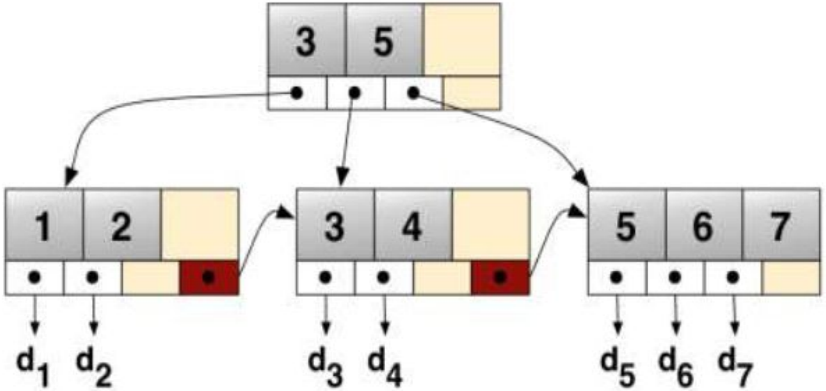
Build a learned index

Reed Callahan, Guanting Chen, Yuan Zhang,
4/30/2019

BACKGROUND

B-Tree Index



- Best choice for range requests
- Self-balanced binary search tree
- Lookup in O(log n)

CDF

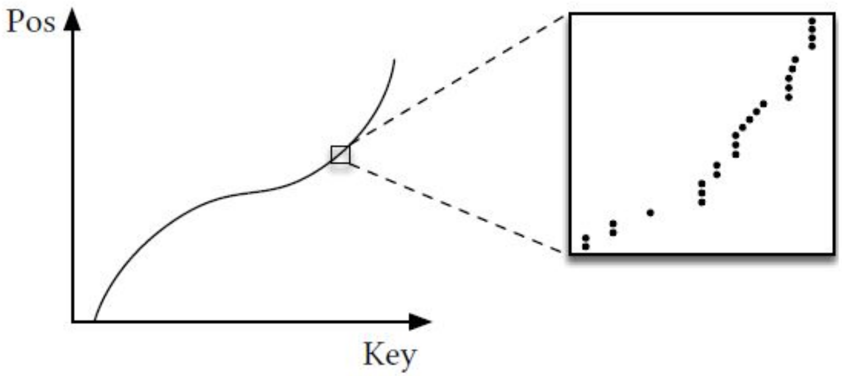


Figure 2: Indexes as CDFs

- Cumulative distribution function
- Range index model are CDF models
- Use ML models to learn CDF of data

THE PROBLEM

Traditional B-Tree Index

- Remain general purpose data structures
- Assume nothing about data distribution

The Problem

- If Knowing exact data distribution ⇒ Instance-based optimization
  e.g, lookups: O(log n) → O(1)

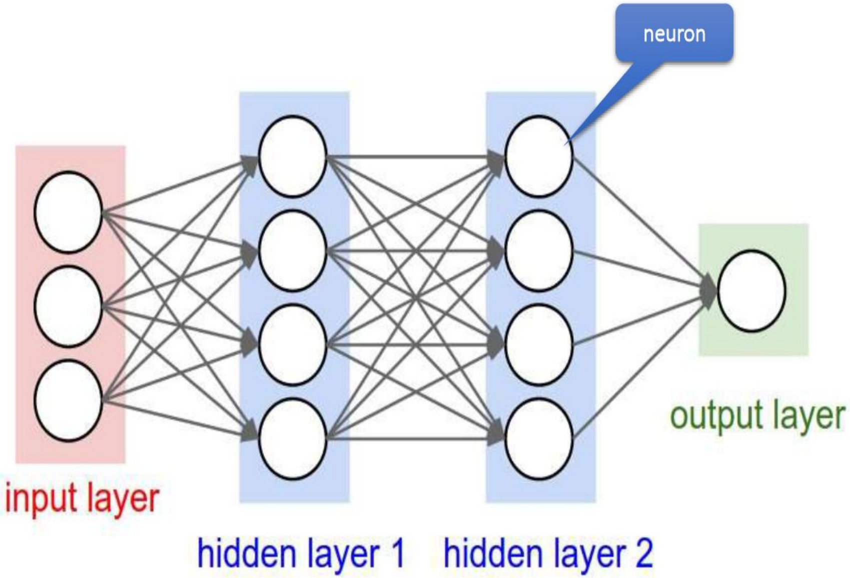
- Not for traditional B-Tree

  How?

SOLUTION

Learned Index

Neural Network



- The most powerful ML models
- However, require large amounts of data for their training due to vast amount of nodes
- Have parameters such as learning rates, activation functions, amounts of layers and # of neurons per layer that all increase complexity

Learned Index

- Built using neural network (feedforward, 2 layers, fully connected, ReLU activation function neurons)
- Trained using cross entropy (loss between probabilities of what it predicted vs actual)
- AdamOptimizer to update weights and minimize entropy
  - Based on stochastic gradient descent
  - Maintains a per-parameter learning rate that improves performance on problems with sparse gradients
  - Uses Root Mean Square Propagation for spikes in gradient descent, allowing it to work better in noisy problems

EVALUATION

Datasets

Data generator

- Generate synthesized dataset of 6 different distributions
- Distributiosn is based on unique integer keys.
- Receive the size dataset and block, and distribution of the data set
- Automatically output sample to a csv format file

Random	Binomial	Poisson	Exponential	Lognormal	Normal
range	N, P	lambda	scale	mean, sigma	mean, sigma

Dataset Example

RANDOM

Index	Random ID	Gender	Age	Zip Code	Partition
0	6021	Male	37	2638	0
1	8843	Female	57	9199	0
20	5340	Male	44	4145	0
26	3415	Female	47	3770	0
44	3072	Male	53	4107	0
45	8651	Female	48	8029	0
48	9734	Female	58	6932	0
49	5821	Female	43	6391	0
51	8865	Female	28	2682	0

LOGNORMAL

Index	Random ID	Gender	Age	Zip Code	Partition
33166	1120	Male	34	6722	0
41872	1438	Male	77	9587	0
43736	8808	Male	25	6395	0
50313	2094	Female	20	1454	0
51895	1007	Female	44	7447	0
58418	6424	Female	26	6701	0
62191	8071	Male	78	6086	0
65504	7976	Male	45	2345	0
67367	1056	Female	52	3805	0

Problem

```
NORMAL Raw dataset: 1000000
NORMAL Unique dataset: 372875
```

- Index should be based on unique interger keys
- Sampled distribution should be integral

Uniqueness and Integrality

- Scaler for float type result
- Filter and pad algorithm based on re-sampled result of same distribution

Scaler \ Distribution	Random	Binomial	Poisson	Exponential	Lognormal	Normal
Multiplier	1 x	1 x	1 x	1000000 x	10000 x	100000 x

Distribution \ Size	1000	10000	100000	1000000	10000000
Random(range = size*10)	≈ 95%	≈ 95%	≈ 95%	≈ 95%	≈ 95%
Binomial(N = size^2, P = 0.5)	≈ 80%	≈ 77%	≈ 77%	≈ 77%	≈ 77%
Poisson(lambda = size^2)	≈ 87%	≈ 87%	≈ 87%	≈ 87%	≈ 87%
Exponential(scale = 10)	≈ 100%	≈ 100%	≈ 100%	≈ 98%	≈ 80%
Lognormal(mean = 5, sigma = 1)	≈ 100%	≈ 100%	≈ 99%	≈ 89%	≈ 45%
Normal(mean = 5, sigma = 1)	≈ 100%	≈ 100%	≈ 87%	≈ 37%	≈ 58%

Evaluation

- Used two datasets of 1 million records each as main testing
    - Randomly and Exponentially distributed
- Saw impressive decrease in build time for random distribution, with less than 50% build time
- Exponential only had a 5% decrease in build time
- Look up times were still much higher with neural network
- In 3 million record dataset, maintained same time to construct

CONCLUSION

Conlcusion

- Our model was able to was able to construct indexes far faster in large datasets (upwards of 1 million records)
- However, search times were still lacking due to the unoptimized parameters, thresholds and variety of different models to utilize
    - Need more datasets and engineers to make this very optimized
- Neural network approach should be primarily used only with large datasets, as the base invocation cost and amount of data required for accuracy is high (2M+ records)
- As mentioned in the supporting paper, use of multiple models is important in building a flexible and reliable learned index application

Future Work

Future Work

- Dealing with error
  - Bound the error probability
  - Quickly recover strategy
    - Multiple models for different size/distributions
- Refining dataset synthesizing and distributions plotting
  - Scaling factor, large keys
- Tuning for updatable data distribution
  - Sampling dataset with less precision or integrality
  - Insert and append

THANK YOU