Learning with random weights

Deep learning has been successfully applied to many pattern recognition tasks. However, deep learning usually necessitates large weight matrices that are finely tuned to the task. Tunable weights are expensive in hardware systems and novel neuromorphic hardware provide massive amounts of random weights. In this project we investigate learning for systems with random fixed weights.

We start with simple fully-connected feed forward networks. Given is a network N with fixed weights. For each neuron, we are allowed to remove connections (setting the weight to 0) and to re-route the incoming remaining connections to those inputs that achieve optimal performance.

To solve this problem, we first train a network N' to obtain a network with a given sparsity. This could be achieved with a pruning algorithm [1] or with a rewiring algorithm [2]. These algorithms typically use an L1 norm regularizer to achieve sparse connectivity. We then compute a mapping from N' to N. For each neuron i in N', we have to find one neuron j in N where we use some of the weights of neuron j, route them to the optimal inputs, and set the other weights to zero.

More formally: Consider a neuron i in N' with weights $\boldsymbol{w'}_i = (w'_{i,1}, \dots, w'_{i,n})$, where n is the number of inputs to the neuron. Since N' is sparse, $\boldsymbol{w'}_i$ has many zero-entries. Let $I_i = \{i_1, \dots, i_{n_i}\}$ denote the index set of non-zero entries, where n_i is the number of non-zero weights of neuron i.
For each neuron i in N' we have to find a neuron j in N and a permutation of indices $P = (p_1, \dots, p_n)$ such that

$$\left(w'_{i,i_1}, \dots, w'_{i,i_{n_i}}\right) \approx \left(w'_{j,p_1}, \dots, w'_{j,p_{n_i}}\right).$$

As a first proof of concept we will test this approach on MNIST.

Fine-tuning: To further optimize performance, we can assume that there is also a smaller number of fully tunable weights available. One could later add theses weights and fine-tune the network.

Goals & Tasks

- Review literature on network pruning and rewiring.
- Train such networks on simple tasks, find a good mapping strategy, and compare performance to standard training.

Qualifications

- Prior knowledge in deep learning.
- Experience with Python and Tensorflow or PyTorch.
- Course Deep Learning is recommended.
- Registered to one of the following:
  - ✓ **Bachelor Thesis**
  - ✓ **Seminar Project**
  - ✓ **Master Thesis**

Contact

Robert Legenstein
robert.legenstein@igi.tugraz.at

[1] Frankle, J., & Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635.
[2] G. Bellec, D. Kappel, W. Maass, and R. Legenstein. Deep rewiring: training very sparse deep networks. International Conference on Learning Representations (ICLR), 2018