Identifying Subnetworks

Sondre Wold

June 7, 2024

Abstract

Studies of the hidden representations of deep neural networks are commonplace in machine learning research. A long-standing problem within the field has been to determine whether or not models learn modular representations that are specialized. For example, if a model is trained on an arithmetic dataset, is there a subnetwork in the model that is responsible for addition? This document tries to synthesize existing methods for identifying such subnetworks under a shared vocabulary.

1 What is a subnetwork?

Csordás et al. (2020) uses the terms *modules* and *subnetwork* interchangeably to refer to subsets of a models' weights that are responsible for performing a specific target functionality, such as solving subtasks of a larger and more complex problem. The same functional definition is used in Lepori et al. (2023) but there under the name *subnetwork*. For example, given a base model M parametrized by θ, a subnetwork S is implemented as a binary mask m over θ such that $S = M_{\theta \odot m}$, where $\odot$ is elementwise multiplication and m is trained on samples that pertain to one functional aspect of a task, such as addition in an arithmetic dataset.

Watanabe (2019) uses the same functional framework, but uses the term *cluster* to refer to a set of feature vectors that are the most influental on the output of a model for a set of specific inputs. This term is also used by Casper et al. (2022), who defines a cluster to be a subset of the network when viewed as a graph structure (with neurons being the node abstraction). These clusters are also analyzed with respect to their *local specialization*: does certain clusters translate to functional abstractions?

2 Identification methods

2.1 Masks

2.1.1 Differentiable weight masks

Csordás et al. (2020) proposes a method for training binary weight masks for all the weights of any NN. Their method first formulates a target task that corresponds to a specific function. This task is typically a subtask of a larger task. Next, they train a mask on this target task while keeping the original weights of the network frozen. The resulting mask reveals the module responsible for solving the functionality isolated in the target task.

The mask is initialized as learnable logits $l_i \in \mathbb{R}$, where $i \in [i, N]$ for N weights that parameterize model M. l_i is set to 0.9 for each i in order to have a high probability of keeping the weight. During training, l_i is regularized such that the probability for weight w_i not being masked out during inference is high if w_i is necessary for solving the subtask. The regularization term r is set as $r = \alpha \sum_i l_i$, where α is a hyperparameter that controls the strength of the regularization. The mask training procedure is based on sampling. For each l_i, a sample $s_i \in [0, 1]$ is drawn from the mask as follows:

$$s_i = \sigma((l_i - \log(\log U_1 / \log U_2)/\tau), \tag{1}$$

with $U_1, U_2 \sim U(0, 1)$, and where τ is a hyperparameter and σ is the sigmoid function. s_i is then gated to become the final binary mask, b_i. This is done with a straight-through estimator, which allows for estimating the gradient of threshold functions—like the one needed here to turn the continuous s_i into the discrete b_i.[1] The authors sample 4-8 binary masks per batch and apply it to different parts of the batch. After training, the mask is applied to M through elementwise multiplication of the mask with the original weights $w_i \odot b_i$.

Lepori et al. (2023) uses almost the exact same approach as Csordás et al. (2020) but with a different, and much simpler, masking technique. They use a pruning technique called *continuous sparsification* (Savarese et al., 2020), which they claim is both deterministic and better at finding sparser subnetworks than the one used in Csordás et al. (2020). This method uses l_0 regularization, where the training incentivize a weight mask to have as many zero-elements as possible. Given a model M parameterized by θ that is trained to solve a task, you first initialize the mask m as $\hat{\theta} = \theta$. You

[1] There was a lot of details here that I did not quite understand, but I think this should explain the gist of it at least

then minimize the following objective function on samples that target the functionality you wish to discover a subnetwork for:

$$\min_{\hat{\theta} \in \mathbb{R}^d} L\left(M_\theta(x)\right) + \lambda * ||\sigma(\beta * s_i)||_1, \tag{2}$$

, where L is the cross entropy of model M's predictions for input sample x, parameterized by weights θ, and λ is a hyperparameter that controls the balance between the loss and the number of zero-elements in θ. After mask training, the mask is made binary and applied elementwise with the original network through a heaviside function, substituting $\sigma(\beta * s_i)$ with:

$$H(S) = \left\{ \begin{array}{l} 0, s < 0 \\ 1, s > 0 \end{array} \right\}. \tag{3}$$

This part is not accurate and needs to be revised substantially.

2.1.2 Lottery Ticket Sparse Fine-Tuning

Another way of identifying masks is the Lottery Ticket Sparse Fine-Tuning approach proposed by Ansell et al. (2022). After finetuning a pretrained network on a target task, they identify the subset of parameters that changed the most during this training phase. Given a pretrained model M parameterized by θ^0, finetuning M on a target task yields the parameters θ^1. Parameters are then ranked according to their greatest absolute difference: $|\theta_i^1 - \theta_i^0|$. A binary mask is then constructed by selecting the top K parameters and setting all weights in M that correspond to these to 1 and the rest to 0.

This method is similar to the algorithm presented in Frankle and Carbin (2019), where they identify subnetworks that are initialized in a way that makes them able to achieve roughly the same task accuracy as the overall network, often at 10% of the size (The Lottery Ticket Hypothesis).

2.2 Clustering

Watanabe (2019); Casper et al. (2022)

References

A. Ansell, E. Ponti, A. Korhonen, and I. Vulić. Composable sparse fine-tuning for cross-lingual transfer. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1778–1796, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.125. URL https://aclanthology.org/2022.acl-long.125.

S. Casper, S. Hod, D. Filan, C. Wild, A. Critch, and S. Russell. Graphical clusterability and local specialization in deep neural networks. In *ICLR 2022 Workshop on PAIR {\textasciicircum} 2Struct: Privacy, Accountability, Interpretability, Robustness, Reasoning on Structured Data*, 2022.

R. Csordás, S. van Steenkiste, and J. Schmidhuber. Are neural nets modular? inspecting functional modularity through differentiable weight masks. In *International Conference on Learning Representations*, 2020.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

M. Lepori, T. Serre, and E. Pavlick. Break it down: Evidence for structural compositionality in neural networks. *Advances in Neural Information Processing Systems*, 36:42623–42660, 2023.

P. Savarese, H. Silva, and M. Maire. Winning the lottery with continuous sparsification. *Advances in neural information processing systems*, 33: 11380–11390, 2020.

C. Watanabe. Interpreting layered neural networks via hierarchical modular representation. In *Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part V 26*, pages 376–388. Springer, 2019.