The Case for Learned Index Structures

Tim Kraska*
MIT
Cambridge, MA
kraska@mit.edu

Alex Beutel
Google, Inc.
Mountain View, CA
alexbeutel@google.com

Ed H. Chi
Google, Inc.
Mountain View, CA
edchi@google.com

Jeffrey Dean
Google, Inc.
Mountain View, CA
jeff@google.com

Neoklis Polyzotis
Google, Inc.
Mountain View, CA
npolyzotis@google.com

分享人：张腾

阿里巴巴数据库内核团队

2017-12-29

Outline

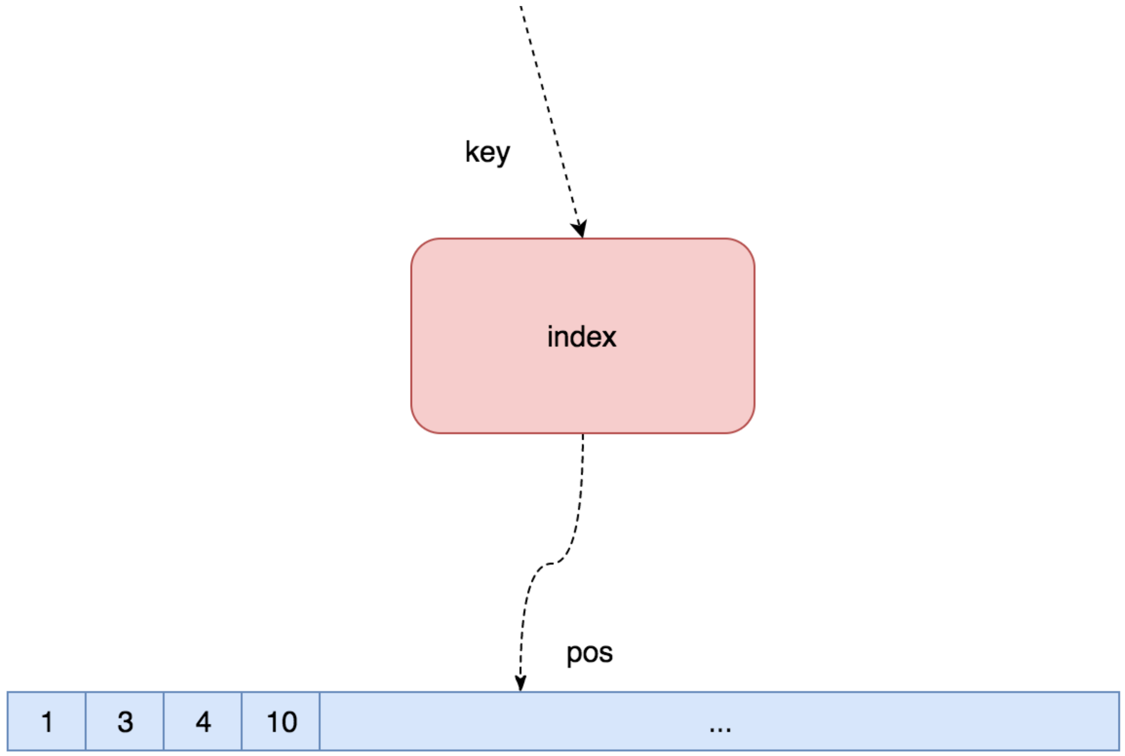
- Introduction
- Range Index
- Point Index
- Existence Index
- Related Work
- Conclusion and Future Work

Introduction

- Index structures are used to speed up queries

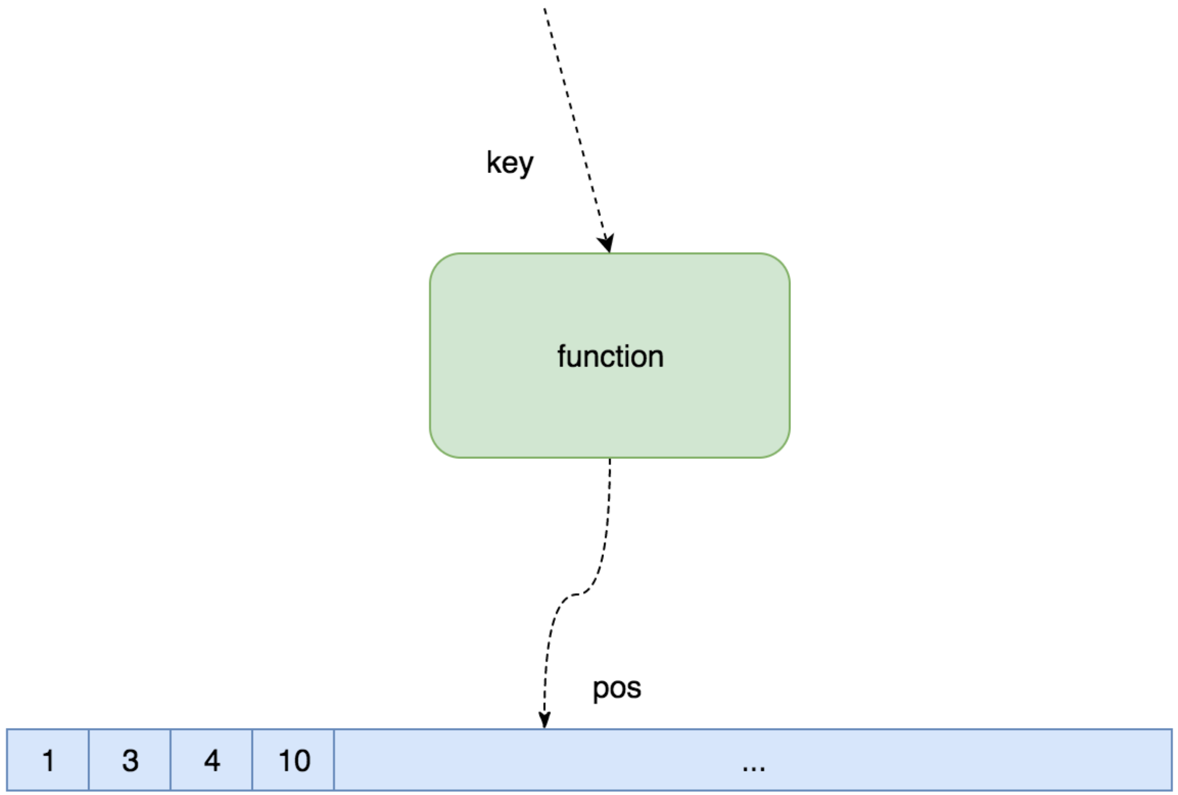
- Need to be stored and maintained

- O(logn) or O(1) time complexity



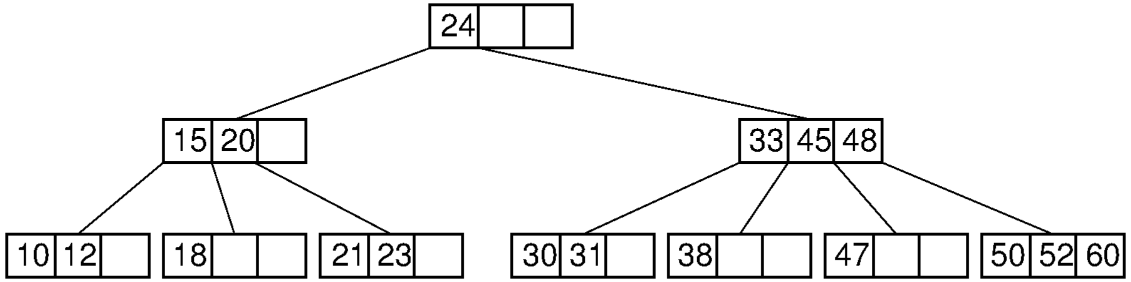
Introduction

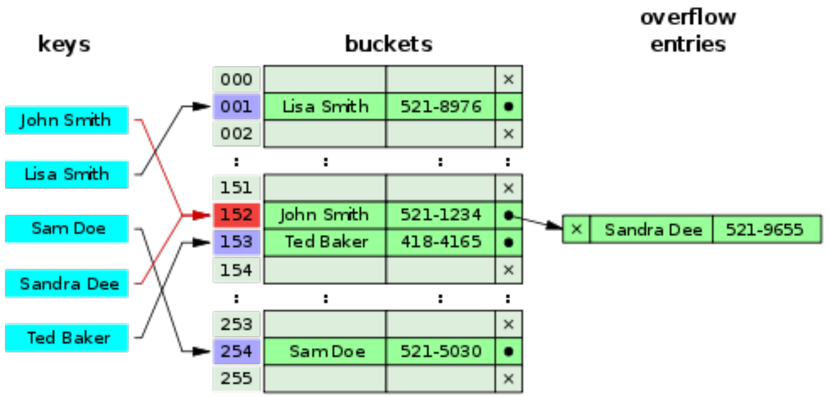
- No index structures any more
- Replaced by a model/function
- Always O(1) time complexity

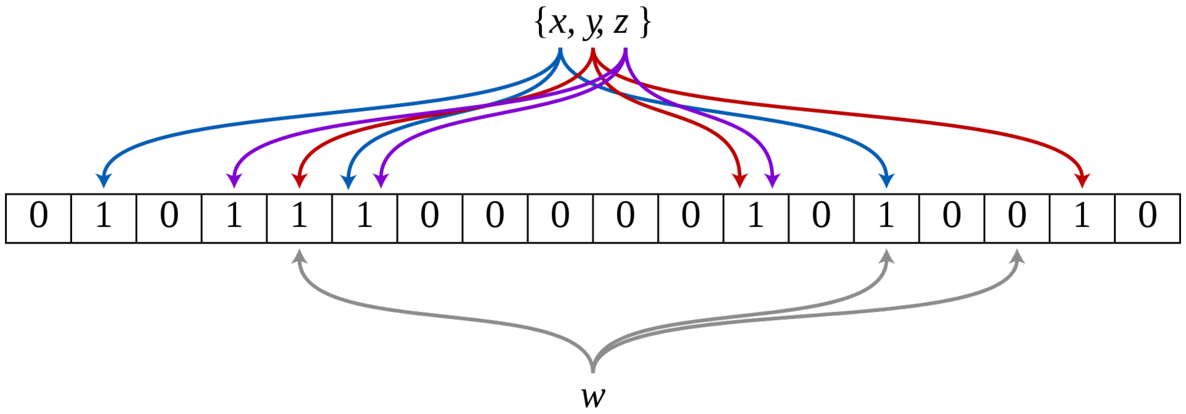


Introduction

- Different data access pattern leads to different index structures
  - Range queries -> B-Trees
  - Key-based lookups -> Hash-Maps
  - Record existence check -> Bloom-Filters

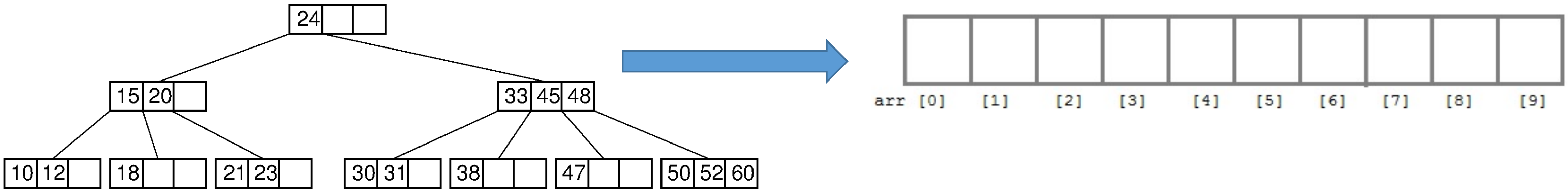






Introduction

- Given a known data distribution
  - Fix-length
  - Continuous Integer Keys
  - 1M ~ 100M



Introduction

- Learned Index
    - Data distribution counts

    - Index structures can be viewed as models
        - B-Tree -> key as input, position as output
        - Bloom-Filter -> Binary Classifier

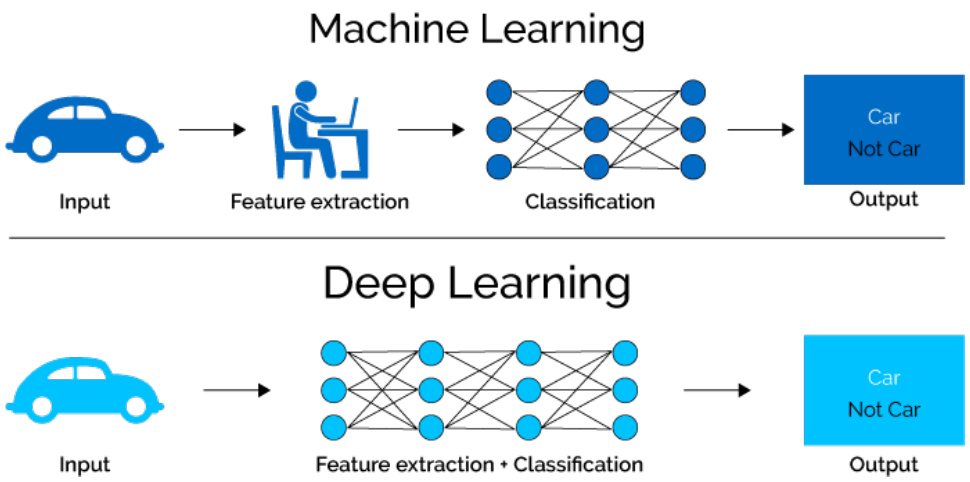
    - Next generation of new hardware
        - 1000x improvement of GPU by 2025

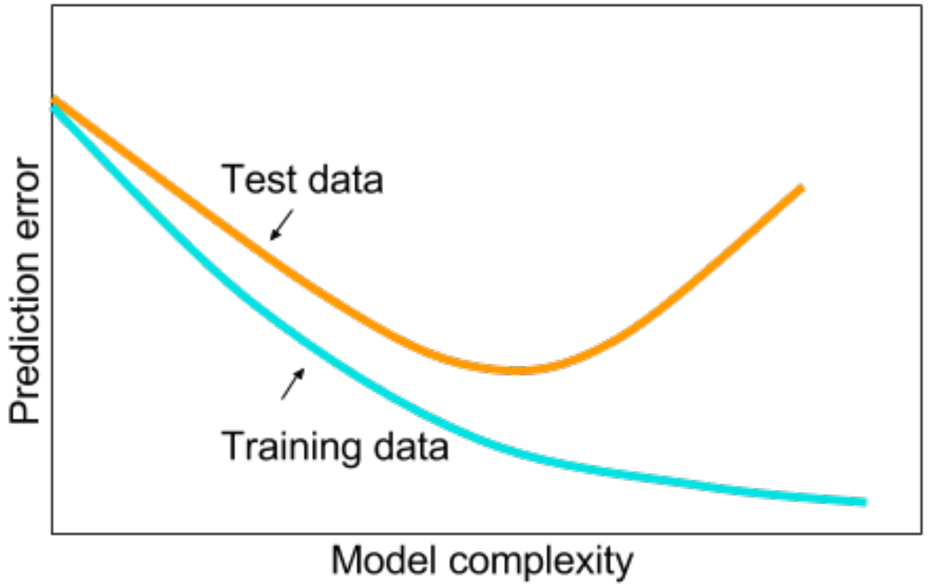
Introduction

- Backgrounds of Machine Learning
  - Task
    - Classification
    - Regression
    - Recommendation
    - Rank
    - Clustering
  - Training & Test
    - Generalization
    - Over-fitting
  - Metrics
    - Average error

Supervised learning

Unsupervised learning

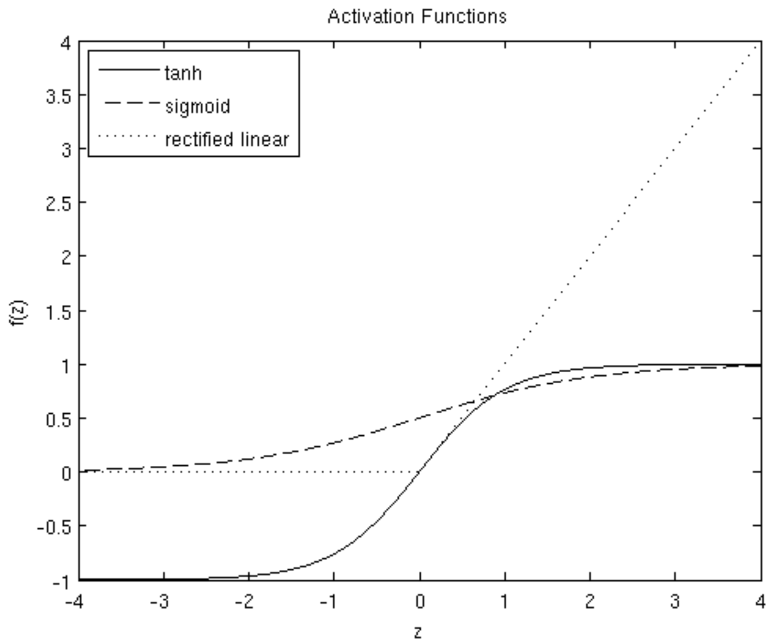


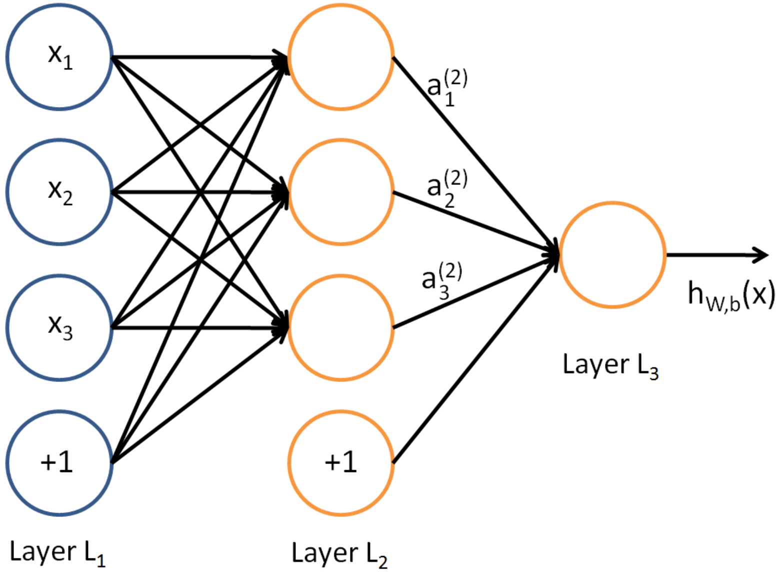


Introduction

- Backgrounds of Machine Learning
  - Model
    - Linear regression

    - Neural Network
      - Neuron
      - Input layer / hidden layer / output layer
      - Fully connected
      - Activation function
        - Sigmod
        - Relu
        - tanh
      - CNN、RNN、LSTM、GAN…



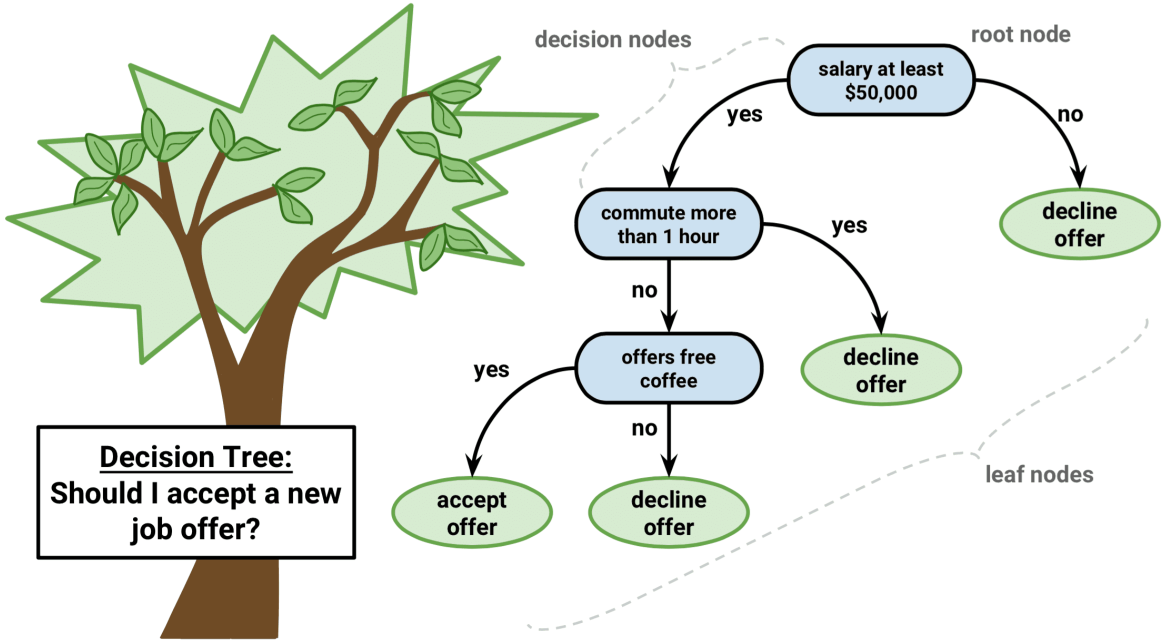


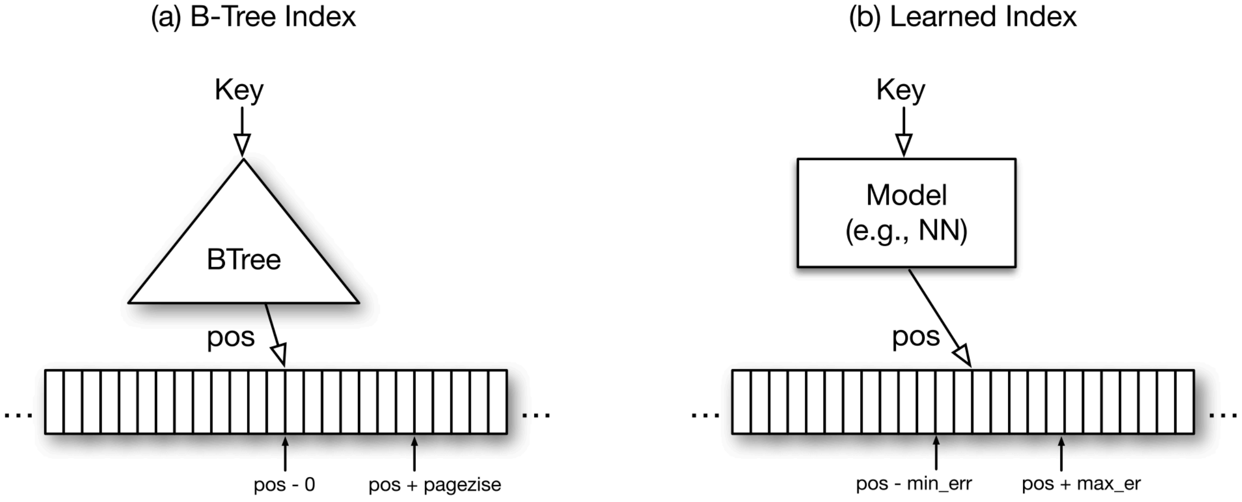
Outline

- Introduction
- Range Index
- Point Index
- Existence Index
- Related Work
- Conclusion and Future Work

Range Index

- B-Tree are models
  - Task: Predict the location of a value given a key
  - Decision tree
    - min-error : 0
    - Max-error : page-size
    - Guarantee that the key can be found
  - Sorted data
    - New data -> re-balanced
    - Re-trained

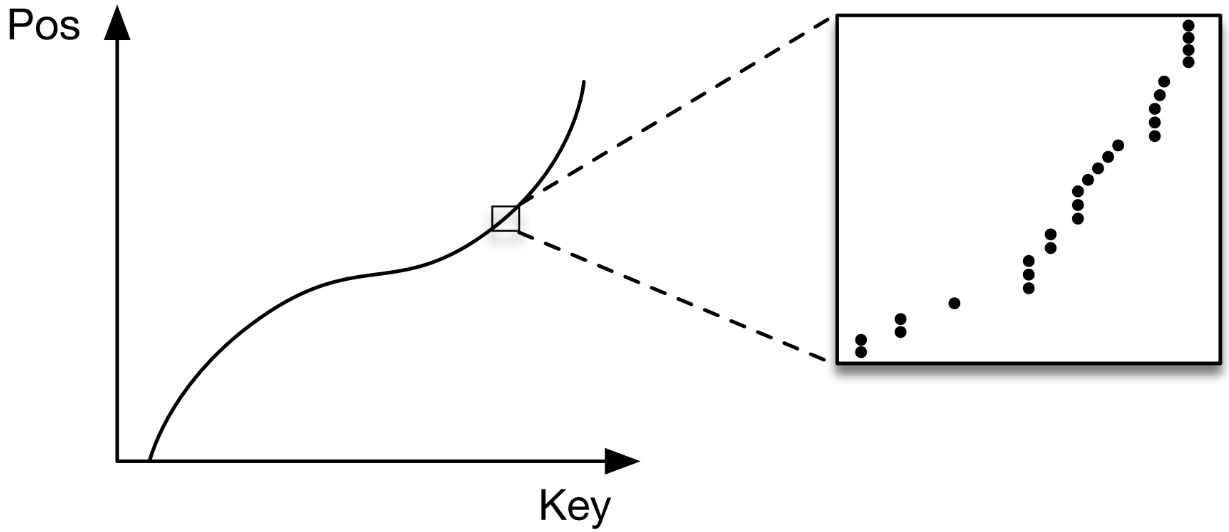




Range Index

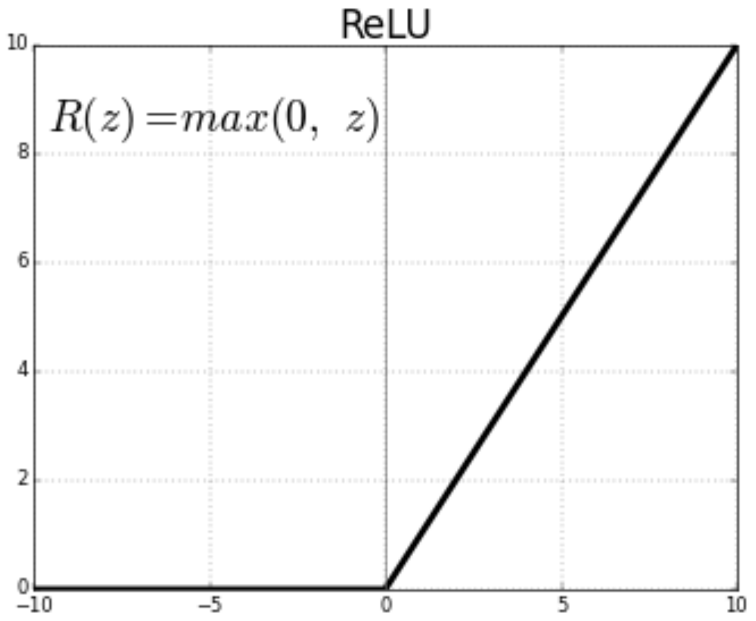
- Range Index are CDF models
  - Cumulative Distribution Function
  - B-Tree learns CDF by forming a regression tree
  - Linear regression model learns CDF by minimizing the RMSE of a linear function

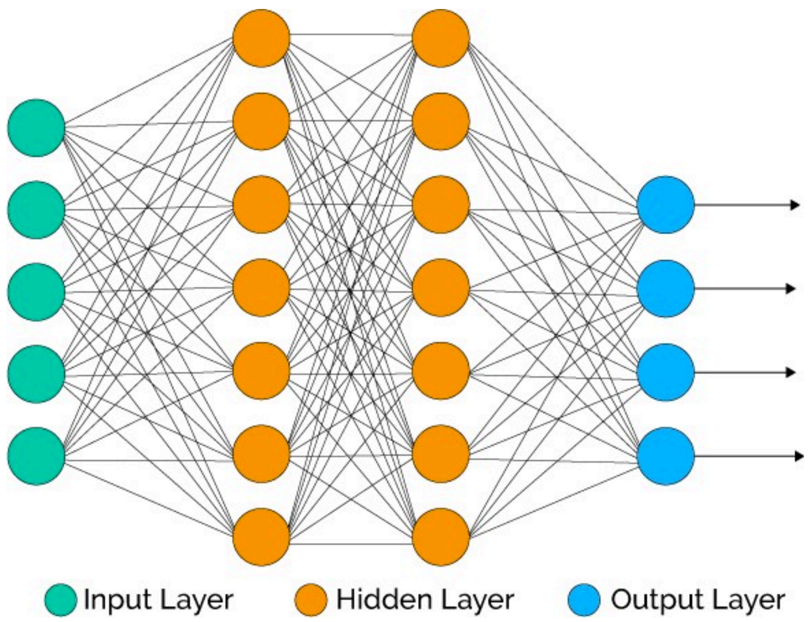
$$p = F(\mathbf{Key}) * N$$



Range Index

- A naïve learned index
  - two-layer fully-connected NN(width: 32)
  - Activation function: RELU
  - Tensorflow and python
- Performance
  - **80000 ns vs 300ns (model execution), no search time benefits**
- Reason
  - Invocation overhead
  - "Last mile accuracy" dilemma
  - Only consider average error
  - Not cache efficient





Range Index

- The Learning Index Framework
  - Index synthesis system
  - Tensorflow for training, C++ for inference
  - Execute simple models in 30ns

Range Index

- The RM-Index (Recursive Model Index)
  - Internal model -> pick model at next stage
  - Leaf mode -> predict position

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

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

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

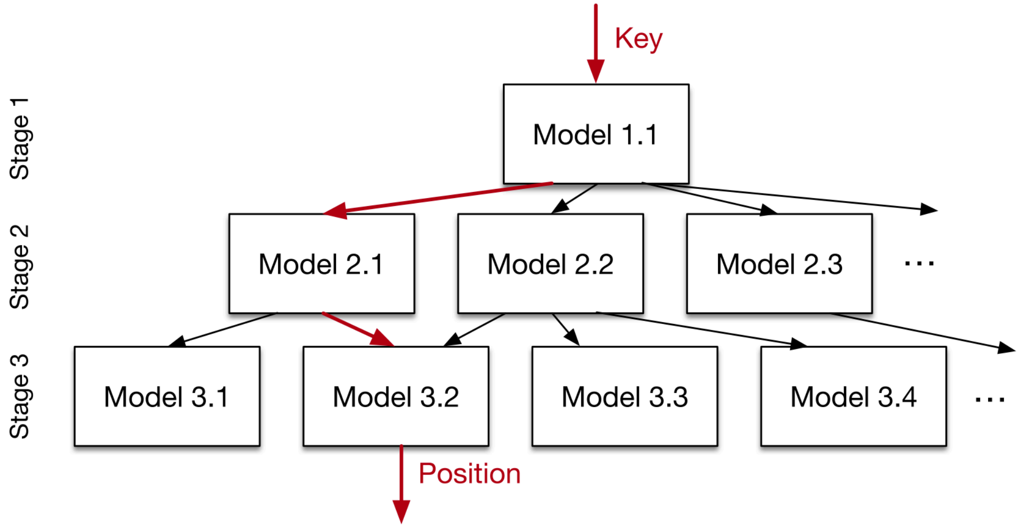


Figure 3: Staged models

Range Index

- The RM Index
  - Differences with B-Tree
    - Each model covers different number of records
    - Internal model output to pick the expert about certain keys
  - Benefits
    - Able to learn the overall shape of the data distribution
    - Divided into sub-range to improve “last mile” accuracy
    - No search process between stages

Range Index

- Hybrid Index
  - NN
  - B-Tree if absolute min/max-error is above the threshold
  - Bound the worst performance to the performance of B-Tree

Algorithm 1: Hybrid End-To-End Training

```
Input: int threshold, int stages[], NN_complexity
Data: record data[], Model index[][]
Result: trained index
M = stages.size;
tmp_records[][];
tmp_records[1][1] = all_data;
for i ← 1 to M do
    for j ← 1 to stages[i] do
        index[i][j] = new NN trained on tmp_records[i][j];
        if i < M then
            for r ∈ tmp_records[i][j] do
                p = f(r.key) / stages[i + 1];
                tmp_records[i + 1][p].add(r);
for j ← 1 to index[M].size do
    index[M][j].calc_err(tmp_records[M][j]);
    if index[M][j].max_abs_err > threshold then
        index[M][j] = new B-Tree trained on tmp_records[M][j];
return index;
```

Range Index

- Indexing strings
  - Tokenization
    - ASCII value
    - Vector as input

      $\mathbf{x} \in \mathbb{R}^n$
    - Complexity grows
      - Linear regression scales linearly *O(N)*
      - NN scales *O(hmN)* -- h (width), m(width)
    - Interaction between characters -> RNN

Range Index

- Search Strategies
  - Model Binary Search
    - *middle* point set to the value predicted by the model
  - Biased Search
    - Considering the standard deviation of the last stage model

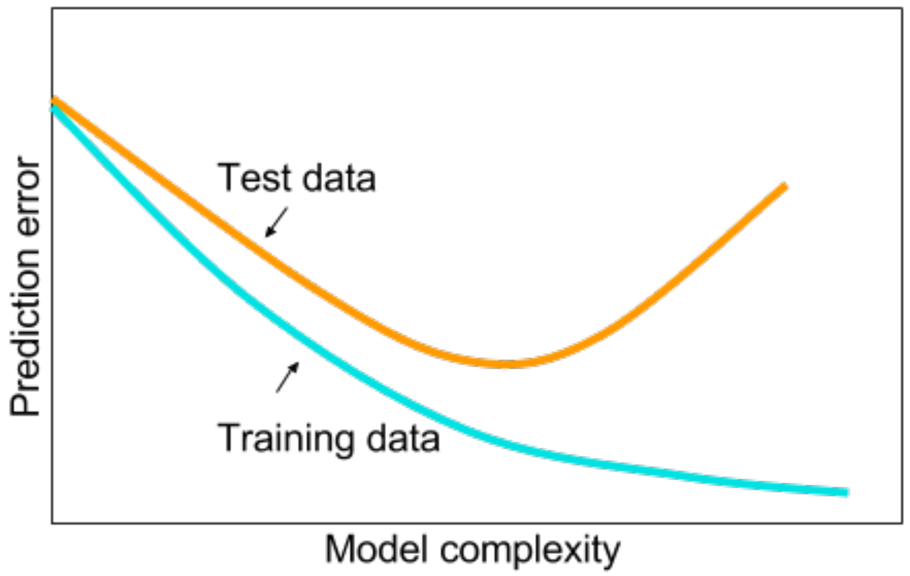
$$min(middle + \sigma, (middle + right)/2)$$

  - Biased Quaternary

$$pos - \sigma, pos, pos + \sigma$$

Range Index

- Inserts and updates
  - Appends/Insert in the middle
  - Generality vs Accuracy
  - Avoid over-fitting
  - Solutions
    - Spread the space dependent on CDF
    - Distribution change detection -> model split and retrain

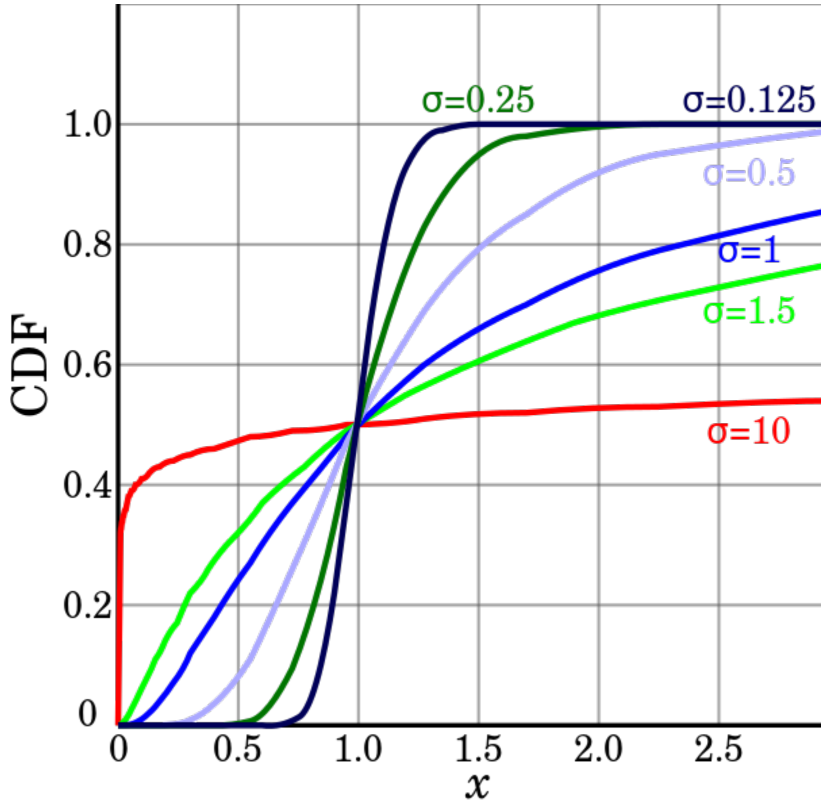


Range Index

- Paging
  - Disk-based system
  - Violation of CDF $p = F(X < \mathbf{Key}) * N$
  - Duplicate records for overlapped partition
  - Additional translate table
    - <first_key, disk_position>

Range Index

- The evaluation of RMI -- Speedup
  - Datasets (200M)
    - Maps (longitude of features)
    - Weblogs (University website request timestamp)
    - Lognormal distribution
  - Metrics
    - Space
    - Time (model execution + search)
    - Model error
  - Baseline
    - B-tree with page size 128



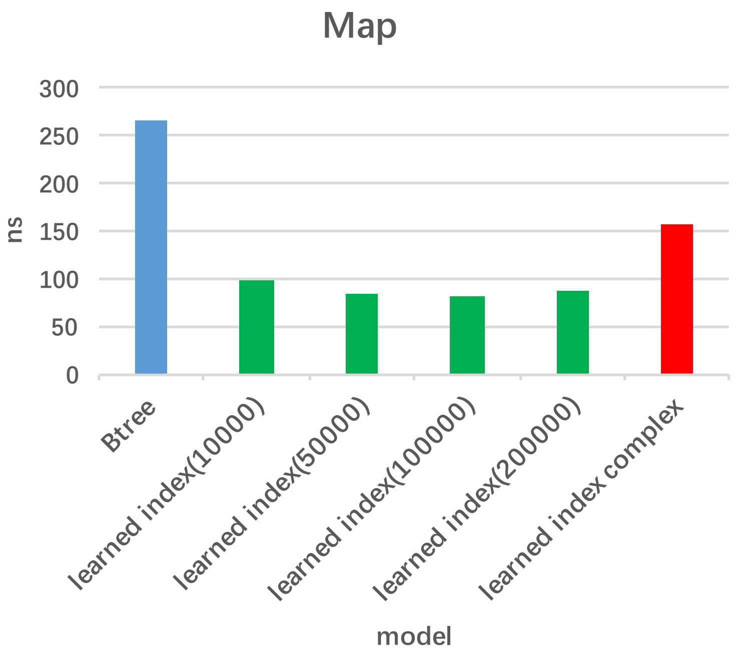
$$\frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\left[\frac{\ln x - \mu}{\sqrt{2}\sigma}\right]$$

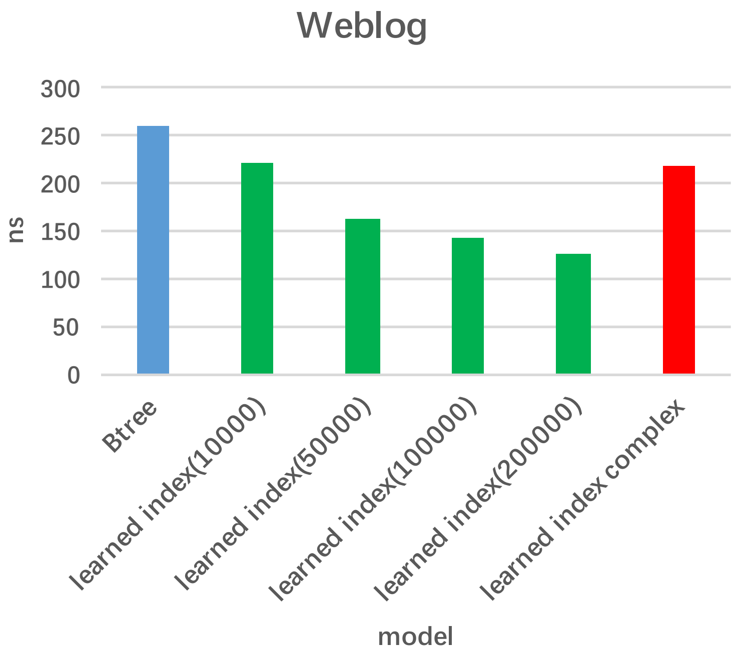
$$\begin{aligned}\mathrm{erf}(x) &= \frac{1}{\sqrt{\pi}} \int_{-x}^{x} e^{-t^2}\,dt \\ &= \frac{2}{\sqrt{\pi}} \int_{0}^{x} e^{-t^2}\,dt.\end{aligned}$$

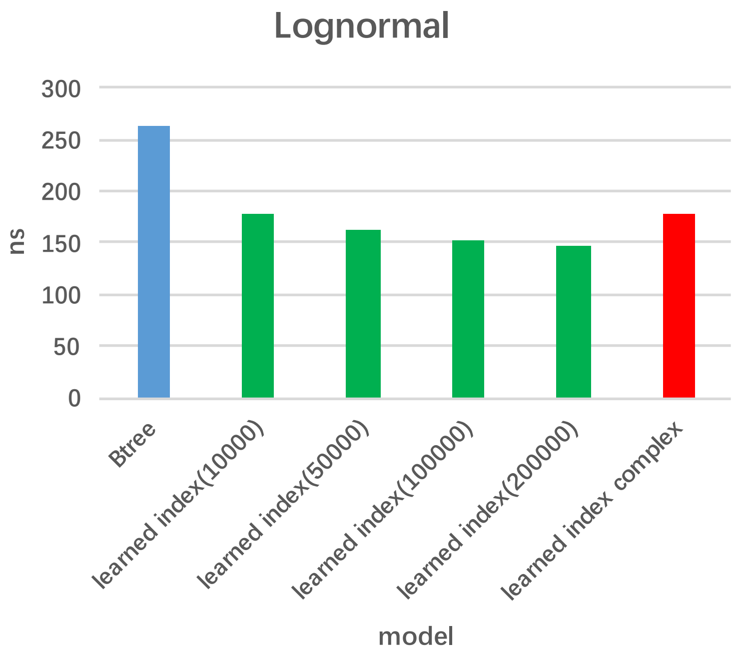
Range Index

- The evaluation of RMI (Speedup)

Linear → **Non-linear**



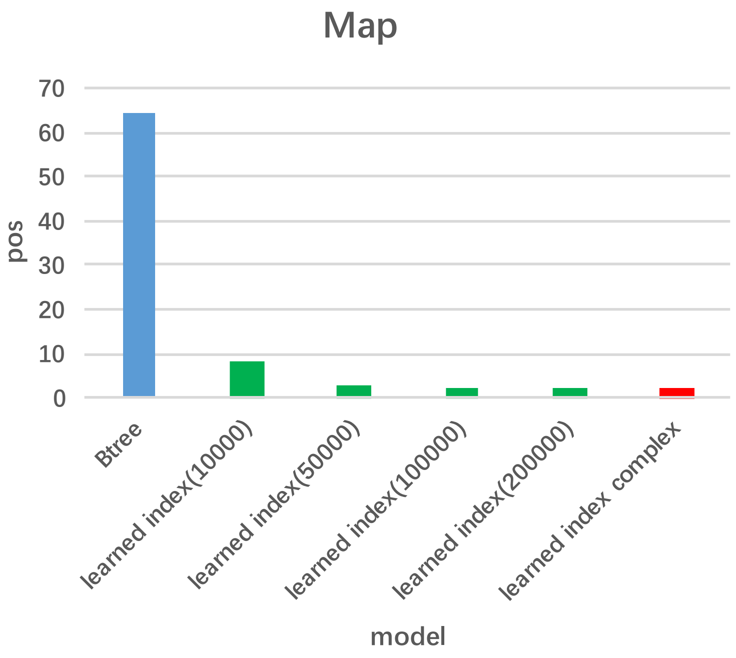


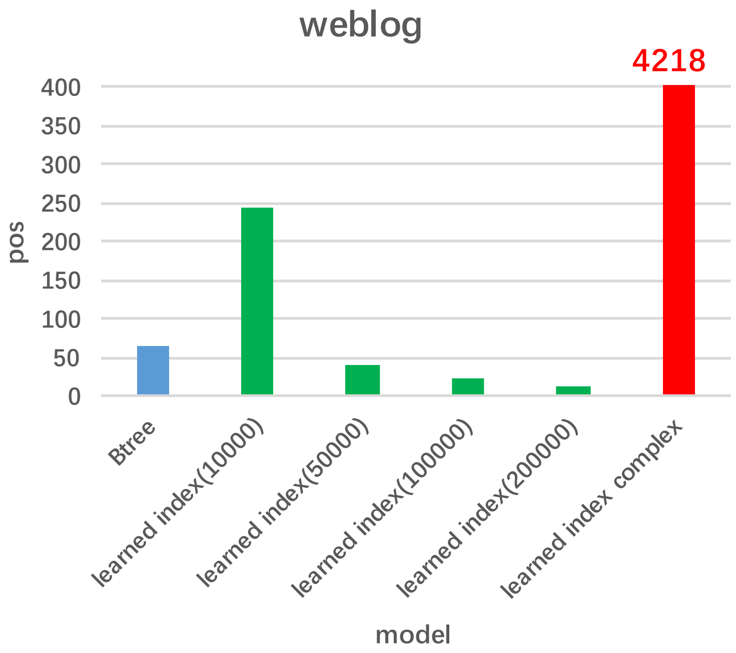


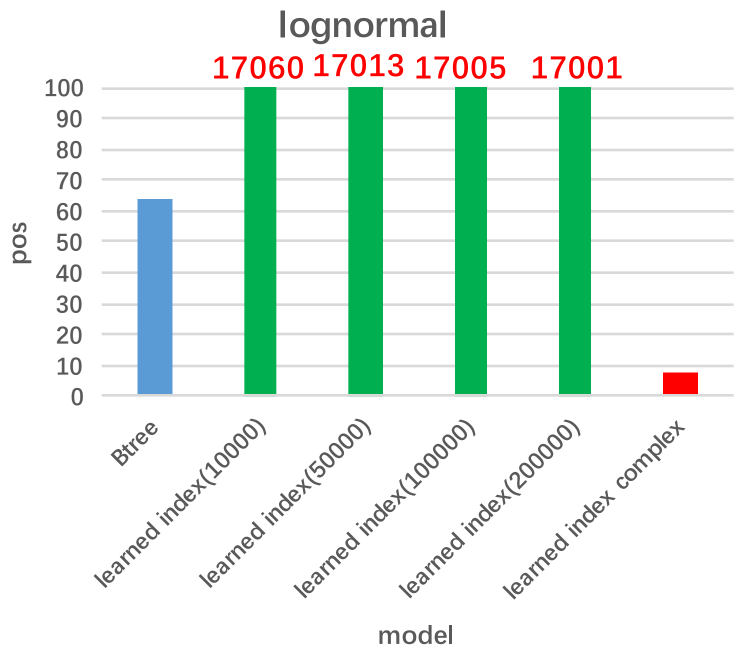
Range Index

- The evaluation of RMI (Model Error)

Linear → **Non-linear**

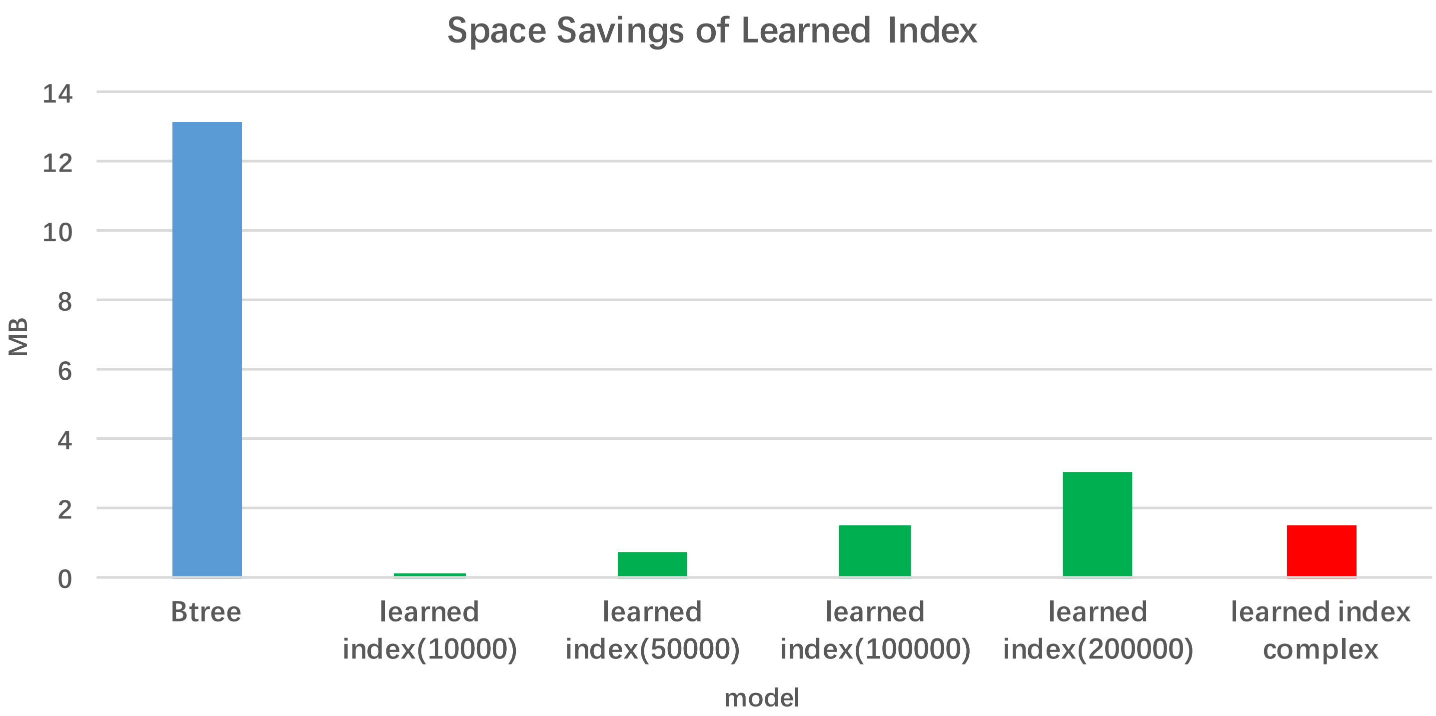






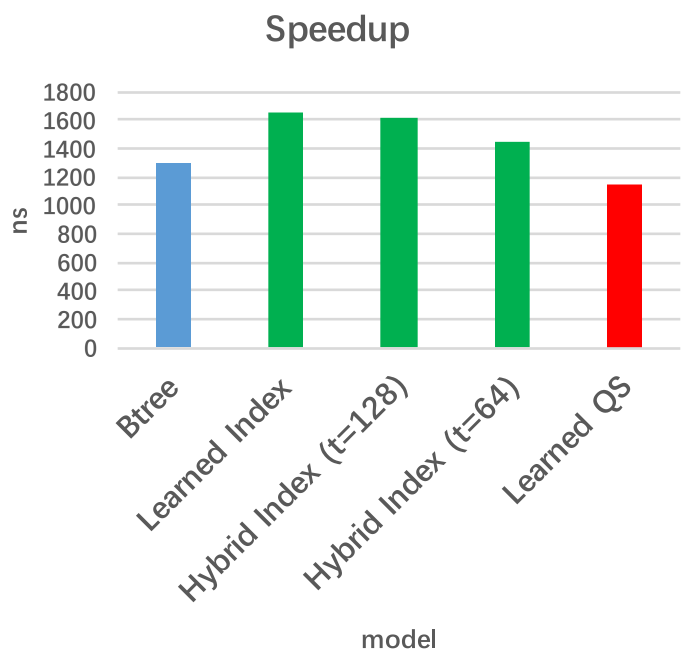
Range Index

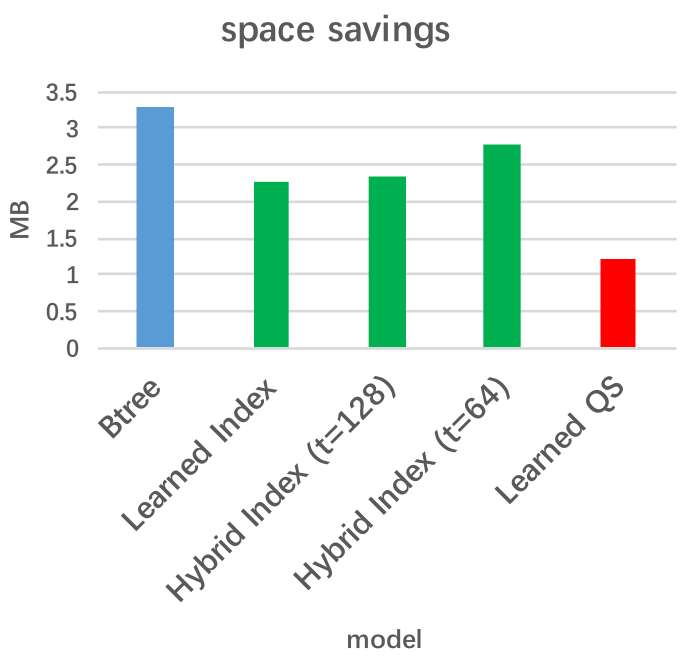
- The evaluation of RMI (Space savings)

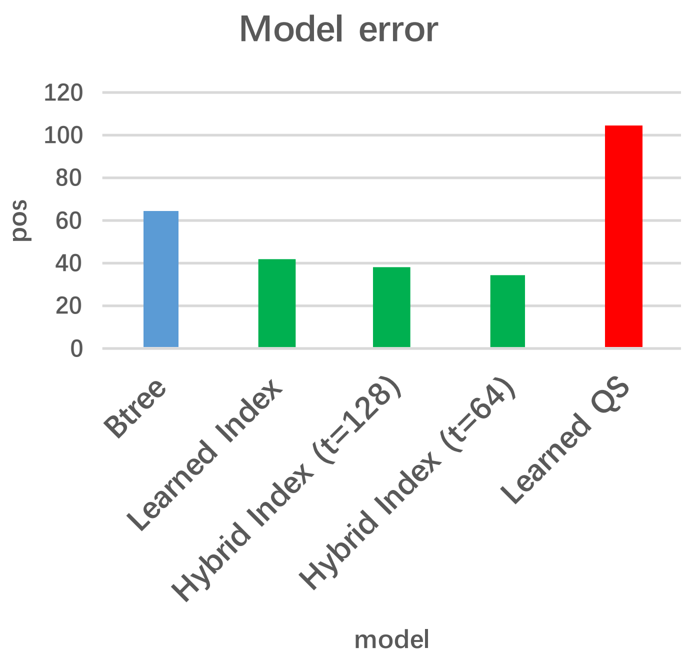


Range Index

- The evaluation of RMI on String dataset
  - String-based document id

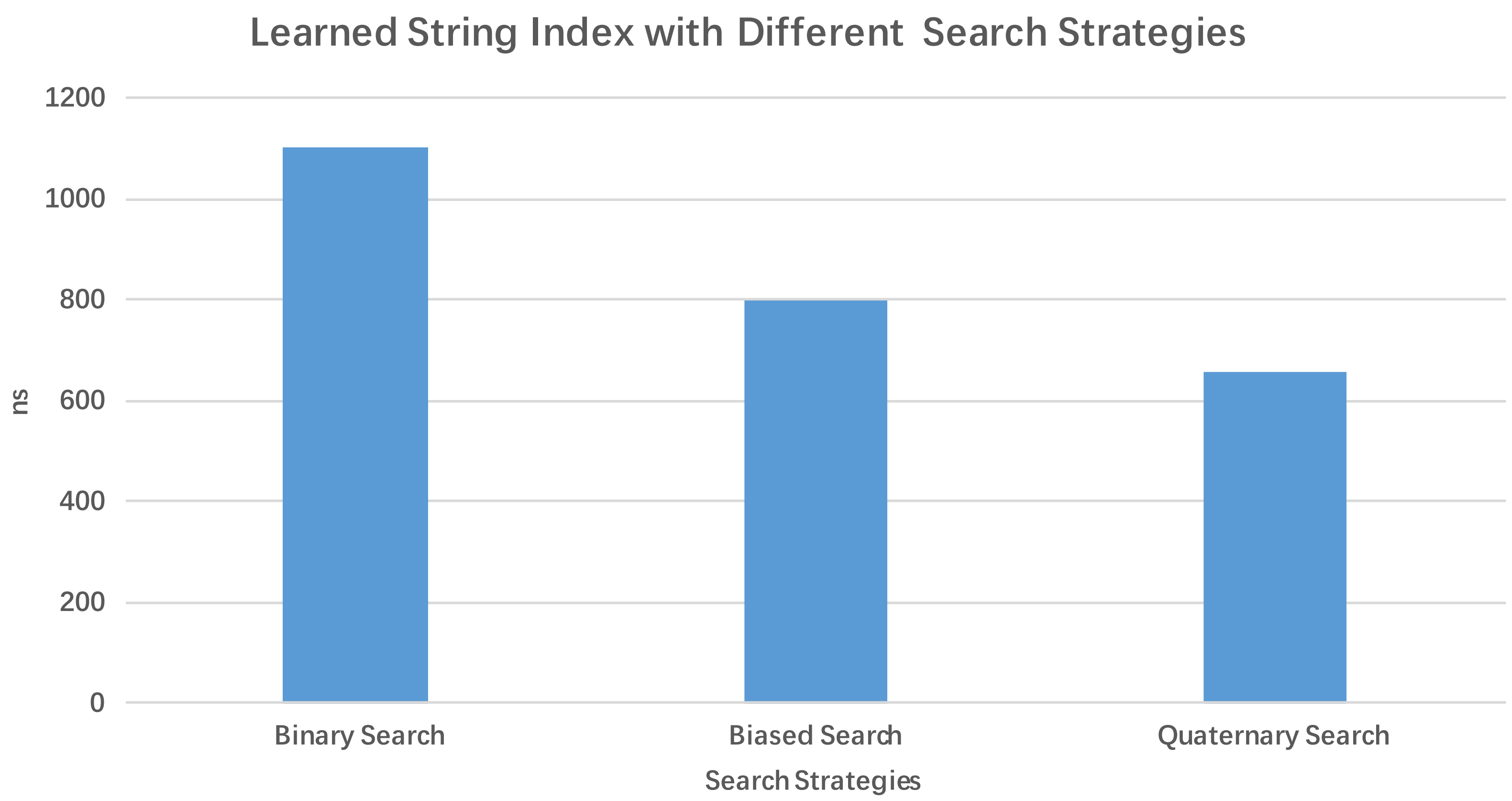






Range Index

- The evaluation of RMI on String dataset



Range Index

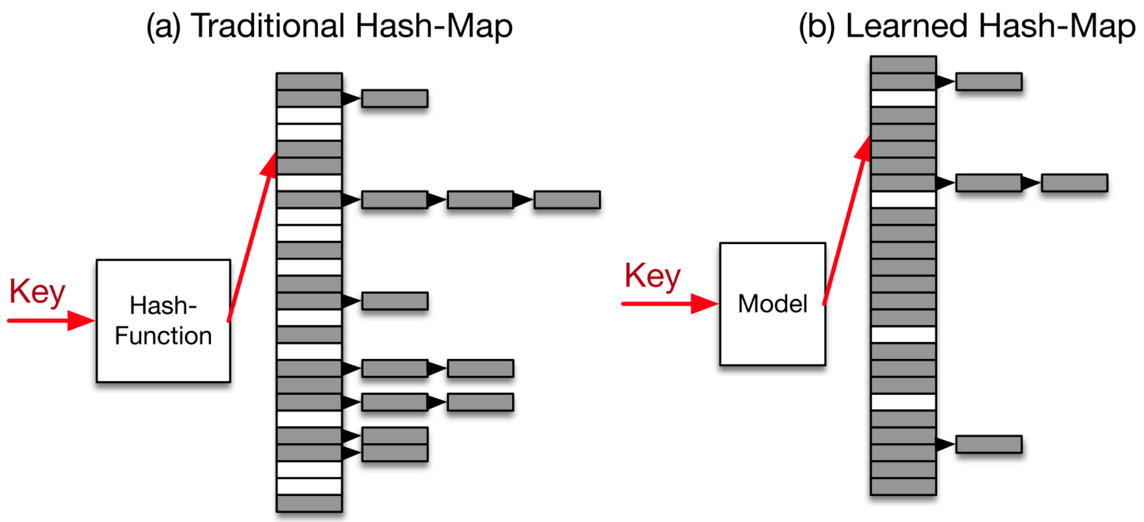
- Conclusion
  - 3x faster, an order-of-magnitude smaller
  - Data distribution dependent
    - Complex model has stronger expression ability, but is prone to over-fitting
  - Performance on String dataset still need to be improved

Outline

- Introduction
- Range Index
- Point Index
- Existence Index
- Related Work
- Conclusion and Future Work

Point Index

- Key challenge of Hash-Maps
  - Hash collisions (33%)
  - Linked list or secondary probing
- Learned Hash-Map
  - Learn better hash function
  - Learn the CDF of the key distribution
  - Scale to target size



$$h(K) = F(K) * M$$

Point Index

- Evaluation
  - Randomized hash function
  - 2 multiplications, 3 bitshifts, 3 XORs
  - 50% slot, no noticeable difference

Dataset	Slots	Hash Type	Search Time (ns)	Empty Slots	Space Improvement
Map	75%	Model Hash	67	0.63GB (05%)	-20%
		Random Hash	52	0.80GB (25%)	
	100%	Model Hash	53	1.10GB (08%)	-27%
		Random Hash	48	1.50GB (35%)	
	125%	Model Hash	64	2.16GB (26%)	-6%
		Random Hash	49	2.31GB (43%)	
Web Log	75%	Model Hash	78	0.18GB (19%)	-78%
		Random Hash	53	0.84GB (25%)	
	100%	Model Hash	63	0.35GB (25%)	-78%
		Random Hash	50	1.58GB (35%)	
	125%	Model Hash	77	1.47GB (40%)	-39%
		Random Hash	50	2.43GB (43%)	
Log Normal	75%	Model Hash	79	0.63GB (20%)	-22%
		Random Hash	52	0.80GB (25%)	
	100%	Model Hash	66	1.10GB (26%)	-30%
		Random Hash	46	1.50GB (35%)	
	125%	Model Hash	77	2.16GB (41%)	-9%
		Random Hash	46	2.31GB (44%)	

Outline

- Introduction
- Range Index
- Point Index
- Existence Index
- Related Work
- Conclusion and Future Work

Existence Index

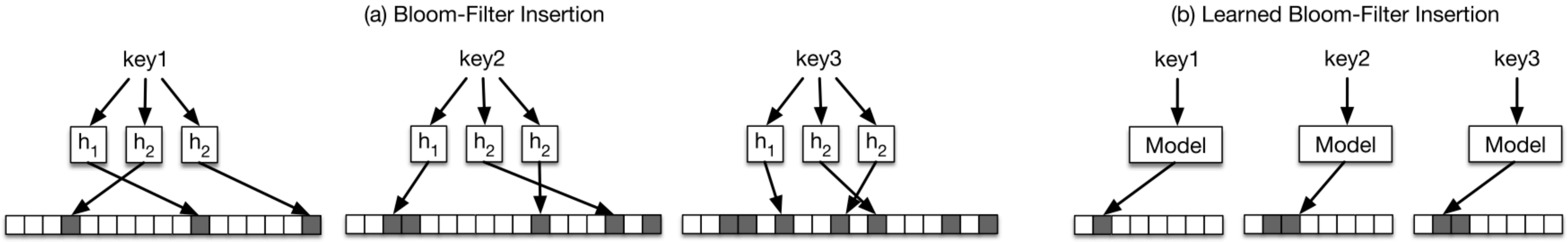
- Bloom-Filters
  - Bit array size *m*, *k* hash functions
  - Targeted FPR, FNR = 0
  - Occupy a significant amount of memory
    - 100 M records
    - FPR = 0.1% -> 1.76G
    - FPR = 0.01% -> 2.23G

Existence Index

- Learned Bloom Filters
  - No memory footprint if we know the exact data distribution

$$f(x) \equiv \mathbb{1}[0 \leq x < n]$$

  - Lots of collisions among keys, few collisions of keys and non-keys
  - Learn a function to separate keys from everything else
    - Non-keys (randomly generated, based on previous queries)



Existence Index

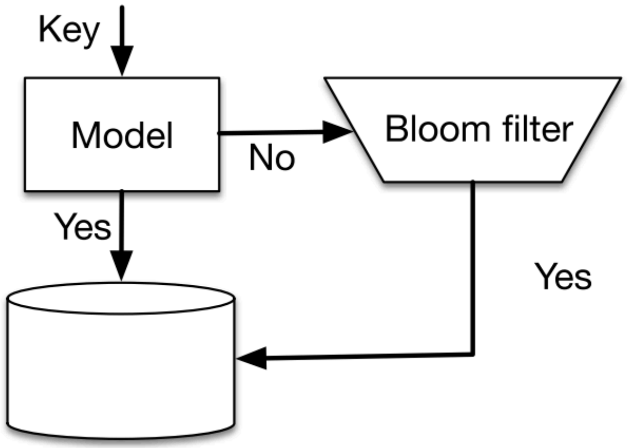
- Classification Problem

$$\mathcal{D} = \{(x_i, y_i = 1) | x_i \in \mathcal{K}\} \cup \{(x_i, y_i = 0) | x_i \in \mathcal{U}\}$$

$$L = \sum_{(x,y)\in\mathcal{D}} y \log f(x) + (1-y)\log(1-f(x)).$$

  - Overflow Bloom-Filter to maintain FNR = 0

$$\mathcal{K}_\tau^- = \{x \in \mathcal{K} | f(x) < \tau\}$$

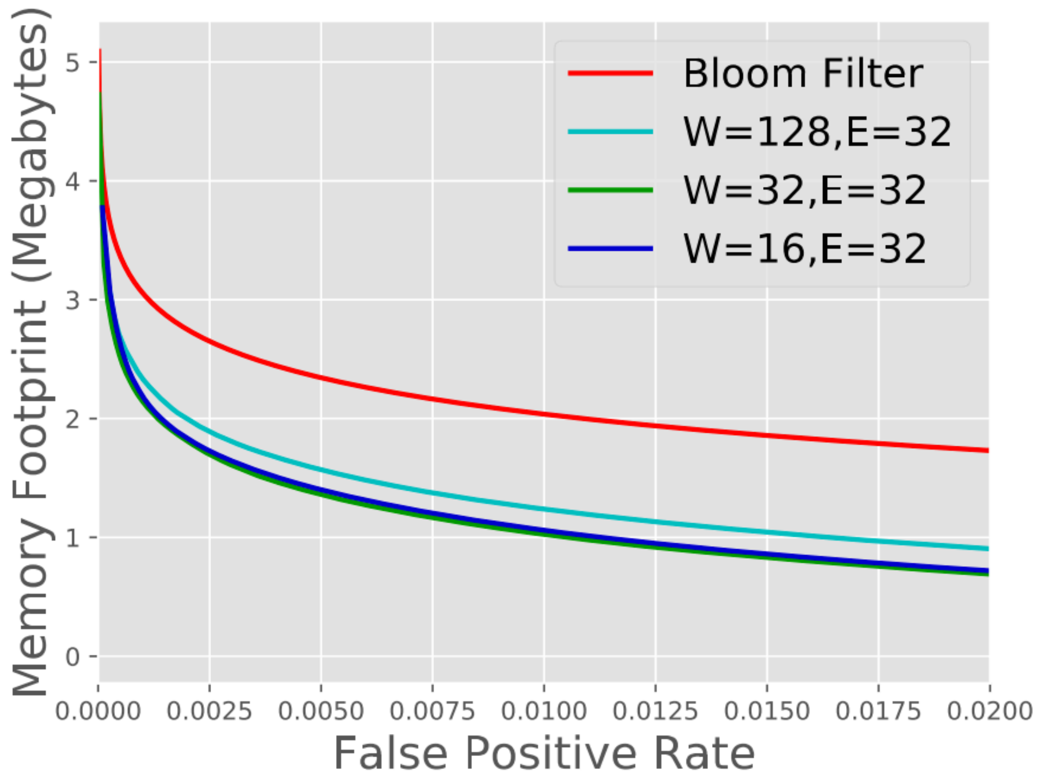


- Map keys to higher bits, non-keys to lower bits

$$d(p) = \lfloor mp \rfloor$$

Existence Index

- Evaluation
  - Task: keeping track of blacklisted phishing URLs
  - Model: RNN
  - More accurate -> better savings



Outline

- Introduction
- Range Index
- Point Index
- Existence Index
- Related Work
- Conclusion and Future Work

Related Work

- B-Tree Optimization
  - A-Trees (piecewise linear functions to reduce leaf nodes)
  - BF-Trees (B+ tree + bloom filter)

- Better Hash Functions
  - Feature hashing

- Bloom-Filters
  - 《Adaptive range filters for cold data》
  - 《Practically better than bloom》

Related Work

- Succinct Data Structures
  - Wavelet trees
- Modeling CDFs
  - PDF vs CDF
- Mixture of Experts
  - Building experts of subset of the data

Outline

- Introduction
- Range Index
- Point Index
- Existence Index
- Related Work
- Conclusion and Future Work

Conclusion and Future Work

- Multi-Dimensional Index
- Learned Algorithm
- New generation of hardware