ corresponds to learning a pure quadratic function and $\epsilon \to 0$ corresponds to learning a pure linear function. We consider training on a fixed dataset $\{(\boldsymbol{x}_\mu, y_\mu)\}_{\mu=1}^{P}$ of P samples. The inputs $\boldsymbol{x}$ are drawn from an isotropic Gaussian distribution $\boldsymbol{x} \sim \mathcal{N}(0, \frac{1}{D}\boldsymbol{I})$. It will be convenient to introduce the following two summary statistics $\bar{\boldsymbol{w}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{w}_i \in \mathbb{R}^D$, $\quad \boldsymbol{M} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{w}_i \boldsymbol{w}_i^\top \in \mathbb{R}^{D\times D}$

Using these two moments of the weights, the neural network function can be written as $f(\boldsymbol{x}) = \alpha \bar{\boldsymbol{w}} \boldsymbol{x} + \frac{\alpha\epsilon}{2} \boldsymbol{x}^\top \boldsymbol{M} \boldsymbol{x}$. Further, the NTK can also be expressed as $K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}\cdot\boldsymbol{x}' + \epsilon(\boldsymbol{x}\cdot\boldsymbol{x}')\,\bar{\boldsymbol{w}}\cdot(\boldsymbol{x}+\boldsymbol{x}') + \epsilon^2(\boldsymbol{x}\cdot\boldsymbol{x}')\,\boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x}'$. At initialization in a wide network, $\bar{\boldsymbol{w}} = 0$ and $\boldsymbol{M} = \boldsymbol{I}$. Diagonalizing this initial NTK with respect to the Gaussian data distribution reveals that linear functions all have eigenvalue $\lambda_{lin} = D^{-1}$ while quadratic functions have eigenvalue $\lambda_{quadr} = 2\epsilon D^{-2}$. We see that ϵ controls the power the kernel places in quadratic functions (including the target $y(x)$). Thus we can see that the sample complexity of the task varies with ϵ and this is what we are changing when we change ϵ in the sweeps below. This is because it is known [Ghorbani et al., 2019, Damian et al., 2023] that the number of samples for such an MLP to learn quadratic vs linear functions are different. The main contribution of kernel theory in this paper is telling us to study pruning in this student-teacher setup and see the effects of varying α, ϵ in the first place. Now we see what these effects are on lottery ticket trainability and performance.

Varying laziness by tuning α

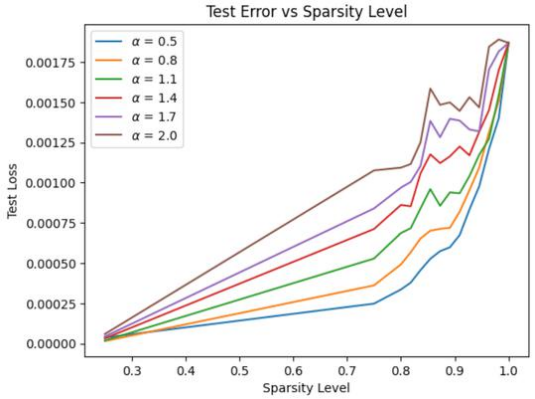

Figure 3: Sweep over laziness parameter α for our toy student-teacher model.

We can see above that changing the rate of feature learning impacts lottery tickets a lot! The plot above shows the same lottery ticket trained at several different rates of feature learning α, and clearly feature learning lottery tickets (small $\alpha = 0.5$) keep their low loss the longest over the course of sparsification. The mechanism behind this is unclear, and what I'm currently investigating.

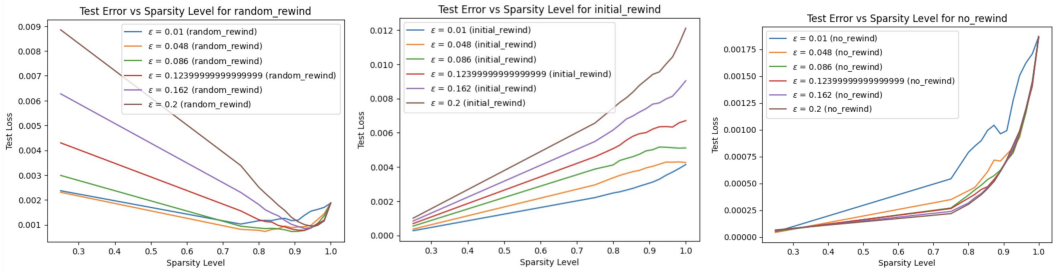

Figure 4: Sweep over CKA parameter ϵ that changes the task-model-alignment (changes the value of the CKA).

Varying task difficulty by tuning ϵ

We see that changing ϵ changes both the target function (teacher) and the learning function (student) since they are both using the same architecture, so the CKA, the quantity $\frac{yKy^T}{||K||_F||y||^2}$ changes because the labels y do (generated by the teacher) and the kernel K changes (because the student architecture does). While we only see the laziness α has a visible impact only on the lottery ticket rewind setting (and not on the random rewind and no rewind settings) here we see that changing the structure of the task has an impact in every setting. Curiously, the impact of ϵ is not uniform. Higher ϵ (learning higher degree polynomials) does better for random rewind (left) and no rewind (middle), but smaller (learning lower degree polynomial, almost a linear regression learning task) does better for the LTH. The mechanistic/theoretical reasons for this are unclear, and a topic for future work.

Lottery Ticket Dynamics

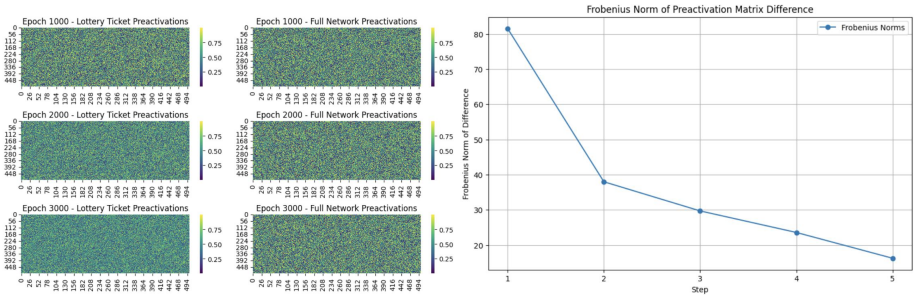

Figure 5: Tracking feature dynamics for lottery ticket vs full network over the course of training. (a) is preactivation (feature) matrices every 1000 epochs, and (b) is Frobenius norm of distance between the two matrices over the course of training. We see that the lottery ticket quickly learns the same features as the full networks as the two preactivation matrices get closer.

This plot tests the conjecture that lottery tickets don't just learn the same end-time solution as the full network, but that they learn the same features at *every level of sparsity* as they are discovered via iterative magnitude pruning. Above on the left is a heat map of the preactivation matrix hh^T of the lottery ticket, and student, respectively. They differ at initialization and converge over the course of training, meaning they learn the same features. That is, this is evidence that lottery tickets generalize as well as the original features because they have the same feature dynamics!

2.2 Results: Beating lottery tickets on MNIST

There's three statements one could make about lottery tickets, each stronger than the last:

1. Pruned networks reach same end-time test error as full network
2. (Stronger) Pruned networks reach same end-time features as full network
3. (Strongest) Pruned networks have same feature dynamics as full network

The lottery ticket paper shows (1) is true, but our toy model and in particular 5 the stronger statement (3) may be true. So we can test this in a non-artificial setting, one that in fact was included in the original lottery ticket paper.

Conjecture: LTH paper shows (1). Toy model suggests (3). This property does not hold for pruning with random rewinding or no rewinding!

Hypothesis: sweeping over rate of feature learning should change performance of lottery tickets uniformly, but performance of random/no rewind in a complicated, messy way (features are not necessarily the same over sweep). As we can see below, this is exactly what happens! Moreover, the orange curve on the left of 6, under the title "LTH Rewind" is the same curve in Figure 2 of [Michael Carbin, 2019]. Notice then that the curves above it on the same plot correspond to lazier version of the lottery ticket: networks whose features (as measured by the NTK of the network throughout training) change less. We see that the curves above the orange curve keep high accuracy for longer under intense sparsification! This means we have beat the original lottery ticket on MNIST by tuning the rate of feature learning using a parameter introduced in Chizat et al. [2019].

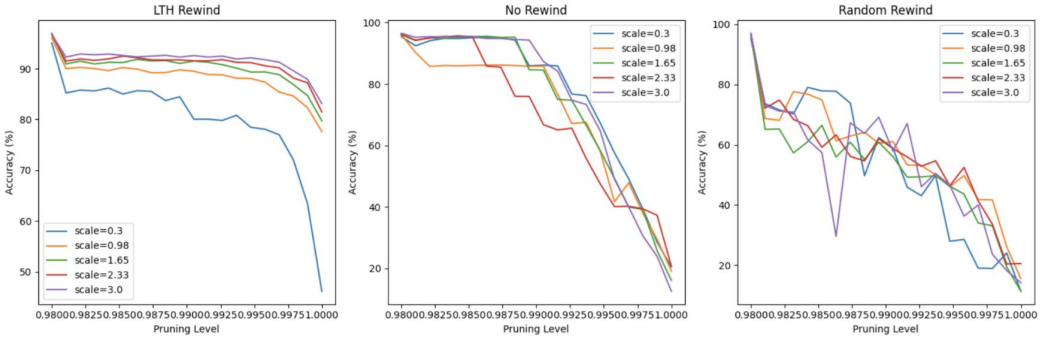

Figure 6: Sparse training dynamics for various pruning methods on MNIST, sweeps over laziness parameter α. This tests and confirms our prediction that changing the rate of feature learning will have a uniform effect in lottery tickets, but not in pruned subnetworks derived without weight rewinding (where the pruned network need not learn identical features to the subnetwork).

2.3 Conclusion and Future Work

In summary, we found that changing the rate of feature learning in networks does affect how you prune them, and evidence that the feature learning *dynamics* for training a sparse network are the same as for the full network. We also saw that varying task difficulty (in the sense of CKA, which we varied by tuning the parameters ϵ on our toy model) changes performance of sparse networks in interesting ways. Future work includes

- Understand why tuning ϵ has an effect on sparsification in the first place when the main way it affects learning in our setting is by changing sample complexity
- In particular, why does ϵ have different effects on sparsification in different rewind mechanisms? It seems that a higher degree learning task $\epsilon \gg 0$ allows us to sparsify more while maintaining accuracy in random/no rewind settings, but the opposite trend holds with lottery tickets.
- Why do feature learning networks $\alpha \to 0$ do better in our toy model, but lazy networks $\alpha \gg 0$ do better on MNIST? Either way, we've seen that laziness affects the performance of lottery tickets, but the mechanism behind this is not clear.

Overall, the experiments were implemented in Google Colab with the JAX framework, training was done on an NVIDIA A100 cluster on GCP, and the experiments were written in around 1000 lines of code.

References

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems*, 32, 2019.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020.

Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in neural information processing systems*, 33:6377–6389, 2020.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.

Ziming Liu, Eric J Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data. *arXiv preprint arXiv:2210.01117*, 2022.

Vikrant Varma, Rohin Shah, Zachary Kenton, János Kramár, and Ramana Kumar. Explaining grokking through circuit efficiency. *arXiv preprint arXiv:2309.02390*, 2023.

Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020 (11):113301, 2020.

Tanishq Kumar, Blake Bordelon, Samuel J Gershman, and Cengiz Pehlevan. Grokking as the transition from lazy to rich training dynamics. *arXiv preprint arXiv:2310.06110*, 2023.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

Cedric Gerbelot, Emanuele Troiani, Francesca Mignacco, Florent Krzakala, and Lenka Zdeborova. Rigorous dynamical mean field theory for stochastic gradient descent methods. *arXiv preprint arXiv:2210.06591*, 2022.

Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. In *International Conference on Machine Learning*, pages 1024–1034. PMLR, 2020.

David Saad and Sara A Solla. On-line learning in soft committee machines. *Physical Review E*, 52 (4):4225, 1995.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Limitations of lazy training of two-layers neural networks, 2019.

Alex Damian, Eshaan Nichani, Rong Ge, and Jason D. Lee. Smoothing the landscape boosts the signal for sgd: Optimal sample complexity for learning single index models, 2023.