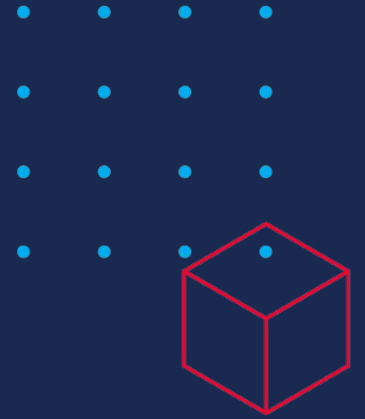
PingCAP



A Simple Introduction to Learning-based Cardinality Estimation

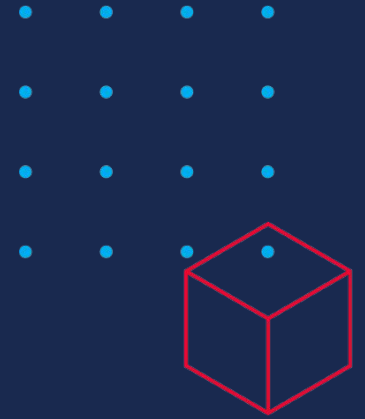
Yuanjia Zhang

Overview

PingCAP

1. **A Simple Introduction to CE and NN**
    a. A simple introduction to cardinality estimation(CE) methods
    b. A simpe introduction to neural networks(NN)
2. **Paper Sharing**
    a. Cardinality Estimation Using Neural Networks [2015]
    b. Learned cardinalities: Estimating correlated joins with deep learning [2019]
    c. Selectivity estimation for range predicates using lightweight models [2019]

PingCAP



A Simple Introduction to CE methods and NN

A Simple Introduction to CE methods

Synopsis-based methods **use some data structures to record the statistics(distribution) information of the data**.

- stable, efficient
- hard to capture correlation

Sampling-based methods **run the (sub)query over samples of the data to estimate cardinalities**.

- good at capturing correlation
- inefficient, more resource consuming → expensive & unstable

Supervised Learning-based methods **learn the statistics(distribution) information of the data from queries and their results**.

- accurate
- unexplainable, hard to handle data updates

Unsupervied Learning-based methods: TODO…

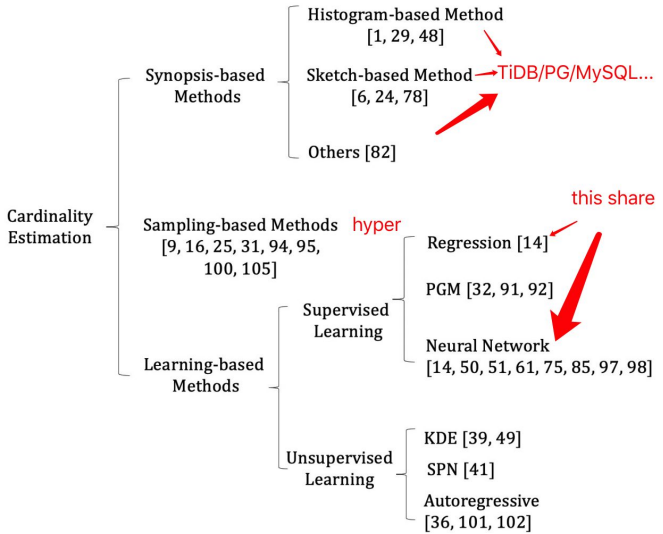


Figure 2: A classification of cardinality estimation methods

Ref: A Survey on Advancing the DBMS Query Optimizer: Cardinality Estimation, Cost Model, and Plan Enumeration

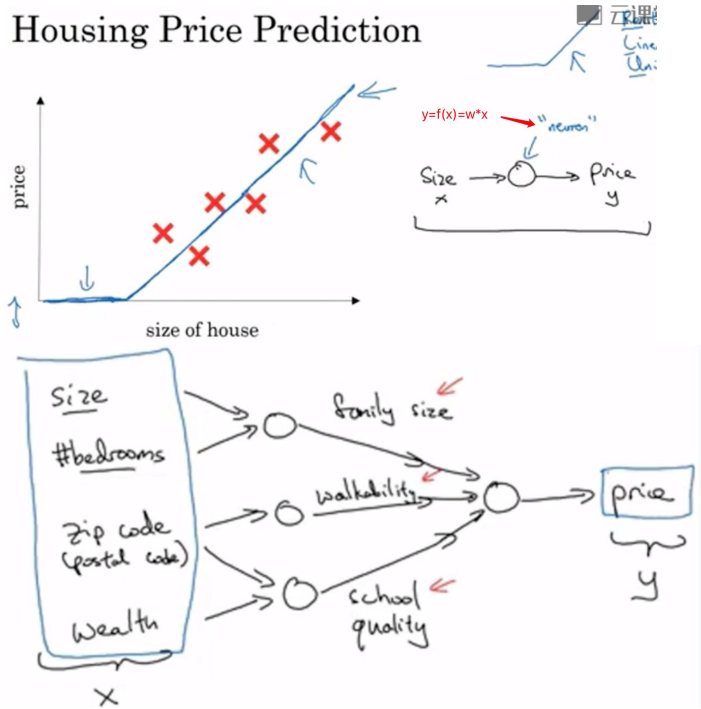
A simple introduction to NN

Example: Housing Price Prediction

- one feature: size -> price
- multiple features -> price
- the loss function
    - e.g. SUM[1~m](|y-y'|) / m
- training data and test data
- the training algorithm can get the best `w` automatically according the the loss function and training data

Our focuses:

1. **Featurization**: How to present queries?
2. **NN Architecture**: Which learning algo should be used?
3. **Cold Start Problem**: How to obtain the initial training data?
4. **Data Changing Problem**: How to handle data changes/updates?



A simple introduction to NN + CE

Example: Housing Price Prediction

- one feature: size -> price
- multiple features -> price
- the loss function
  - e.g. SUM[1~m](|y-y'|) / m
- training data and test data
- the training algorithm can get the best `w` autom
  the the loss function and training data

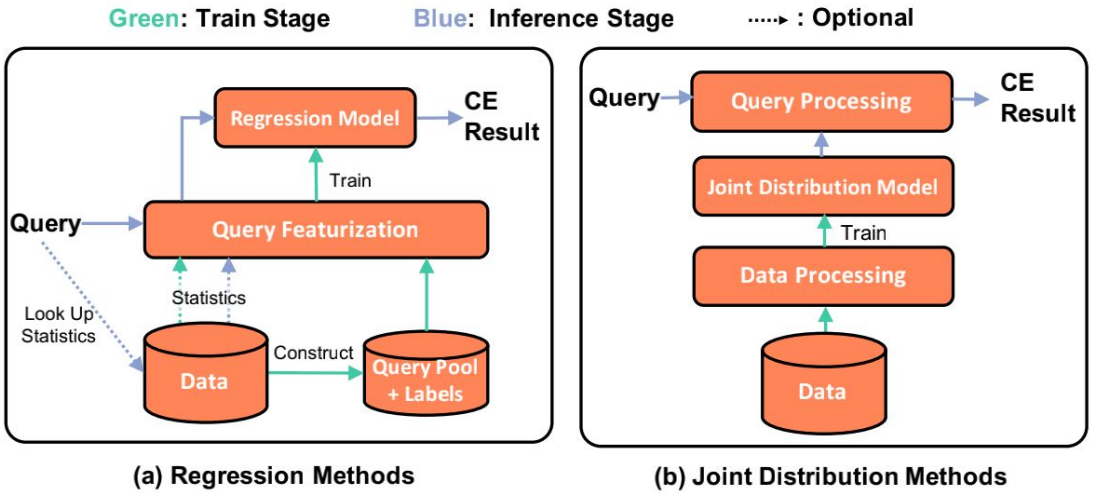
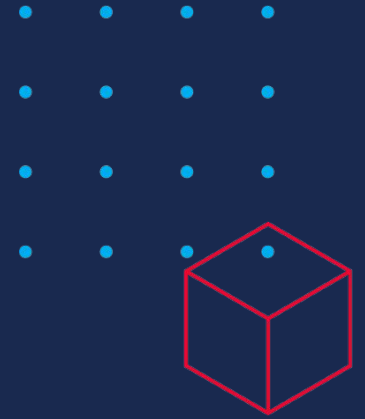


Figure 1: Workflow of Learned Methods.

Our focuses:

1. **Featurization**: How to present queries?
2. **NN Architecture**: Which learning algo should be used?
3. **Cold Start Problem**: How to obtain the initial training data?
4. **Data Changing Problem**: How to handle data changes/updates?

PingCAP



Paper1: Cardinality Estimation Using Neural Networks [2015]

Paper1: Cardinality Estimation Using Neural Networks

PingCAP

Featurization: How to present queries?

Only consider single-table queries with range predicates[2015]:

```
select * from t where t.c1 > l1 and t.c1 < u1 and
                      t.c2 > l2 and t.c2 < u2 and
                      t.c3 > l3 and t.c3 < u3 ...
```

Let N be the number of columns in this table `t`. We may view the selectivity of queries of this sort as a function of the values l_i and u_i. Formally, we have the selectivity function

$$\text{Sel} : \mathbb{R}^{2N} \to \mathbb{R}, \quad (l_1, u_1, \ldots, l_N, u_N) \mapsto c \tag{6}$$

where c is the selectivity of the query represented by the values l_i and u_i.

Paper1: Cardinality Estimation Using Neural Networks

PingCAP

NN Architecture: Which learning algorithm should be used?

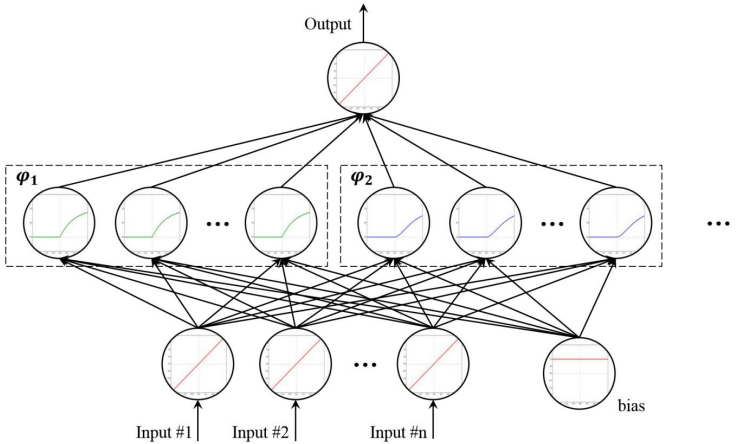
The Loss Function: minimize the difference

sum[1~N](|est_sel - act_sel|)

```
select * from t where t.c1 > l1 and t.c1 < u1 and
                      t.c2 > l2 and t.c2 < u2 and
                      t.c3 > l3 and t.c3 < u3 ...
```

featurization

$$\text{Sel} : \mathbb{R}^{2N} \to \mathbb{R}, \quad (l_1, u_1, \ldots, l_N, u_N) \mapsto c$$



training: minimize the loss

Paper1: Cardinality Estimation Using Neural Networks

PingCAP

The Cold Start Problem: How to obtain the initial training data?

The training data should be: broad and varied <-> specific and precise;

Use a simple query generation algorithm to generate the training data:

Algorithm 1 Automatic Training Query Generator

```
1: procedure GenQuery(numCol)
2:     nShrink ← randInt(1, numCol)
3:     C ← a subset of {1, ..., numCol} that contains nShrink elements
4:     for all C_i ∈ C do
5:         GenBounds(C_i)    ▷ assign predicate bounds to each column
6:     end for
7: end procedure
```

Paper1: Cardinality Estimation Using Neural Networks

PingCAP

Data Changing Problem: How to handle data changes?

This problem is not considered in this paper [2015].

Paper1: Cardinality Estimation Using Neural Networks

PingCAP

Evaluation:

1. dataset1: Strongly Correlated Columns

$$x_{i,1} = \texttt{randInt}(0, 100)$$

$$x_{i,n} = (x_{i,n-1} + \texttt{randInt}(2, 3)) \mod 100$$

2. dataset2: Car Accidents
   a. four columns: seatbelton, with, driver, damage;
   b. the usage of seatbelt decreases the injurity level of the driver;
   c. the object that the car hits affects the damage level;

Paper1: Cardinality Estimation Using Neural Networks

PingCAP

Evaluation:

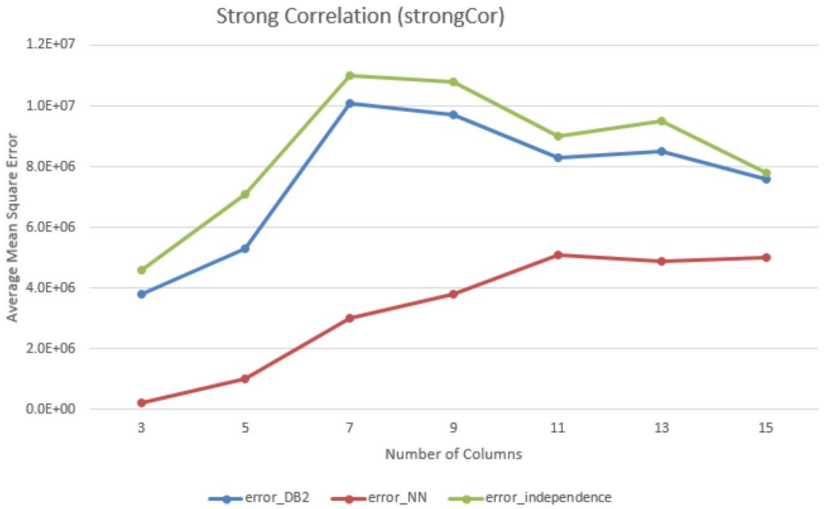
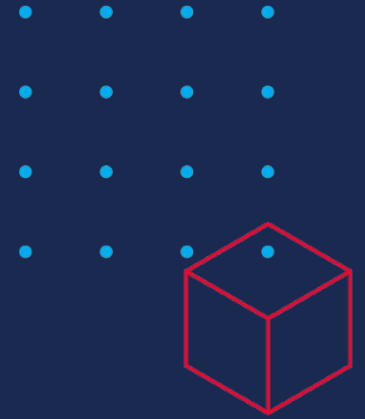


Figure 5: Error of strongCor

with	driver	damage	NN	DB2	ACTUAL
ANIMAL CAR	DEAD INJURED	MINOR SEVERE	11385	2946	9761
ANIMAL CAR	INJURED UNHARMED	MINOR SEVERE	13295	2946	11498
CAR PEOPLE	DEAD INJURED	MINOR SEVERE	13207	2946	10634
CAR PEOPLE	INJURED UNHARMED	MINOR SEVERE	21548	2946	18703
ANIMAL CAR	DEAD INJURED	NONE SEVERE	7902	1726	7710
ANIMAL CAR	INJURED UNHARMED	NONE SEVERE	8485	1726	7268
CAR PEOPLE	DEAD INJURED	NONE SEVERE	8605	1726	8195
CAR PEOPLE	INJURED UNHARMED	NONE SEVERE	15792	1726	11310
ANIMAL CAR	DEAD INJURED	MINOR SEVERE	6190	2727	11454
ANIMAL CAR	DEAD INJURED	MINOR SEVERE	7338	2727	12422
CAR PEOPLE	DEAD INJURED	MINOR SEVERE	7082	2727	11963
CAR PEOPLE	DEAD INJURED	MINOR SEVERE	12326	2727	16350
ANIMAL CAR	DEAD INJURED	MINOR SEVERE	6748	3328	11676
ANIMAL CAR	DEAD INJURED	MINOR SEVERE	7979	3328	12950
CAR PEOPLE	DEAD INJURED	MINOR SEVERE	8000	3328	12720
CAR PEOPLE	DEAD INJURED	MINOR SEVERE	13793	3328	21280
ANIMAL CAR	DEAD INJURED	MINOR SEVERE	6384	1821	2273
ANIMAL CAR	DEAD INJURED	MINOR SEVERE	7581	1821	4758
CAR PEOPLE	DEAD INJURED	MINOR SEVERE	7607	1821	3196
CAR PEOPLE	DEAD INJURED	MINOR SEVERE	13338	1821	12323

Table 5: Result of Queries with Range Predicates

PingCAP



Paper2: Selectivity estimation for range predicates using lightweight models[2019]

Paper2: Selectivity estimation for range predicates using lightweight models

PingCAP

Featurization: How to present queries?

- only consider single-table queries with range predicates;
- the table T with d attributes: A1, A2, …Ad;
- the domain of the k-th attribute is [mink, maxk];
- a CNF query can be represented: (lb1<=A1<=ub1) AND (...A2…) … AND (...Ad…);
- if some attribute is not contained: mink<=Ak<=maxk;

Then the training dataset S can be represented as:

$$S = \{(q_1 : act(q_1)), \ldots, (q_m : act(q_m))\}$$

e.g.

$$S_{example} = \{((10 \leq A_1 \leq 20) \wedge (0 \leq A_2 \leq 100) : 500), ((10 \leq A_1 \leq 20) \wedge (40 \leq A_2 \leq 80) : 300), \ldots\}.$$

Paper2: Selectivity estimation for range predicates using lightweight models

PingCAP

Featurization: Extra Features!

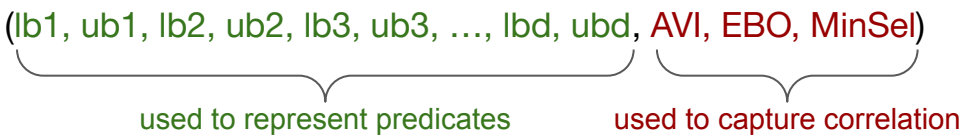
Use results of heuristic estimators as features to **capture the degree of correlation**!

The three heuristic estimators:

1. Attribute Value Independence(AVI): sel(A1) * sel(A2) … * sel(Ad)
2. Exponential BackOff(EBO): sel(Ak1) * sel(Ak2)/2 * sel(Ak3)/4 * sel(Ak4)/8 ← ordered by sel(Ak) desc
3. MinSel: min(sel(A1), sel(A2), … sel(Ad))

The extra feature can be prepresented as: (AVI, EBO, MinSel)

Then the whole feature vector is:



Paper2: Selectivity estimation for range predicates using lightweight models

PingCAP

Which learning algorithm should be used?

The Loss Function: log-transform + MSE(low mean-squared error)

$$\frac{1}{|S_{test}|}\sum_{i=1}^{|S_{test}|}[\log act(q_i) - \log est(q_i)]^2 = \frac{1}{|S_{test}|}\sum_{i=1}^{|S_{test}|}\log^2 e_i$$

If no log-transform, for two queries q1 and q2:

- if act(q1) = 20000 and act(q2) = 100;
- result1: est(q1)=19500 + est(q2)=600 → |20000-19500|^2+|600-100|^2 = 500000
- result2: est(q1)=19100 + est(q2)=200 → |20000-19100|^2+|200-100|^2 = 820000
- then the model would think result1 is bettern than result2;

Use the log-transform to unify the difference between varied queries.

$$|\log act(q_i) - \log est(q_i)| = \left|\log \frac{act(q_i)}{est(q_i)}\right|$$

→ q-error?

Paper2: Selectivity estimation for range predicates using lightweight models

Which learning algorithm should be used?

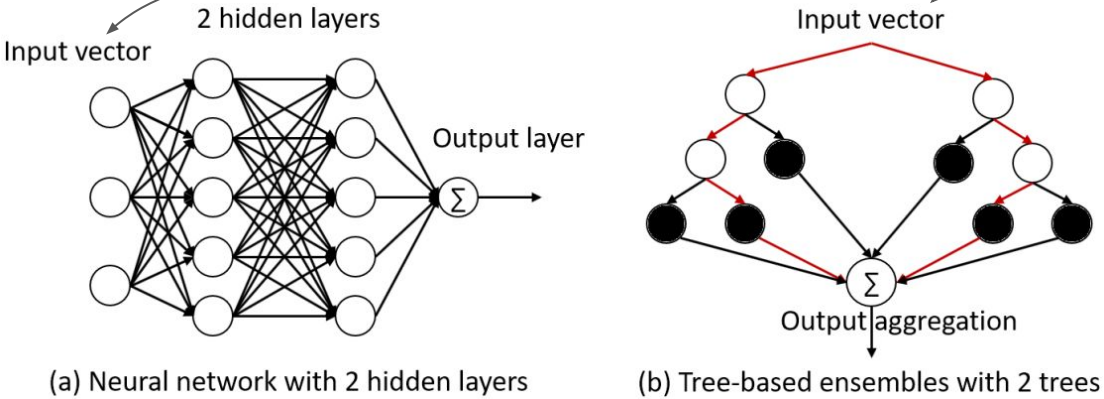
1. NN: 2 hidden layers + ReLU;
2. Tree-based ensembles: random forest + GBDT
3. The Loss Function: log-transform + MSE

$$\frac{1}{|S_{test}|}\sum_{i=1}^{|S_{test}|}[\log act(q_i) - \log est(q_i)]^2 = \frac{1}{|S_{test}|}\sum_{i=1}^{|S_{test}|}\log^2 e_i$$

(lb1, ub1, lb2, ub2, lb3, ub3, …, lbd, ubd, AVI, EBO, MinSel)

used to represent predicates

used to capture correlation



(a) Neural network with 2 hidden layers

(b) Tree-based ensembles with 2 trees

Paper2: Selectivity estimation for range predicates using lightweight models

PingCAP

The Cold Start Problem: How to obtain the initial training data?

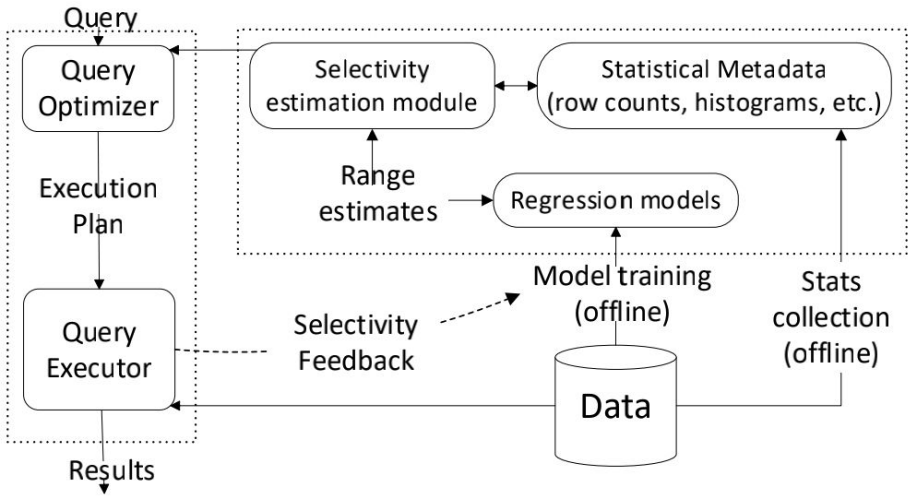
Use a simple query generation algorithm to generate training data:

- for a table with D attributes, generate queries with 2~D attributes randomly;
- for a query with d attributes, choose a subset of attributes randomly (Ak1, … Akd);
- for a given attribute in a query, generate the range by **uniform or exponential** distribution;

Paper2: Selectivity estimation for range predicates using lightweight models

Data Changing Problem: How to handle data changes?

- The retraining can be triggered by a bulk of data update or a large fraction of bad estimations.
- The actual cardinalities used for training can be obtained from the executor feedbacks.
- Too ideal?



Paper2: Selectivity estimation for range predicates using lightweight models

PingCAP

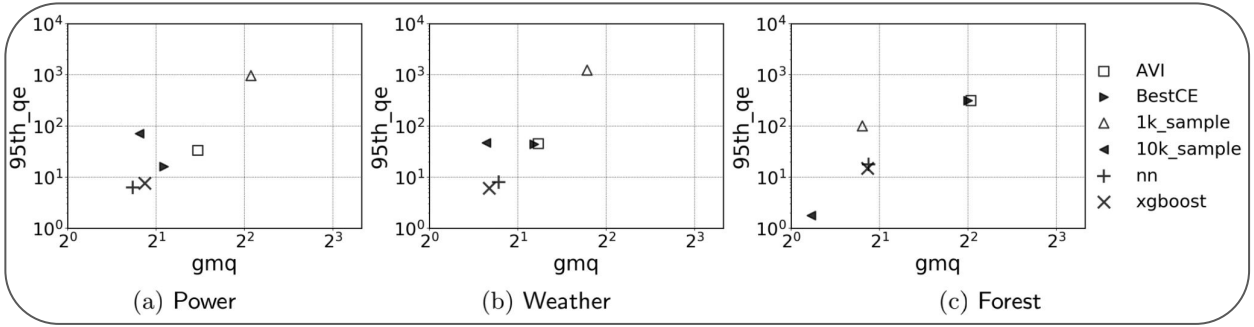
Evaluation:

Use 4 real-world datasets from different domains:

1. Forest: The original forest cover type dataset has 581012 rows and 54 attributes;
2. Power: It contains measurements of electric power consumption in a household at one-minute sampling rate over 4 years;
3. Weather: This is a dataset constructed by authors of using data sourced from daily global historical climate network;
4. Higgs: This is a scientifific dataset that has 11 million rows with features regarding kinematic properties measured by particle detectors in the accelerator;

Paper2: Selectivity estimation for range predicates using lightweight models

Evaluation: BestCE → Best(AVI, EBO, MinSel); gmq → geometric-mean-q-error;



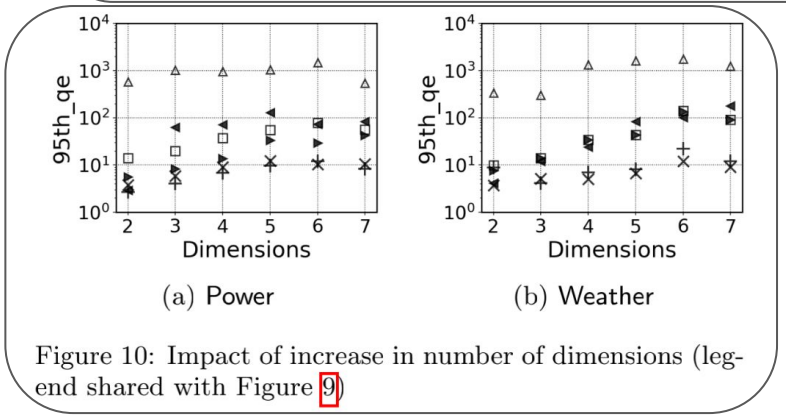


Figure 10: Impact of increase in number of dimensions (legend shared with Figure 9)

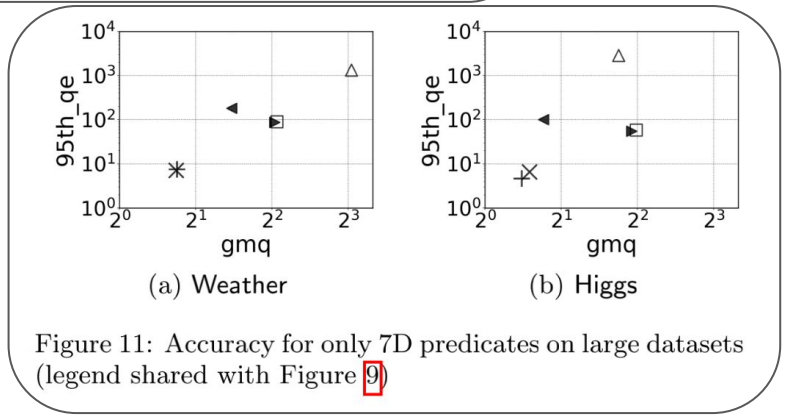
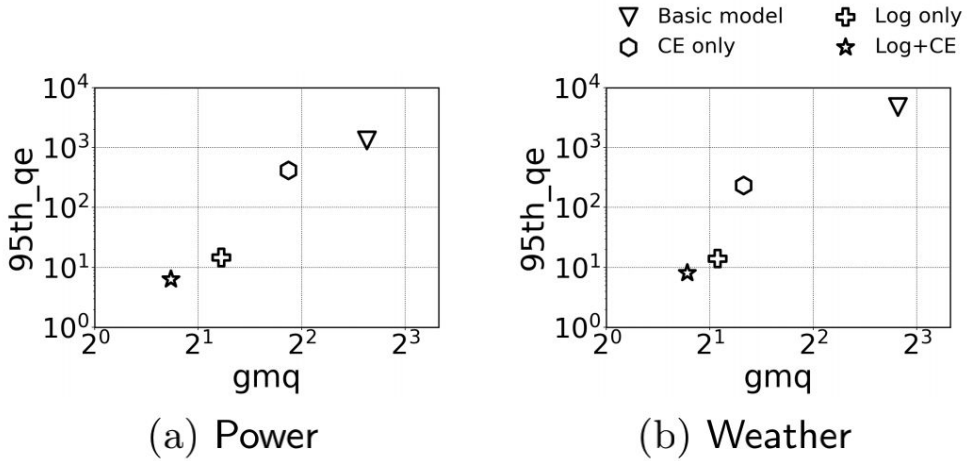


Figure 11: Accuracy for only 7D predicates on large datasets (legend shared with Figure 9)

Paper2: Selectivity estimation for range predicates using lightweight models

Evaluation: impact of log-tran and extra features

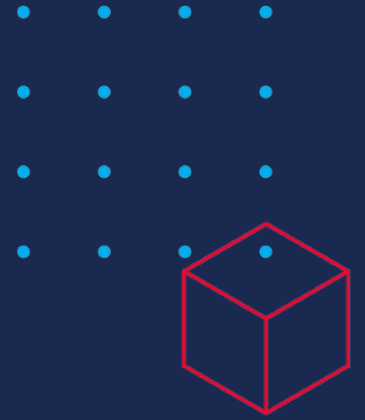
- Basic Model: no log-transform and no extra features
- Log Only: no extra freatures
- CE Only: no log-transform
- Log+CE: has them both



(a) Power (b) Weather

Figure 12: Design impact on NN models

PingCAP



Paper3: Learned cardinalities: Estimating correlated joins with deep learning [2019]

Paper3: Learned cardinalities: Estimating correlated joins with deep learning

PingCAP

Featurization: How to present queries?

- consider join queries with range/eq predicates
- present a query as a unique vector with 3 parts
  - table-set: indicate which tables(and their data) are contained by this query;
  - join-set: enumerate all join cases and encoding them;
  - pred-set: contain all information of predicates;

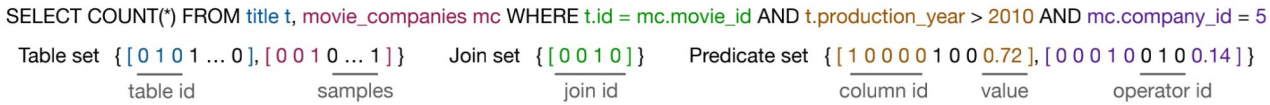
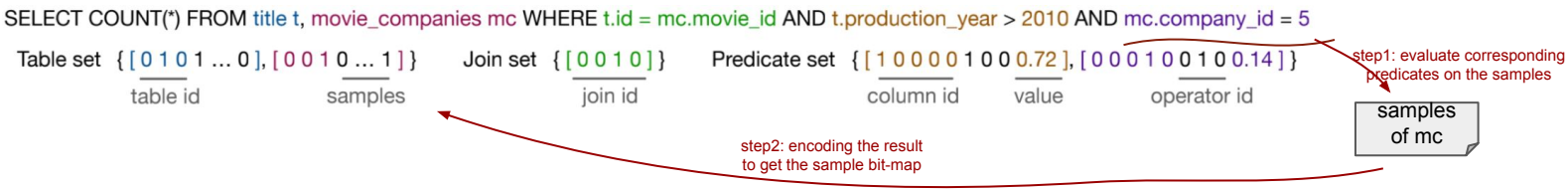


Figure 2: Query featurization as sets of feature vectors.

- samples: evaluate the corresponding predicates of a table on its samples to get the sample-bit-map; it can reflect the data distribution(position) of the table used to join.



Paper3: Learned cardinalities: Estimating correlated joins with deep learning

NN Architecture: Which learning algorithm should be used?

Two-layer NN:

- the first layer is used to learn each feature separatly
  - apply a MLP on every element of the set
  - use an avg to pool the results of the same set
  - MLPs for the same set share same parameters
  - concat results from all MLPs(set) togather
- the second layer consider all features and generate the final result

The Loss Function: minimize the mean q-error

- sum[1~N](q-error) / N

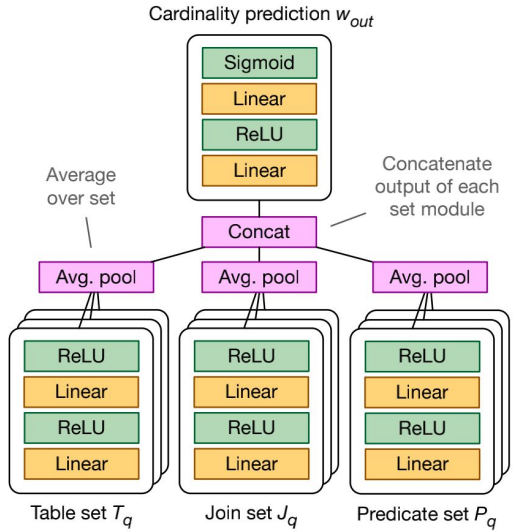


Figure 1: Architecture of our multi-set convolutional network. Tables, joins, and predicates are represented as separate modules, comprised of one two-layer neural network per set element with shared parameters. Module outputs are averaged, concatenated, and fed into a final output network.

Paper3: Learned cardinalities: Estimating correlated joins with deep learning

PingCAP

The Cold Start Problem: How to obtain the initial training data?

- generate queries with up to 2 joins and let the model generalize to more joins;
- select tables to join **uniformly**;
- genearte operators =, >, < **uniformly**;

Paper3: Learned cardinalities: Estimating correlated joins with deep learning

PingCAP

Data Changing Problem: How to handle data changes?

There is some discussion about re-training and incremental-training, but it didn't describe how to handle this issue formally.

Paper3: Learned cardinalities: Estimating correlated joins with deep learning

PingCAP

Evaluation:

- Dataset: IMDB;
- Workload:
  - a synthetic workload generated by the same method as the trainning data;
  - another synthetic workload with more joins;
  - JOB-light with 113 queries without any predicates on strings;

number of joins	0	1	2	3	4	overall
synthetic	1636	1407	1957	0	0	5000
scale	100	100	100	100	100	500
JOB-light	0	3	32	23	12	70

Table 1: Distribution of joins.

Paper3: Learned cardinalities: Estimating correlated joins with deep learning

PingCAP

Evaluation:

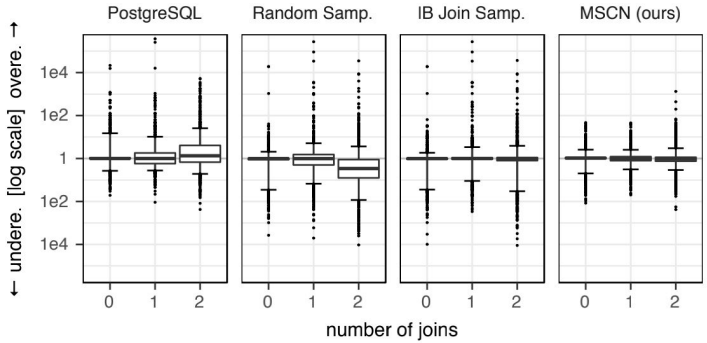


Figure 3: Estimation errors on the synthetic workload. The box boundaries are at the 25th/75th percentiles and the horizontal "whisker" lines mark the 95th percentiles.

	median	90th	95th	99th	max	mean
PostgreSQL	1.69	9.57	23.9	465	373901	154
Random Samp.	1.89	19.2	53.4	587	272501	125
IB Join Samp.	**1.09**	9.93	33.2	295	272514	118
MSCN (ours)	1.18	**3.32**	**6.84**	**30.51**	**1322**	**2.89**

Table 2: Estimation errors on the synthetic workload.

	median	90th	95th	99th	max	mean
PostgreSQL	4.78	62.8	107	1141	21522	133
Random Samp.	9.13	80.1	173	993	19009	147
MSCN	**2.94**	**13.6**	**28.4**	**56.9**	**119**	**6.89**

Table 3: Estimation errors of 376 base table queries with empty samples in the synthetic workload.

Paper3: Learned cardinalities: Estimating correlated joins with deep learning

PingCAP

Evaluation: impact of sample bitmaps

```
SELECT COUNT(*) FROM title t, movie_companies mc
WHERE t.id = mc.movie_id AND t.production_year > 2010 AND mc.company_id = 5
```

- no samples: without any sampling features

Table set { [0 1 0 ...], [0 0 1 ...] }
table id samples

- #samples: with one cardinality

Table set { [0 1 0 ...], [0 0 1 ...] }
table id 155 samples 32

- bitmaps: with one bitmap

Table set { [0 1 0 1 ... 0], [0 0 1 0 ... 1] }
table id samples

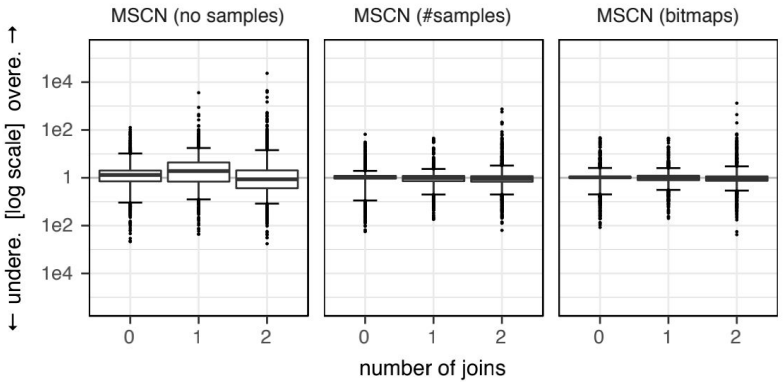
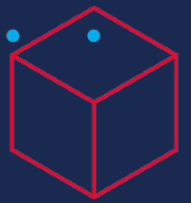


Figure 4: Estimation errors on the synthetic workload with different model variants.

PingCAP



Summary

Summary

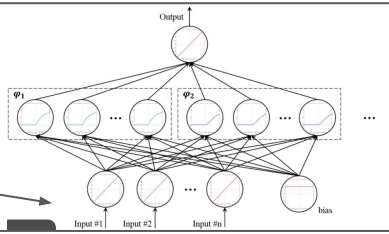
paper1: Cardinality Estimation Using Neural Networks [2015]

```
select * from t where t.c1 > l1 and t.c1 < u1 and
                      t.c2 > l2 and t.c2 < u2 and
                      t.c3 > l3 and t.c3 < u3 ...
```

$(l_1, u_1, \ldots, l_N, u_N)$

The Loss Function:

sum[1~N](|est_sel - act_sel|)



paper2: Selectivity estimation for range predicates using lightweight models Cardinality[2019]

```
select * from t where t.c1 > l1 and t.c1 < u1 and
                      t.c2 > l2 and t.c2 < u2 and
                      t.c3 > l3 and t.c3 < u3 ...
```

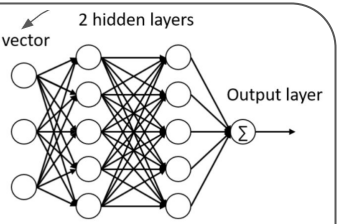
(lb1, ub1, lb2, ub2, lb3, ub3, …, lbd, ubd, AVI, EBO, MinSel)

used to represent predicates

used to capture correlation

The Loss Function: log-transform + MSE(low mean-squared error)

$$\frac{1}{|S_{test}|}\sum_{i=1}^{|S_{test}|}[\log act(q_i) - \log est(q_i)]^2 = \frac{1}{|S_{test}|}\sum_{i=1}^{|S_{test}|}\log^2 e_i$$



(a) Neural network with 2 hidden layers

paper3: Learned cardinalities: Estimating correlated joins with deep learning [2019]

join queries

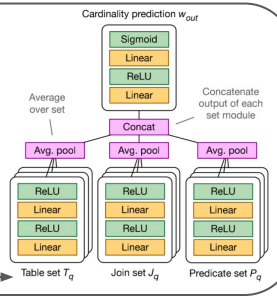
SELECT COUNT(*) FROM title t, movie_companies mc WHERE t.id = mc.movie_id AND t.production_year > 2010 AND mc.company_id = 5

Table set { [0 1 0 1 … 0], [0 0 1 0 … 1] } Join set { [0 0 1 0] } Predicate set { [1 0 0 0 0 1 0 0 0.72], [0 0 0 1 0 0 1 0 1 0 0.14] }

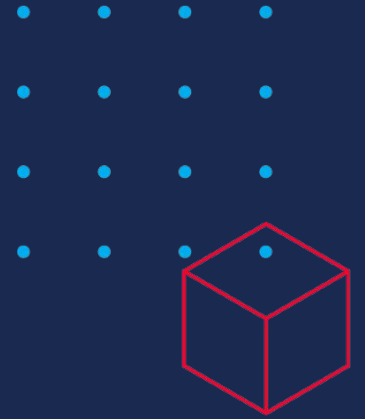
table id samples join id column id value operator id

The Loss Function: minimize the mean q-error

- sum[1~N](q-error) / N



PingCAP



Thank You
&
Discussion