Upcoming of the learned data structures

Abhishek Tiwari

Citation: *A. Tiwari*, "Upcoming of the learned data structures", Abhishek Tiwari, 2017. doi:10.59350/rjcp2-vn266

Published on: December 14, 2017

Can machine learning-based data structures i.e. learned data structures replace traditional data structures? This is a question recently asked and explored by a team of Google researchers led by Jeff Dean with a major focus on database indexes. Jeff is a Google Senior Fellow in the Google Brain team and widely known as a pioneer in artificial intelligence (AI) and deep learning community. The initial results show that learned data structures can outperform existing data structures. Obviously, Jeff and team don't claim to completely replace traditional data structures with learned data structures, but their seminal work opens up an entirely new direction for the research in this space.

When we speak of learned data structures we are talking about the possibility to replace traditionally models with machine learning and deep learning models. This has far-reaching implications how future data systems and algorithms will be designed. Apart from indexes, super efficient sorting and join operations are some major areas come to my mind with immediate benefits of using learned data structure. More importantly, if this works out well, this could lead to a radical improvement in performance by leveraging hardware trends such as GPUs and TPUs.

Learned indexes

In their paper titled as The Case for Learned Index Structures Kraska et al. applied their approach of learning based model to B-Tree, Hash-map, and Bloom filter to develop corresponding learned models. Their result is particularly promising for specialized index structures termed as **learned indexes** which has potential to replace B-Trees without major re-engineering.

They demonstrated that neural nets based learned index outperforms cache-optimized B-Tree index by up to 70% in speed while saving an order-of-magnitude in memory. The benchmarking was performed using 3 real-world data sets (weblogs, maps, and web-documents), and 1 synthetic dataset (lognormal).

In many ways, traditional index structures such as B-Trees are like predictive models because if anything they are *predicting* the location of a value in the index given you have a key. According to Kraska et al.,

> B-Tree is a model, in ML terminology a regression tree: it maps a key to a position with a min- and max-error (a min-error of 0 and a max-error of the page-size) and the guarantee that the key can be found in that region if it exists. Consequently, we can replace the index with other types of machine learning models, including deep learning models, as long as they are also able to provide similar strong guarantees about the min- and max-error.

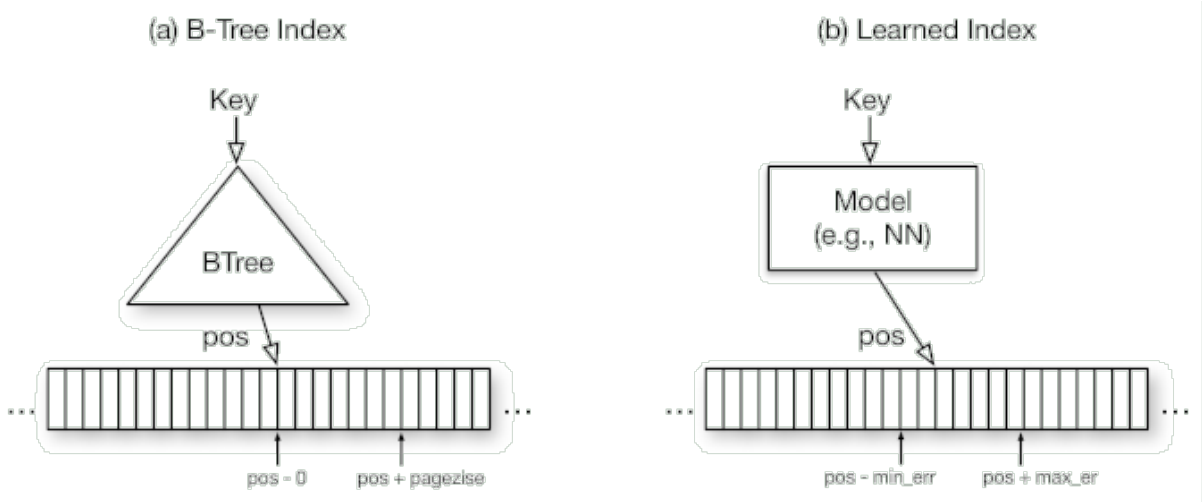

Figure 1: B-Trees are models

In this new landscape, indexes are modeled as cumulative distribution functions (CDFs) of the data i.e. a learned model “learns” the data distribution and predicts the position of a given key p.

```
p = F(Key) ⨯ N

where p is the position estimate, F(Key) is the estimated cumulative
   distribution function for the data to estimate the likelihood to
   observe a key smaller or equal to the lookup key P(X ≤ Key), and N is
   the total number of keys.
```

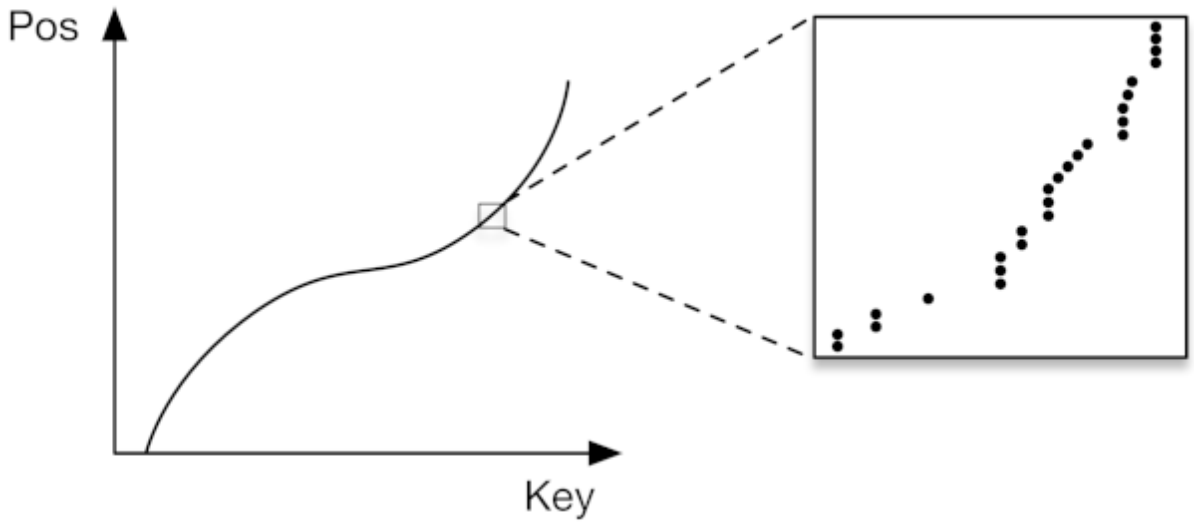

Figure 2: Indexes as cumulative distribution functions (CDFs). Learning the CDF plays a key role in optimizing learned index structures

Learned Hash-Maps

We can also use machine learning based models to replace traditional hash-maps. A learned hash-map can uniquely map every key into a unique position inside the array. Based on initial observations it seems that machine learning based models may not speed-up the hash-maps when compared to tradition hash-maps, but depending on the data distribution learned hash-maps can achieve higher memory utilization (*less than 20% wasted space at small overhead per record*).

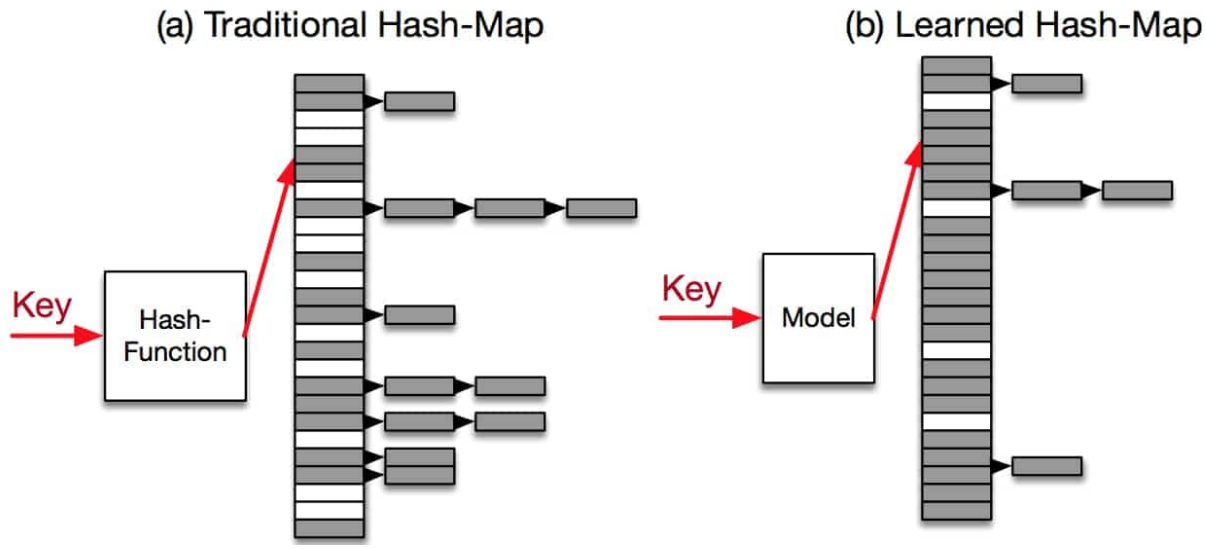

Figure 3: Traditional Hash-map vs Learned Hash-map

Learned Bloom filters

A learned bloom filter can either learn bloom filters as a classification problem or utilize models as special hash-functions. More importantly, learned bloom filter definitely improves memory footprint - roughly 2X space improvement over traditional bloom filter with the same false positive rate.

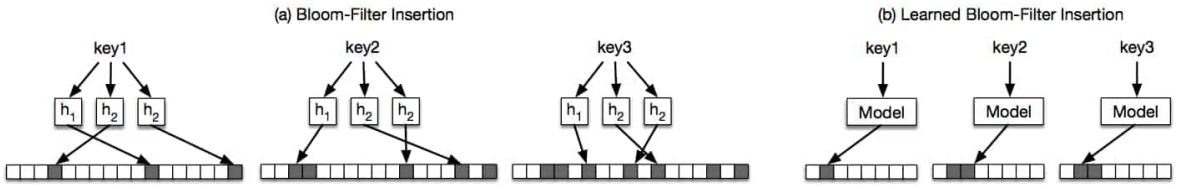

Figure 4: Bloom-filters with learned hash-functions

What's next

As mentioned earlier, this seminal research opens door to several new areas. For instance, possibility to extend the concept of learned indexes to multi-dimensional index structures. These areas benefit not only from neural networks ability to handle high-dimensional relationships but also recent hardware trends. It is fair to say that only due to GPU/TPUs the idea of learned indexes is more viable and they key reason to adopt them in practice. For instance, one can practically fit an entire learned index into the GPU's memory.

Bigger picture

Data structures are very foundational for the computer science and software engineering. These days, most of the algorithms and software logic is modeled around these data structures. Learned data structures have potential to disrupt the whole landscape, although it may be too early to say anything about to what extent. If AI has to write the software code by the first principle and not by piecing together lines of code taken from existing software, then apart from mastering learning to learn quite possibly AI also need to synthesize an entirely new landscape of learned data structures.