Eigensolver Report

Jason Cheng

February 15, 2025

Implementation

My implementation closely follows the ideas in the lecture notes/the paper (Cerezo 2022). The important functions are

- `ansatz` — Implements $V(\theta)$ as described in the paper
- `cost` — Calculates the cost function according to the formula in the lecture notes
- `grad` — Calculates the cost function gradient according to the formula in the paper
- `read_ansatz` — Reads the eigenvectors from $V(\theta)$
- `train` — Implements gradient descent optimization
- `train_incremental` — Train layers incrementally rather than all together

`cost` and `read_ansatz` both use experimental results to estimate the vectors being read. Since it's based on probability, that means the eigenvectors produced by `read_ansatz` are missing their signs.

`train_incremental` is an experimental training approach. It repeatedly adds new layers and optimizes only the new layers, holding the parameters for the old layers constant. This improved efficiency, but at the cost of some accuracy.

For gradient descent, I used PyTorch's `SGD` optimizer, which is gradient descent with momentum. I hoped that the momentum would help prevent getting stuck in local minimums.

Experiments

I evaluated my results based on two metrics: cosine similarity between the measured eigenvectors and the true eigenvectors (obtained with `np.linalg.eig`), and the magnitude of the gradient. This helps determine the cause of bad performance. For example, if gradient magnitude converges while similarity fails to converge, we could be stuck in a local minimum.

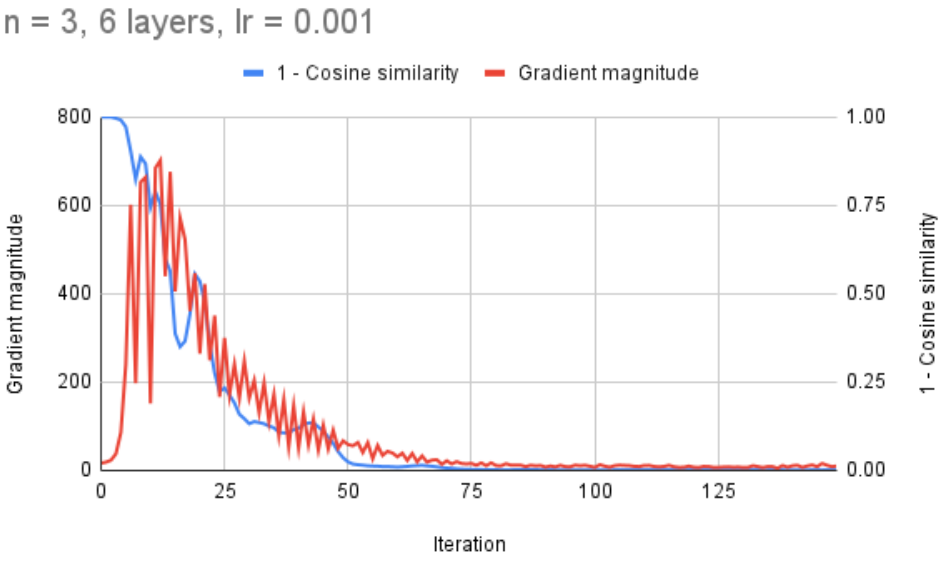
Hyperparameters

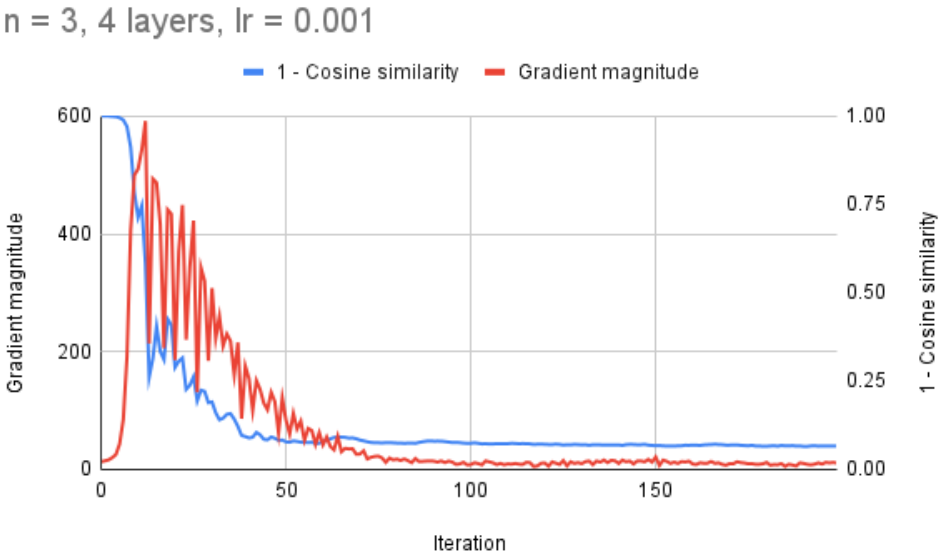
ID	Qubits	Layers	Iterations	Learning rate	Similarity
1	2	2	100	0.01	0.9997
2	3	6	150	0.001	0.9986
3	3	4	200	0.001	0.9338
4	3	$4 + 2$	$100 + 100$	0.001	0.9834
5	4	16	200	10^{-4}	0.9732
6	4	4	1000	10^{-5}	0.9322
7	4	8	1300	10^{-6}	0.8239
8	4	$4 + 4 + 4 + 4$	$200 + 200 + 200 + 200$	$10^{-4} + 10^{-4} + 10^{-5} + 10^{-5}$	0.9247

TBD: hasn't finished running

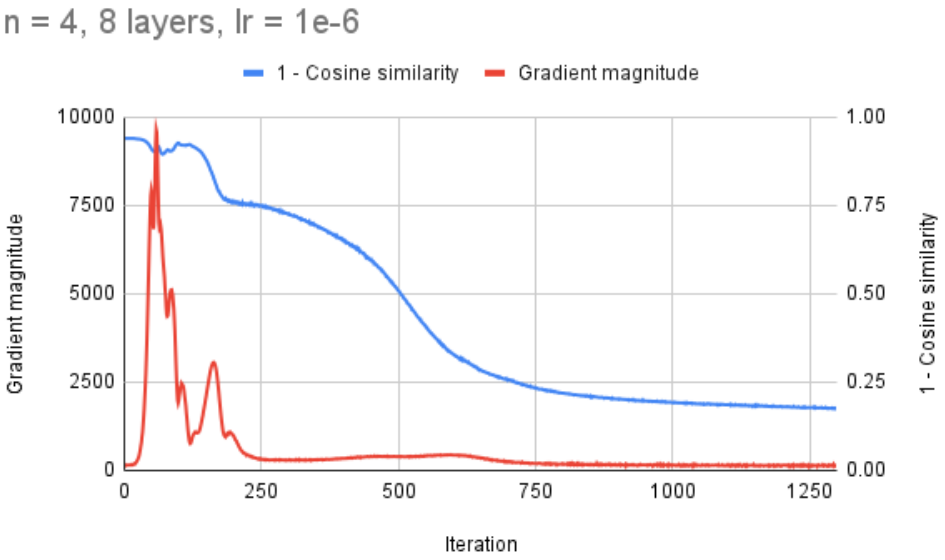
Discussion of results

Experiment 3 shows what happens when we decrease the number of layers. In the paper, they only used 3 layers for up to 10 qubits, so I wondered if I could replicate the same results using fewer layers. It turns out having fewer layers decreases accuracy. Here are the training graphs of experiment 2 and 3:

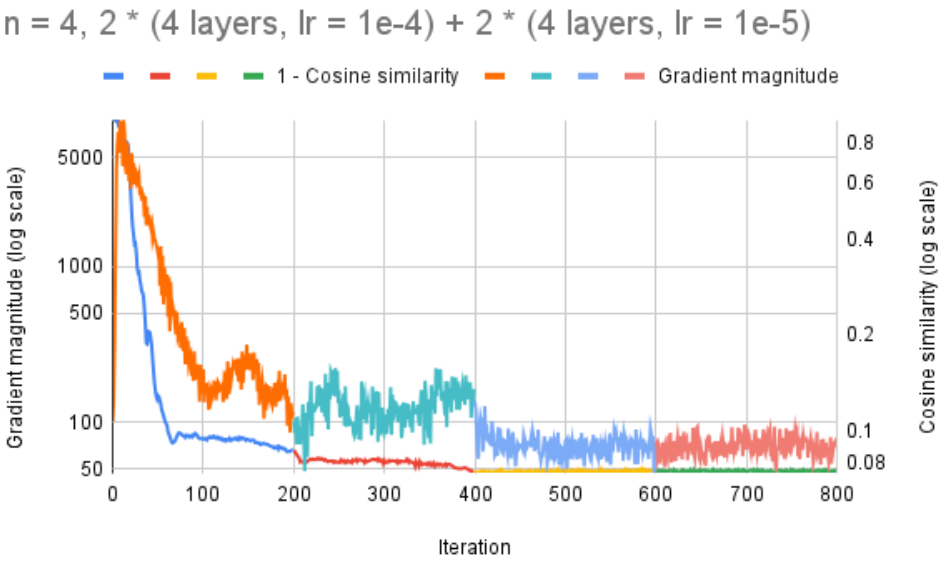




We can see that the gradient converges to around the same value, but the similarity fails to converge. This might be because there are fewer parameters, so each parameter is more affected by the gradient. Thus, a smaller learning rate might fix the problem. In experiment 6, I tested this theory by decreasing the learning rate by a factor of 0.1 compared to experiment 5, but I still had a lower accuracy. In experiment 6, I tried to decrease the learning rate even further but was not able to finish training because it took so long. The similarity was still increasing slightly when I stopped the simulation.



Experiment 8 shows the effect of my incremental training approach:



Each color represents a set of 4 layers. The first two sets were trained with learning rate 10^{-4} and the second two sets were trained with learning rate 10^{-5}. The accuracy isn't as good as experiment 5, where I trained all 16 layers together. However, it was much more efficient because I was only training 1/4 of the parameters at once. Experiment 5 took 2 days to train, while experiment 8 finished in a few hours.