Learned Index : Point Index

Rahul Palamuttam
Stanford University

Introduction

Neural networks could improve on existing hashing schemes found in databases by learning properties of the data being indexed. A recent paper "The Case for Learned Index Structures" argues that indexes are really models which can be approximated by neural networks. In short, Deep Learning could achieve better hash table utilization when storing indices.

Problem Statement

Point indexes are simple and its goals are exactly the same as a hash function. We want to map one set of data to unique buckets and minimize collisions. We explore the challenges in using neural networks to learn a specific type of index: a point indexes.Following the example set by Kraska et al. we seek to implement the recursive index model to map timestamps (our input) to unique indices (our output) in a hash table.

Methods & Models

We utilized the recursive index model (RMI) presented in the paper. It is similar to a B-tree structure in that a model at stage ℓ will pick a sub-model among k_ℓ models for the subsequent layer. The last stage predicts the position. The models at each stage are shallow neural networks. We compute the l2 loss at each stage. The equations governing the recursive index model is given below.

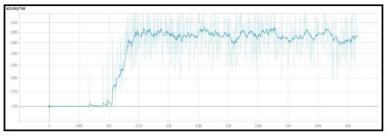
$$L_\ell = \sum_{(x,y)} (f_\ell^{(\lfloor M_\ell f_{\ell-1}(x)/N \rfloor)}(x) - y)^2$$

$$L_0 = \sum_{(x,y)} (f_0(x) - y)^2$$

Results

Data	1 Layer	RMI
Weblog	10%	55%
Linear	65%	45%
Quad	45%	35%

Recursive Index Model Accuracy



Discussion

Unlike traditional goals of a model learning from a training set and generalizing well on a test set, the goals here were to literally memorize the training set.

While we were able to achieve a high threshold on the synthetic linear dataset (as expected) a real world timestamp dataset was too hard to learn. We can closely approximate it, but as evidenced by a the cumulative distribution function an exact match would require many more iterations, and more stages.

Kraska et al. claims that the model itself trains in under an hour (minutes). However, after replicating the model as close to the paper as possible (clearly they utilize many more stages and more models per stage) the exact engineering effort to do this is missing from the paper details.

It's interesting to note that the recursive index model performs Worse on the linear and quadratic datasets than the simple 1 layer.

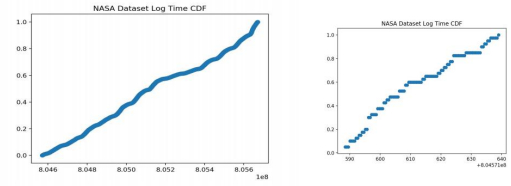
Training Parameters : AdamOptimizer, Learning Rate = 0.001, Shallow Networks of two hidden layer and 32 neurons.

Dataset

Nasa Webserver Logs (Time) - Two month's worth of all HTTP requests to the NASA Kennedy Space Center WWW server in Florida.

Features

1. **timestamp** Convert to epoch time
2. **request** given in quotes.
3. **HTTP reply code**.



Cumulative Distribution Functions differ in shorter ranges

Conclusions

- A strong case is made that neural networks could be used as index approximators in databases.
- While the work was inspired by the idea that indexes are really models, it could also be said that models are also indexes. By this reasoning we should be able to handle deletes as an operation that "undoes" a gradient update for a set of items being removed from the hashtable.
- Apart from asking the authors what exactly their hyperparameters were, I think for learned indexes to be of practice use there needs to be a way to account for inserts and deletes in a learned index model. This could be a promising next research goal for subsequent work.

References

Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, Neoklis Polyzotis: The Case for Learned Index Structures. SIGMOD Conference 2018: 489-504

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, Jeff Dean: Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. CoRRabs/1701.06538 (2017)

http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html