Hash Layers for Large Sparse Models

Main Idea: this paper experiments with using a hash router in an MoE architecture and compares it to other methods like top-k (Switch) and BASE.

Hashing

- Hashing refers to using a function that converts the input into a fixed size output that is unique for each input.
- Hashing does not need to be learned and there is no need for a load balancing loss.
- In a setting where the goal is to scale a model as much as possible (MoE), it is not realistic to try to optimize every hyperparameter and modeling decision (extensive tuning at this scale is too expensive). Hashing does not require this.
- Hashing is also a fixed mapping function, meaning it does not suffer from the issue from routing fluctuation/inconsistency during training (this was explored in the StableMoE paper).
- Some hashing functions used:
    - Clustered hashes – hash training inputs based on k-means clustering.
    - Dispersed hashes – assume the opposite of clustered hashes, that similar inputs need a more fine-grained distinction and should be assigned to different experts (closer inputs should be routed to different experts).
    - Random hashing.
    - Balanced assignment hashing.

    - Oracle future hash – obtains a hash to route token t based on token t+1 (the next token).
- This paper also experiments with what they call MultiHashLayer, which consists of using different hashing strategies in the same network, as to not rely on a single hashing strategy.

<u>Models used in Experiments</u>

- Baseline is a 222M parameter dense Transformer.
- Wider dense Transformer of 755M parameters.
- Deeper dense Transformer of 755M parameters.
- To compare to BASE, a 4.5B total parameter architecture with balanced assignment hashing is used.

<u>Results</u>

- When using a single MoE layer in a Transformer architecture (all other FFs remain the same), balanced Hash slightly outperforms Switch (using 64 experts in a model of 751M total parameters).
- Deep dense models of the same size as MoE (in terms of total parameters) outperform MoE, showing good dense models make better use of each parameter. At the same inference speed (active parameters), MoE performs better.

- Increasing the number of experts (from 64 to 128) leads to a better increase in Hash over Switch.
    - This indicates that the more experts there are in a layer, the less important it is to learn to route.
- As with BASE layers, adding a single Hash layer to a Transformer is shown to work better at later layers of the network.
- Increasing the number of sparse layers (in a setting where dense FFs and MoE layers are alternated) (5 sparse layers with 16 experts each) leads to better Switch performance over Hash.
    - Increasing the number of experts per layer might change this.
- Fine-tuning trends are consistent with pre-training trends.
    - The only part that can be frozen without hurting performance are the sparse layers.

<u>Analysis/Results of Hashing Strategies</u>

- Random and balanced hashing have similar performance (but balanced hashing has training advantages over distributed training schemes).
- Random hashing outperforms clustered hashes.
    - Proves the hypothesis that if tokens are like each other, a more fine-grained distinction is needed, and the tokens need to be routed differently.

- Dispersed hashing (opposite of clustered hashing) performs slightly better than random hashing.
- Learned routing (like BASE or Switch) generally provide clustered expert modules, which could be a disadvantage based on the results obtained during this research.
- Bigram and previous token hashing perform worse than just relying on the current token.
    - This indicates that using the previous token to help with routing is harmful.
- Increasing the dictionary size used for tokenization (thus increasing the number of possible hashes) leads to a decrease in performance against Switch.
    - This indicates that Hash might be better suited for scenarios where the dictionary size is small (so there are less possible hashes), while Switch is better suited to large dictionary size scenarios.
- Oracle future token hashing essentially solves the task.
    - This is expected since the hashing is performed on the target token (the answer).
- Increasing the diversity of hashing strategies (MultiHashLayer) seems to help.
- A learned routing based on the current token (and not on the hidden state, as Switch routing works) leads to small improvements.
    - This is a mix between the hashing strategy and Switch.
- When comparing Hash vs BASE, Hash outperforms BASE in every training step. BASE also shows instabilities at late stages of training, while Hash's performance consistently improves (due to fixed routing).

Conclusion

Hash shows that there are lots of room for improvement in learned routing strategies. Hash should be used as a baseline for improving learned strategies in future work.

My takeaways:

- Why is it that random routing outperforms clustered hashes and dispersed hashing performs even better? Shouldn't clustered hashing make more sense since we want experts to specialize in specific clusters of the input space? These results seem to indicate the opposite.