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 index. 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 t will pick a sub-model among its 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.

\[ L_2 = \sum \frac{1}{(a,b)} \left( f(a,b) - \frac{(a+b)}{2} \right)^2 \]

Results

<table>
<thead>
<tr>
<th>Data</th>
<th>1 Layer</th>
<th>RMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weblog</td>
<td>10%</td>
<td>55%</td>
</tr>
<tr>
<td>Linear</td>
<td>65%</td>
<td>45%</td>
</tr>
<tr>
<td>Quad</td>
<td>45%</td>
<td>35%</td>
</tr>
</tbody>
</table>

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 far too hard to learn. We can closely approximate it, but as evidenced by 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 “updates” a gradient update for a set of items being removed from the hash table.
- Apart from asking the authors what exactly their models try to learn indexes to be of practical 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, Nickola Polyzotis: The Case for Learned Index Structures. SIGMOD Conference 2018: 455-504