
Investigating the Effect of Model Capacity Constraints on Belief State Representations¹

Ari Brill
NASA Goddard Space
Flight Center

Chu Chen
Independent

Organized by
Adam Shai, Paul Riechers, PIBBSS

Abstract

Computational mechanics provides a formal framework for understanding the concepts needed to perform optimal prediction. Abstraction and generalization seem core to the function of intelligent systems, but are not yet well understood. Computational mechanics may present a promising approach to studying these capabilities. As a preliminary exploration, we examine the effect of weight decay on the fractal structure of belief state representations in a transformer’s residual stream. We find that models trained with increasing weight decay coefficients learn increasingly coarse-grained belief state representations.

Keywords: Computational Mechanics, AI safety, abstraction, generalization, regularization

1. Introduction

Computational mechanics provides a quantitative description of the computations and structures needed to optimally predict sequential data [1, 2]. By studying the belief state representations learned by neural networks trained on data generated by simple processes, we may be able to develop better tools for understanding the world models developed by realistic, powerful AI systems. An improved theoretical understanding of learned world models can pay great dividends for AI safety by

¹ Research conducted at the Computational Mechanics For AI Safety Hackathon, 2024 (<https://www.apartresearch.com/event/compmech>)

allowing researchers to interpret, overwrite, and elicit latent knowledge from an AI system’s internal representations.

The ability to abstract and generalize concepts is core to human intelligence, and may be essential for generally intelligent AI systems as well. In this project, we take first steps toward exploring how computational mechanics can provide a framework to describe abstraction, generalization, and hierarchical approximation of concepts in AI systems.

First, abstraction is the ability of a cognitive system to distill a general high-level concept from specific examples and flexibly apply it in disparate situations. One toy model of how AI systems could apply abstract concepts is to train a neural network on sequences generated by two models yielding identical mixed-state presentations (MSPs) but different vocabularies. A network representing a single belief state geometry used for predicting both sets of sequences could be considered to be applying a single abstract concept.

Second, concepts that generalize to unseen situations are crucial to predictive capabilities in both human and machine learning systems. Generalization could be modeled with computational mechanics by training a neural network on noisy data, and investigating the factors that determine the extent to which it learns to represent the underlying MSP, rather than a more complex, non-generalizing one incorporating memorized noise that improves training performance only.

Third, many natural systems are well-modeled by a hierarchy of abstractions including both approximate high-level concepts and fine-grained lower-level ones. In the context of computational mechanics, approximation can be understood by considering the predictive power of finite approximations to the ideal MSP containing infinite features [3]. For a process giving an MSP with fractal structure, this manifests as learning a coarse-grained approximation to the optimal MSP.

These fundamental cognitive abilities arise because a bounded intelligent system can’t simply memorize everything. Adaptable, generalizable, and hierarchically organized concepts allow a limited agent to efficiently handle a limitless variety of situations. Studying these behaviors in neural networks requires developing methods to effectively and precisely control model capacity.

The model capacity of neural network architectures is poorly understood and is an active research area [e.g. 4, 5]. For this project, we explored the effect of constraining model capacity on learned belief state representations via regularization. Specifically, we constrained model capacity using weight decay [6], and hypothesized that as the weight decay coefficient increases, the model capacity decreases, and the network learns an increasingly coarse-grained approximation to the optimal MSP.

Weight decay shrinks all parameters by a fixed factor at every training step, decaying them exponentially. Mathematically, it is described by the relation

$$\Theta_{t+1} = (1 - \gamma\lambda)\Theta_t - \gamma\nabla\Theta_t,$$

where Θ represents the network parameters, γ is the learning rate, λ is the weight decay coefficient, and $\nabla\Theta$ is the gradient update term. For vanilla SGD, weight decay is equivalent to L_2 regularization.

2. Methods

To investigate the effect of weight decay, we trained transformers on data generated by the Mess3 Hidden Markov Model (HMM). We extracted the belief state geometry by computing the linear projection of the theoretical MSP onto the model’s final-layer residual stream activations.

Our methods closely follow those of [2], and are based on the provided example notebooks. Our code is available in Google Colab, at [weight_decay.ipynb](#). The final training runs were conducted on a machine with a GA104M GPU. For all experiments, we used the AdamW optimizer [7] with a learning rate of $3e-4$.

We chose Mess3 as the data-generating process in order to resolve fractal MSP geometry. It also has a one-to-one correspondence between belief states and next-token predictions, simplifying extraction of the belief state geometry from the residual stream. We trained our models on a stream of randomly generated data, and monitored the loss throughout training by evaluating them on a fixed validation set at regular intervals.

Essentially all models we studied in this project were able to achieve low training and validation loss, similar to the theoretical myopic entropy rate derived from the MSP. For this reason, our model evaluations primarily focused on the qualitative appearance of the simplex projections extracted from the residual stream.

After implementing a pipeline to train transformers on Mess3 data, we explored ways to decrease the model size while still obtaining clear belief state representations. We did so both to increase our experiments’ speed during the short duration of the hackathon and to make constraining the model capacity easier.

To reduce the number of model parameters, we first explored using attention-only transformers, with no MLP layers. We used 4-layer transformers with 4 attention heads, head dimension 16, and model dimension 64. For these hyperparameters, removing the MLP layers reduced the parameter count from 201,091 to 68,227, an almost 3x reduction. We found that attention-only transformers achieved adequate performance and used them for the remainder of our experiments.

Next, we tried reducing the number of layers from 4 to as few as 1. This experiment was inspired by Fig. 7E of [2], which shows that for the Mess3 process, the belief state geometry can be recovered with low mean squared error as early as after the first layer.

However, we observed that while models with 1 or 2 layers were able to attain good performance on some runs, this change appeared to increase instability in the quality of the learned belief state geometry from run to run. Rather than explore this further, we reverted to using 4-layer transformers for our final experiments.

Finally, we trained transformers on a range of weight decay coefficients, as well as with no weight decay, and recovered projections of the belief state geometry at several checkpoints during training.

3. Results

We trained a 4-layer attention-only transformer with weight decay coefficients of 0, 0.3, 0.5, 0.7, 0.85, and 1. All other hyperparameters were held constant. Figure 1 shows the linear projection of the transformer’s final layer activations onto the theoretical MSP for each model after training for 500 epochs.

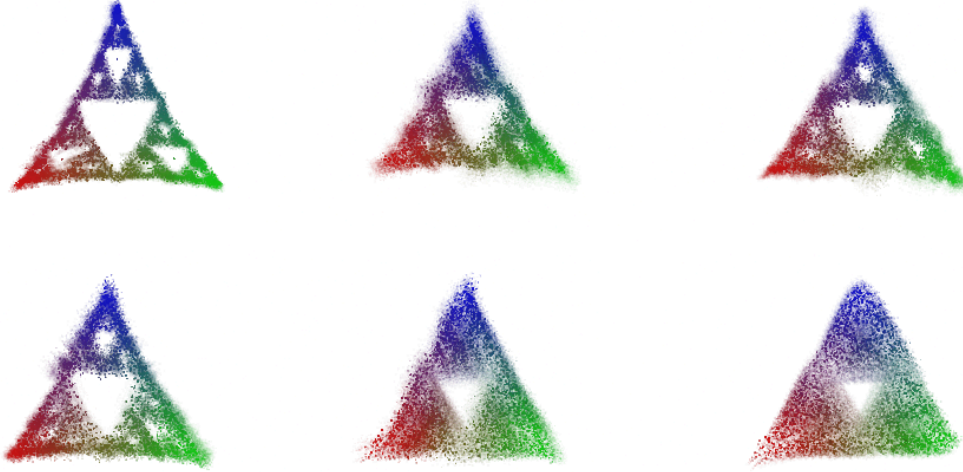


Figure 1 – Residual stream simplex plots after 500 epochs. Going left to right: (Top row) weight decay = 0, 0.3, 0.5. (Bottom row) weight decay = 0.7, 0.85, 1.

As can be seen in Figure 1, as the weight decay coefficient increases from 0, the learned belief state representations tend to become increasingly coarse-grained. In particular, while 27 distinct triangular structures can be visually resolved in the belief state simplex of the model trained without weight decay, only 3 large structures can be resolved clearly in the simplex of the model trained with a weight decay coefficient of 1.

We also plot the final layer projections over the course of training for each weight decay coefficient, shown in Figure 2. We observe that models trained with weight decay sometimes undergo an apparent phase transition in the belief state structure during training, in which a relatively crisp structure blurs before equilibrating to a stable geometry. For example, this is visible in Figure 2 for the model with a weight decay coefficient of 0.3 (second row).

We did not note any obvious loss increases associated with these transitions. We hypothesize that this could occur because weight decay requires significant training time before its impact on the model weights gradually becomes apparent, similar to

the phase transitions observed in grokking [8]. Investigating this phenomenon could be an interesting subject for future work.

-

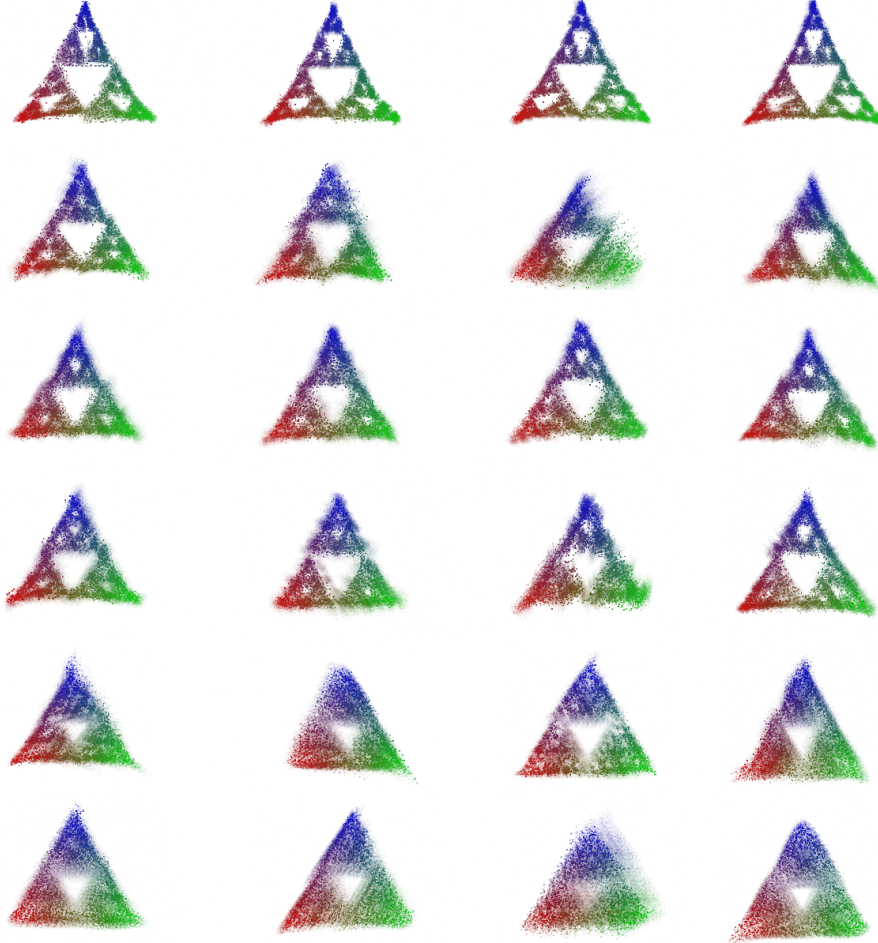


Figure 2 - Belief geometry over training. From right to left: 125, 250, 375, 500 epochs. From top to bottom: weight decay = 0, 0.3, 0.5, 0.7, 0.85, 1.

4. Discussion and Conclusion

Looking ahead, our qualitative findings in this project could be built on to quantitatively compare the recovered belief states with theoretical predictions for finite predictive models [3]. In addition, models trained for longer than possible during the course of the hackathon may converge to a crisper simplex, allowing investigation of a broader range of weight decay coefficients. Additional future work could investigate theoretically motivated approaches for normalizing weight decay to the total number of training steps [7]. Finally, we hope in future work to apply the methods for constraining model capacity developed in this project to study toy models of abstraction and generalization using computational mechanics.

5. References

- [1] Shalizi, C. R., & Crutchfield, J. P. (2001). Computational mechanics: Pattern and prediction, structure and simplicity. *Journal of statistical physics*, 104, 817-879.
- [2] Shai, A. S., Marzen, S. E., Teixeira, L., Oldenziel, A. G., & Riechers, P. M. (2024). Transformers represent belief state geometry in their residual stream. arXiv preprint arXiv:2405.15943.
- [3] Marzen, S. E., & Crutchfield, J. P. (2017). Nearly maximally predictive features and their dimensions. *Physical Review E*, 95(5), 051301.
- [4] Roberts, D. A., Yaida, S., & Hanin, B. (2022). *The principles of deep learning theory* (Vol. 46). Cambridge, MA, USA: Cambridge University Press.
- [5] Cabannes, V., Dohmatob, E., & Bietti, A. (2023). Scaling laws for associative memories. arXiv preprint arXiv:2310.02984.
- [6] Hanson, S., & Pratt, L. (1988). Comparing biases for minimal network construction with back-propagation. *Advances in neural information processing systems*, 1.
- [7] Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.
- [8] Nanda, N., Chan, L., Lieberum, T., Smith, J., & Steinhardt, J. (2023). Progress measures for grokking via mechanistic interpretability. arXiv preprint arXiv:2301.05217.

6. Appendix

A. Configuration Settings

TransformerLens configuration

d_model: 64
n_ctx: 10
d_head: 16
n_heads: 4
n_layers: 4
d_mlp: 4*d_model
d_vocab: 3
act_fn: relu
attn_only: True
attention_dir: causal
normalization_type: LN
seed: 42

Optimizer configuration

optimizer: AdamW
learning_rate: 0.0003
weight_decay: [various]

Training configuration

batch_size: 64
num_epochs: 500
fixed_train_dataset: False # stream random samples
num_train_samples: 10000 # per epoch
fixed_valid_dataset: True
num_valid_samples: 2000

Process configuration

process: Mess3
x: 0.15
a: 0.6