AI Meets Query Optimization

Luming Sun @ DW&BI

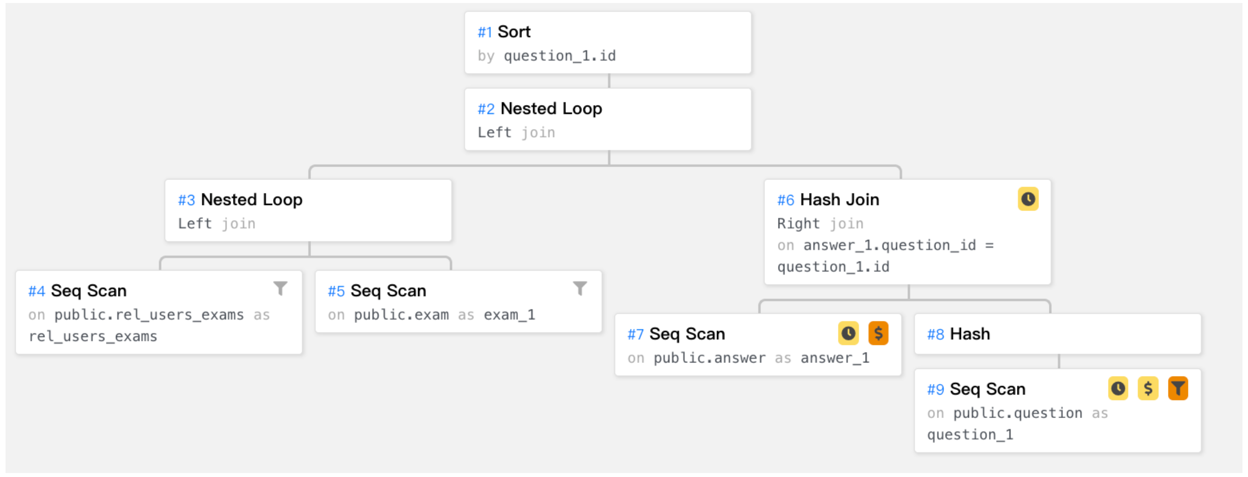
2022/9/19

Query Optimization

Query optimizer is at the heart of the database systems

Query Optimization

```sql
SELECT rel_users_exams.user_username AS rel_users_exams_user_username,
         rel_users_exams.exam_id AS rel_users_exams_exam_id,
         rel_users_exams.started_at AS rel_users_exams_started_at,
         rel_users_exams.finished_at AS rel_users_exams_finished_at,
         answer_1.id AS answer_1_id,
         answer_1.text AS answer_1_text,
         answer_1.correct AS answer_1_correct,
         answer_1.fraction AS answer_1_fraction,
         answer_1.question_id AS answer_1_question_id,
         question_1.id AS question_1_id,
         question_1.title AS question_1_title,
         question_1.text AS question_1_text,
         question_1.file AS question_1_file,
         question_1.type AS question_1_type,
         question_1.source AS question_1_source,
         question_1.exam_id AS question_1_exam_id,
         exam_1.id AS exam_1_id,
         exam_1.title AS exam_1_title,
         exam_1.date_from AS exam_1_date_from,
         exam_1.date_to AS exam_1_date_to,
         exam_1.created AS exam_1_created,
         exam_1.created_by_ AS exam_1_created_by_,
         exam_1.duration AS exam_1_duration,
         exam_1.success_threshold AS exam_1_success_threshold,
         exam_1.published AS exam_1_published
FROM rel_users_exams LEFT OUTER
JOIN exam AS exam_1
    ON exam_1.id = rel_users_exams.exam_id LEFT OUTER
JOIN question AS question_1
    ON exam_1.id = question_1.exam_id LEFT OUTER
JOIN answer AS answer_1
    ON question_1.id = answer_1.question_id
WHERE rel_users_exams.user_username = %(param_1)s
        AND rel_users_exams.exam_id = %(param_2)s
ORDER BY  question_1.id;
```

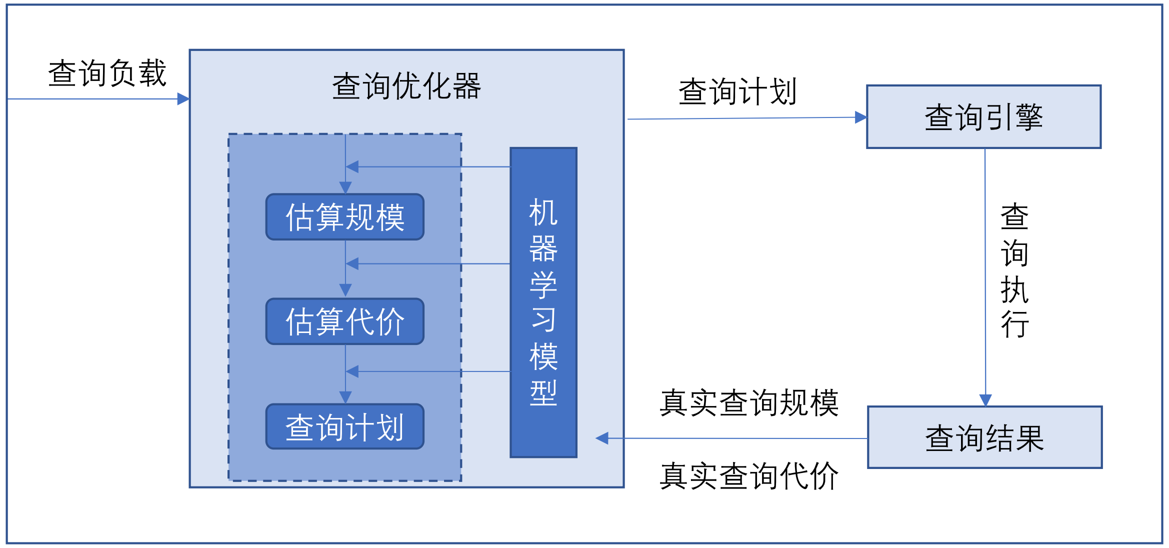


```
#1 Sort
by question_1.id

#2 Nested Loop
Left join

#3 Nested Loop
Left join

#4 Seq Scan
on public.rel_users_exams as rel_users_exams

#5 Seq Scan
on public.exam as exam_1

#6 Hash Join
Right join
on answer_1.question_id = question_1.id

#7 Seq Scan
on public.answer as answer_1

#8 Hash

#9 Seq Scan
on public.question as question_1
```

Cost-based optimizer

- A cost-based optimizer introduces a plan enumeration algorithm to find a (sub)plan, and then uses a cost model to obtain the cost of that plan, and selects the plan with the lowest cost.
- In the cost model, cardinality, the number of tuples through an operator, plays a crucial role.

Why learning-based methods can help?



- Machine Learning can be used in:
  - Modeling data distribution & data correlation
  - Optimize function parameter
  - Markov decision process

Cardinality/Selectivity Estimation

What is Cardinality/Selectivity

```
:  SELECT *
   FROM         WHERE     > 15
   AND        = 'Male';
```

Card(Q) = 4
Sel(Q) = Card(Q) / #row
=4/9 = 0.444

age	gender	GPA
21	Female	3.42
20	Male	2.58
18	Female	2.79
20	Female	3.98
24	Female	3.71
20	Male	3.50
21	Male	4.0
23	Female	3.66
22	Male	3.12

How Learned Selectivity Estimators Work

- Methodology 1: **Query-driven**
  - Key Idea: Model as a Regression problem

$$\text{Query} \xrightarrow{\text{Feature Engineering}} \text{Feature Vector} \xrightarrow{\text{ML Models}} \text{Selectivity}$$

- Methodology 2: **Data-driven**
  - Key Idea**:** Model as a Joint Distribution Estimation problem

A_1	A_2	…	A_n

$$\mathbf{P(A_{1,} A_{1,} \dots , A_n)}$$

Methodology 1: Query-Driven

Training

Query Pool	Labels
$Q1$: SELECT $*$FROM $Student$ WHERE age > 20;	4
$Q2$: SELECT $*$FROM $Student$ WHERE GPA < 3.5 AND GPA > 3.0;	2
$Q3$: SELECT $*$FROM $Student$ WHERE $gender$ = 'Female';	5
…	…

$\downarrow$ **Featurize**

$Q1$: <0.8, 1.0, 0.0, 0.0, 0.0, 1.0>	4
$Q2$: <0.0, 1.0, 0.0, 1.0, 0.3, 0.6>	2
$Q3$: <0.0, 1.0, 1.0, 1.0, 0.0, 1.0>	5
…	

$\downarrow$ **Train**

Regression Model

Inference

Q: SELECT $*$ FROM $Student$
WHERE age > 15 AND gender = "Male"

$\downarrow$ **Featurize**

Q: <0.0, 0.9, 0.0, 1.0, 0.8, 1.0>

$\downarrow$ **Inference**

Estimation: 4!

Methodology 1: Query-Driven

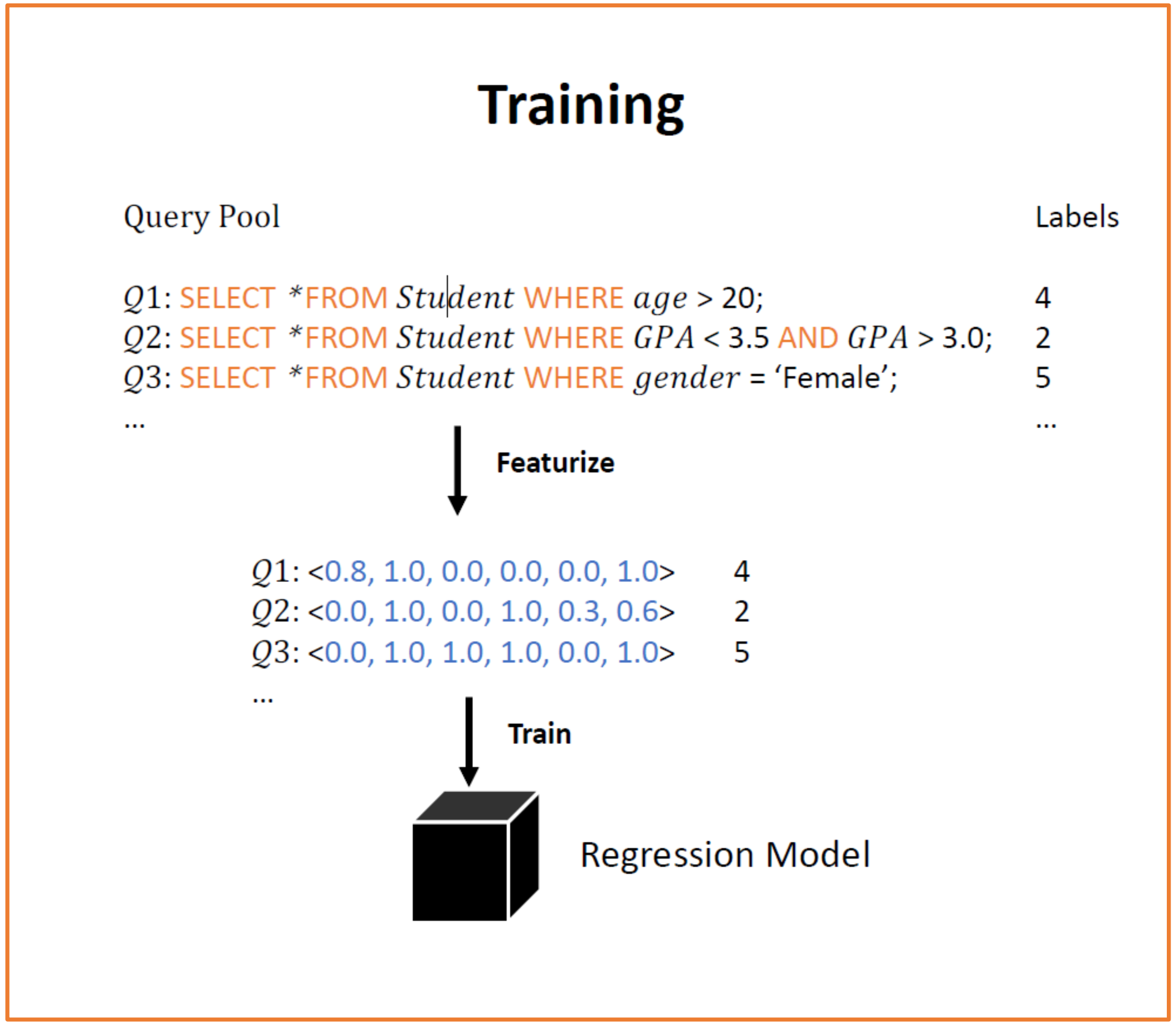
Single table:

- **LW-XGB** [Dutt, A et all. VLDB 19]

  XGBoost+ Heuristic statistics
- **LW-NN** [Dutt, A et all. VLDB 19]

  NN+ Heuristic statistics
- **QuickSel** [Yongjoo, P et all. SIGMOD 20]

  Mixture Model



Query Pool	Labels
$Q1$: SELECT *FROM $Student$ WHERE age > 20;	4
$Q2$: SELECT *FROM $Student$ WHERE GPA < 3.5 AND GPA > 3.0;	2
$Q3$: SELECT *FROM $Student$ WHERE $gender$ = 'Female';	5
…	…

$Q1$: <0.8, 1.0, 0.0, 0.0, 0.0, 1.0>	4
$Q2$: <0.0, 1.0, 0.0, 1.0, 0.3, 0.6>	2
$Q3$: <0.0, 1.0, 1.0, 1.0, 0.0, 1.0>	5
…	

Methodology 1: Query-Driven

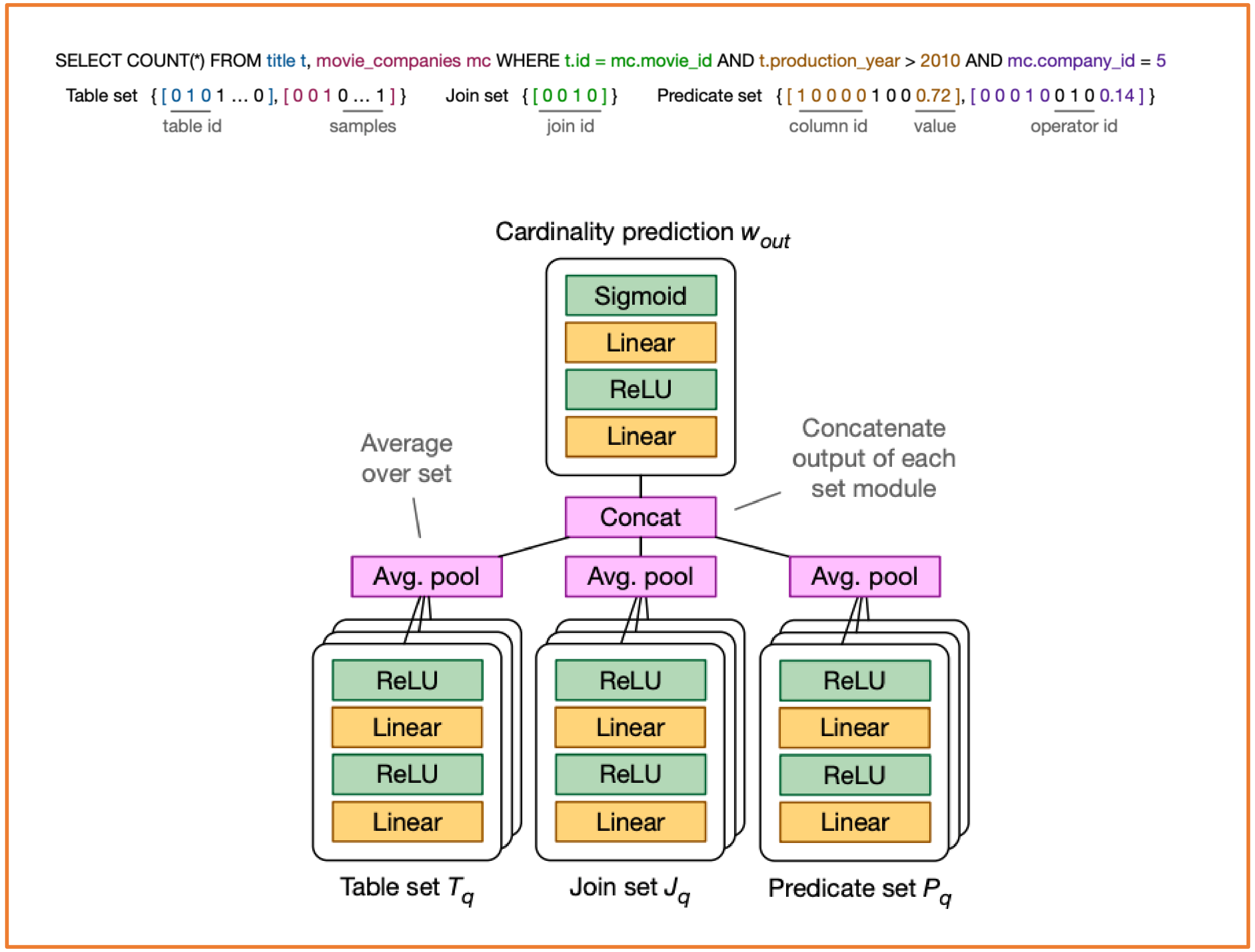
Multiple tables:

- **LW-XGB** [Dutt, A et all. VLDB 20]

  XGBoost + incremental data gen

- **MSCN** [Kipf, A et all. CIDR 19]

  Neural Network + Sampling



```
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 0.14 ] }
```

Query-Driven Shortcomings:

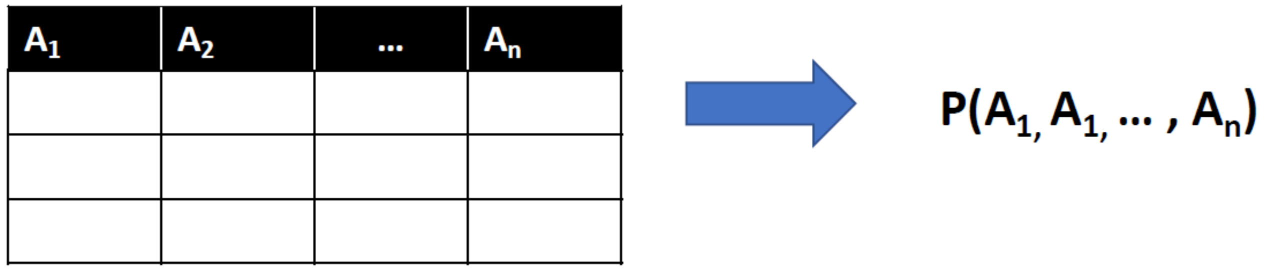
- Require large amount of training data
- Violate basic rule of selectivity estimation
    - - Monotonicity: $sel(1 < x < 2) \leq sel(1 < x < 3)$
    - - Validity: $sel(1 < x < 0) = 0$
    - - Consistency: $sel(1 < x \leq 2) + sel(2 < x < 3)$
    - $= sel(1 < x < 3)$

How Learned Selectivity Estimators Work

- Methodology 1: **Query-driven**
  - Key Idea: Model as a Regression problem

$$\begin{tikzcd}\text{Query} \arrow[r, "\text{Feature Engineering}"] & \text{Feature Vector} \arrow[r, "\text{ML Models}"] & \text{Selectivity}\end{tikzcd}$$

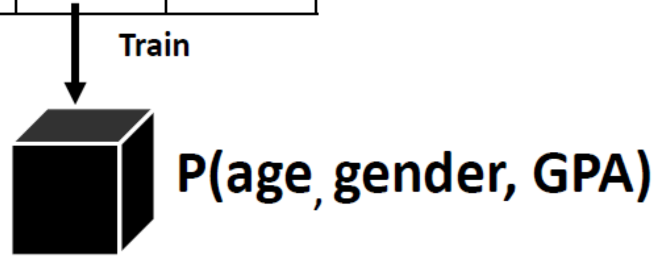
- Methodology 2: **Data-driven**
  - Key Idea**:** Model as a Joint Distribution Estimation problem



Methodology 2: Data-Driven

Training

age	gender	GPA
21	Female	3.42
20	Male	2.58
18	Female	2.79
20	Female	3.98
24	Female	3.71
20	Male	3.50
21	Male	4.0
23	Female	3.66
22	Male	3.12

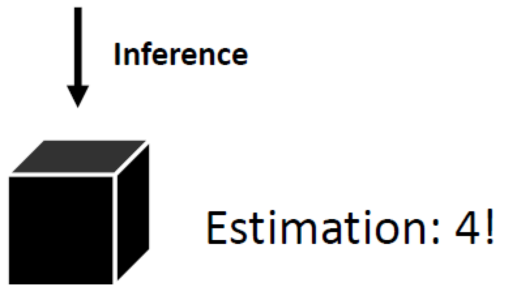


Joint Distribution Estimation Model

Inference

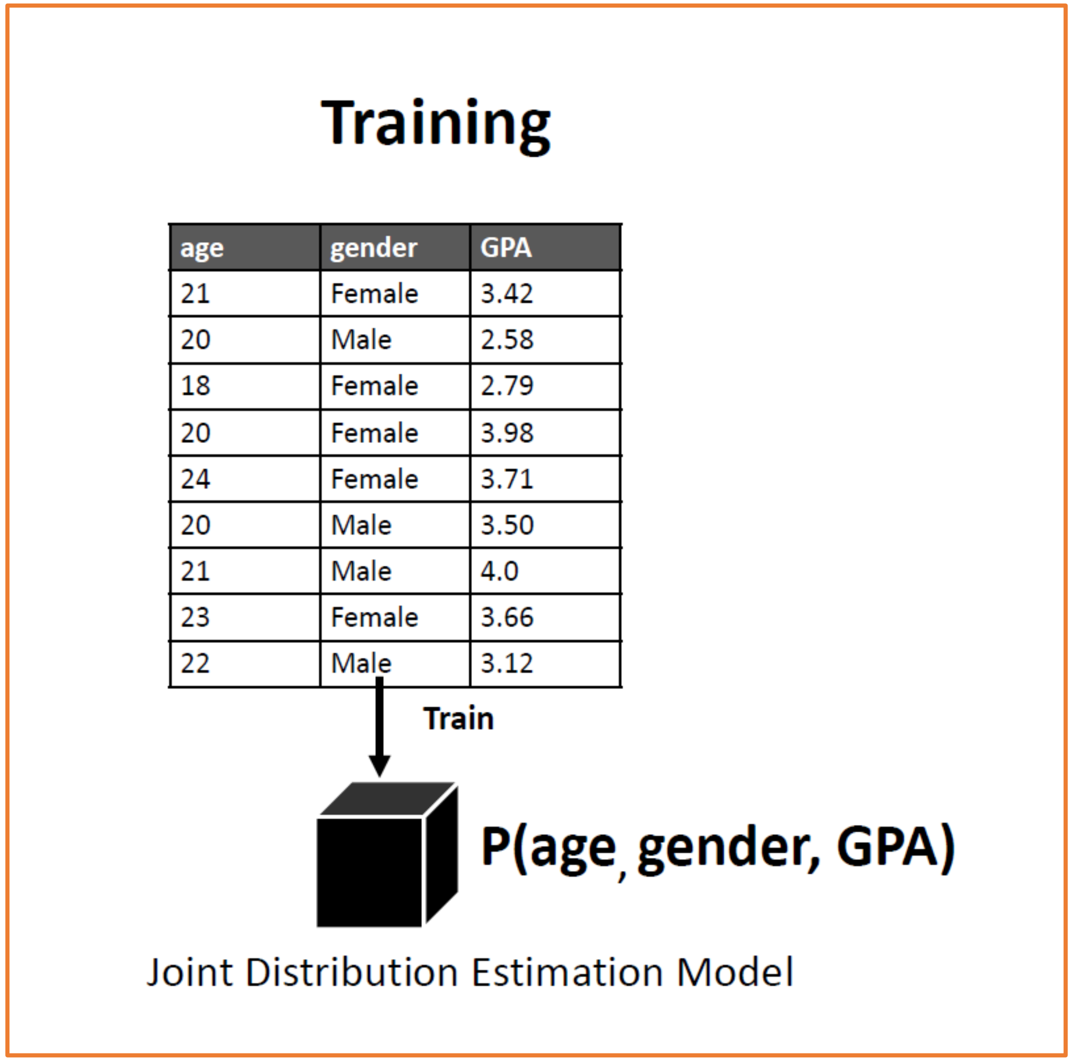
Q: SELECT * FROM $Student$
WHERE age > 15 AND gender = “Male”

$$P(age > 20, \text{gender} = \text{“Male”})$$



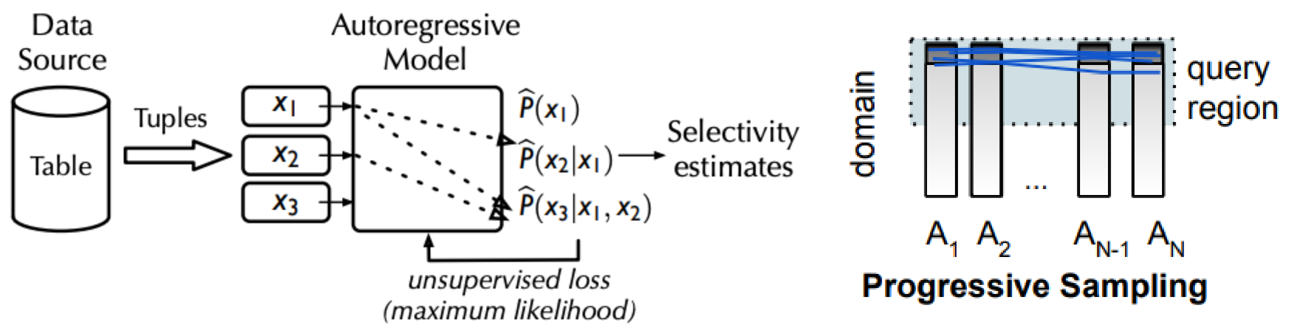
Methodology 2: Data-Driven

- **Naru** [Yang, Z et all. VLDB 20]
  Auto-regressive Model
- **DeepDB** [Hilprecht, B et all. VLDB 20]
  Sum Product Network
  (Graphical Model)
- **FACE** [Wang, J et all. VLDB 2022]
  Normalizing Flow

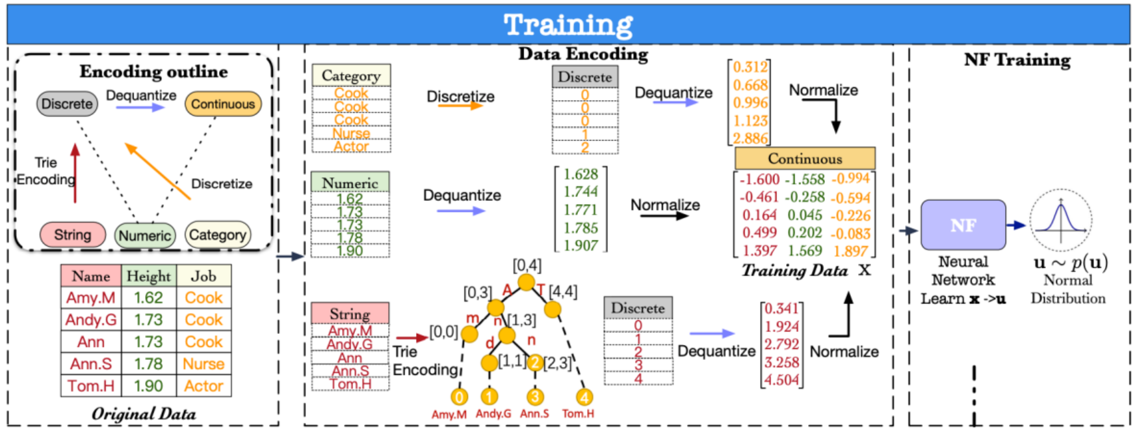


age	gender	GPA
21	Female	3.42
20	Male	2.58
18	Female	2.79
20	Female	3.98
24	Female	3.71
20	Male	3.50
21	Male	4.0
23	Female	3.66
22	Male	3.12

Data-Driven Methods



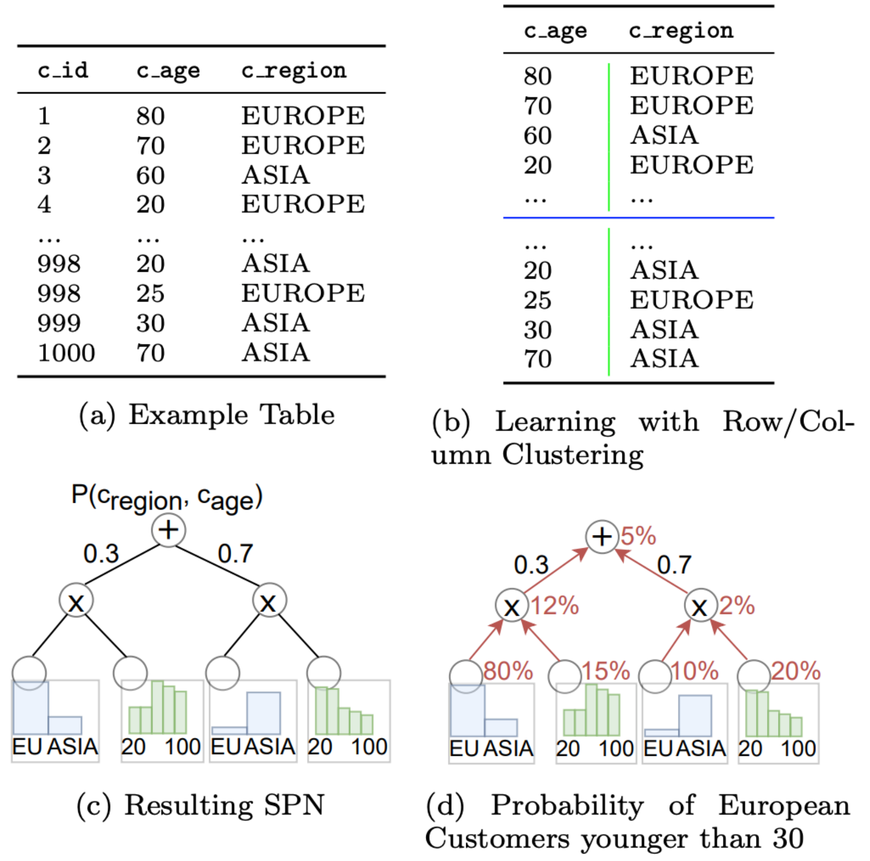
Naru: Auto-regressive model



Name	Height	Job
Amy.M	1.62	Cook
Andy.G	1.73	Cook
Ann	1.73	Cook
Ann.S	1.78	Nurse
Tom.H	1.90	Actor

Original Data

FACE: Normalizing flow



c_id	c_age	c_region
1	80	EUROPE
2	70	EUROPE
3	60	ASIA
4	20	EUROPE
...	...	...
998	20	ASIA
998	25	EUROPE
999	30	ASIA
1000	70	ASIA

(a) Example Table

c_age	c_region
80	EUROPE
70	EUROPE
60	ASIA
20	EUROPE
...	...
...	...
20	ASIA
25	EUROPE
30	ASIA
70	ASIA

(b) Learning with Row/Column Clustering

(c) Resulting SPN

(d) Probability of European Customers younger than 30

DeepDB: Sum-product network

Shortcomings

- Heavy costs on model training and inference
- Violate basic rule of selectivity estimation
  - - Monotonicity: $sel(1 < x < 2) \leq sel(1 < x < 3)$
  - - Validity: $sel(1 < x < 0) = 0$
  - - Consistency: $sel(1 < x \leq 2) + sel(2 < x < 3)$
  - $= sel(1 < x < 3)$
  - - **Stability**: $sel(1 < x < 2) = sel(1 < x < 2)$

MOSE: A Monotonic Selectivity Estimator Using Learned CDF

- **Aim**

  Reliable, accurate and efficient learned selectivity estimator

- **Problem Settings**

  Multi-dimensional predicates on single table

- **Methodology**

  Query-based

- **Key observation**

  The joint cumulative distribution function (CDF) of the data in a table can be used to compute the selectivity for query range predicates

CDF to Selectivity

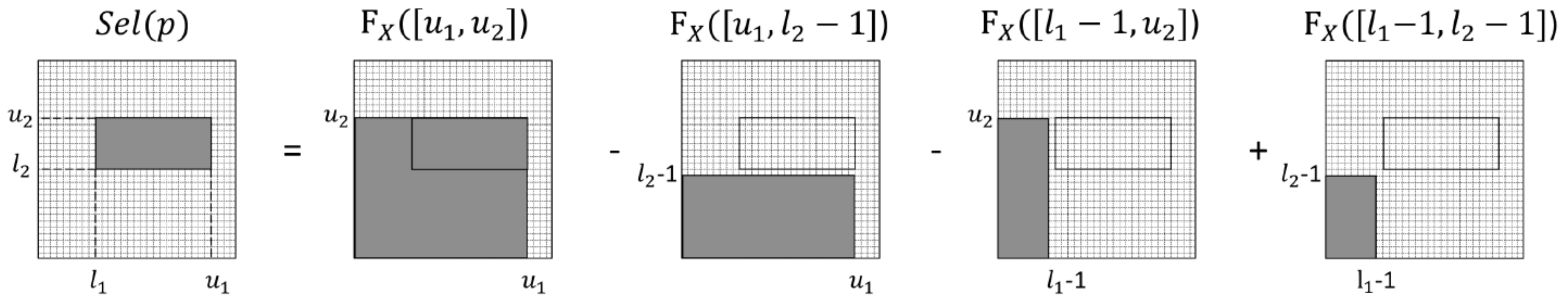
- Multi-dimensional CDF:

  Random variable $X = (X_1, X_2, \cdots, X_d)$,

  CDF: $F_X(x) = \Pr(X_1 \le x_1, X_2 \le x_2, \cdots, X_d \le x_d)$
- CDF to selectivity

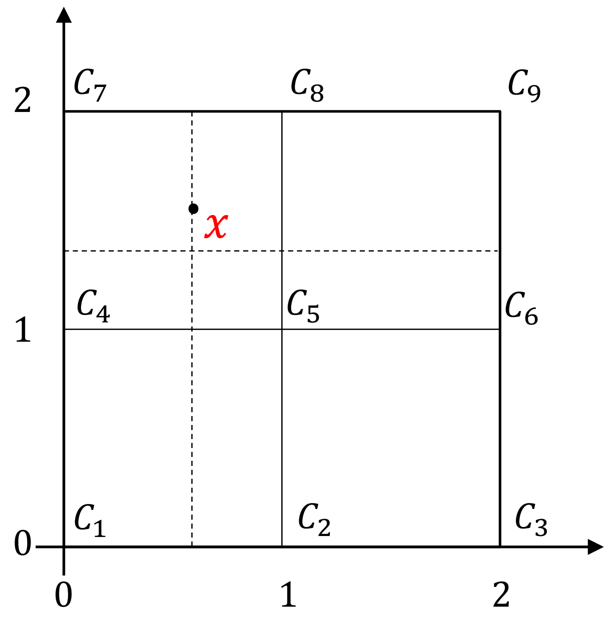
$$sel(p) = \sum_{\forall x_i \in \{l_i - 1, u_i\}} \left\{ \left(\prod_{i=1}^{d} s(i) \right) F_X(x) \right\}$$

$$\begin{aligned} sel(p) = & F_X([u_1, u_2]) - F_X([u_1, l_2 - 1]) \\ & - F_X([l_1 - 1, u_2]) + F_X([l_1 - 1, l_2 - 1]) \end{aligned}$$



CDF Learner

- **Monotonic Lattice Regression Model**



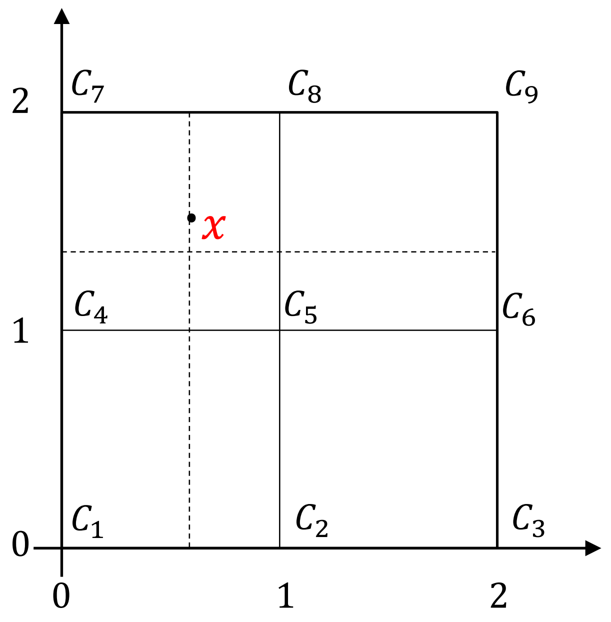
C: Lattice Node
Theta: Lattice Parameter

$$\hat{y} = F_X(x) = \textstyle\sum_{j=1}^{m} \phi(x)_j \theta_j, \quad \sum_{j=1}^{m} \phi(x)_j C_j = x, \quad \sum_{j=1}^{m} \phi(x)_j = 1.$$

$$\begin{aligned} \theta &= \arg\min_{\theta \in \mathbb{R}^m} \textstyle\sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \\ &= \arg\min_{\theta \in \mathbb{R}^m} \textstyle\sum_{i=1}^{n} \left(\left(\sum_{j=1}^{m} \phi(x)_{ij} \theta_j \right) - y_i \right)^2 . \end{aligned}$$

$$\theta = \arg\min_{\theta \in \mathbb{R}^m} \textstyle\sum_{i=1}^{n} (\hat{y}_i - y_i)^2, s.t.\ A\theta^T \leq 0.$$

Monotonic Constrain

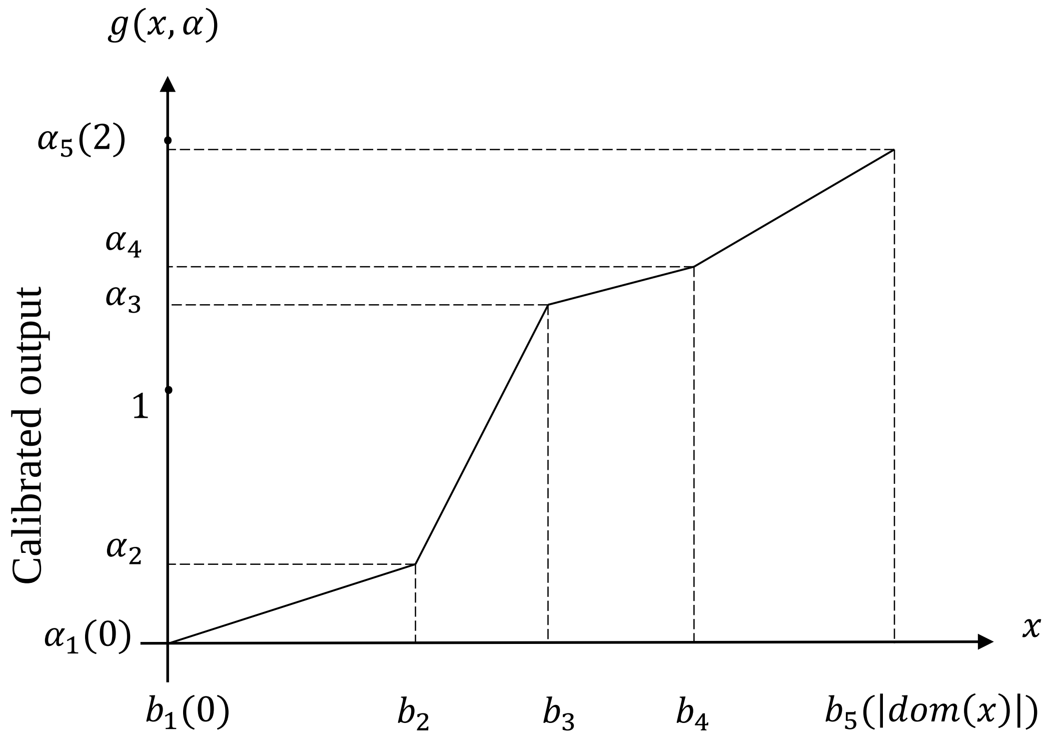


C: Lattice Node
Theta: Lattice Parameter

$$\theta = \arg\min_{\theta \in \mathbb{R}^m} \sum_{i=1}^{n} \left(\hat{y}_i - y_i\right)^2, s.t.\ A\theta^T \le 0.$$

$$\begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \\ \theta_8 \\ \theta_9 \end{bmatrix} \le \mathbf{0} \Leftrightarrow \begin{array}{l} \theta_1 \le \theta_2 \\ \theta_2 \le \theta_3 \\ \theta_4 \le \theta_5 \\ \theta_5 \le \theta_6 \\ \theta_7 \le \theta_8 \\ \theta_8 \le \theta_9 \\ \theta_1 \le \theta_4 \\ \theta_2 \le \theta_5 \\ \theta_3 \le \theta_6 \\ \theta_4 \le \theta_7 \\ \theta_5 \le \theta_0 \\ \theta_6 \le \theta_9 \end{array}$$

Attribute-Aware Calibration

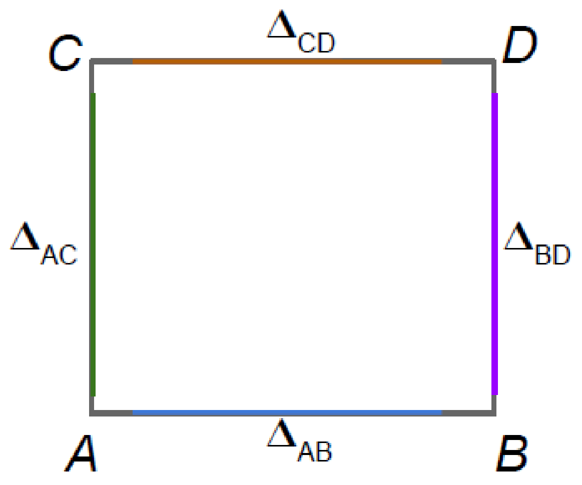


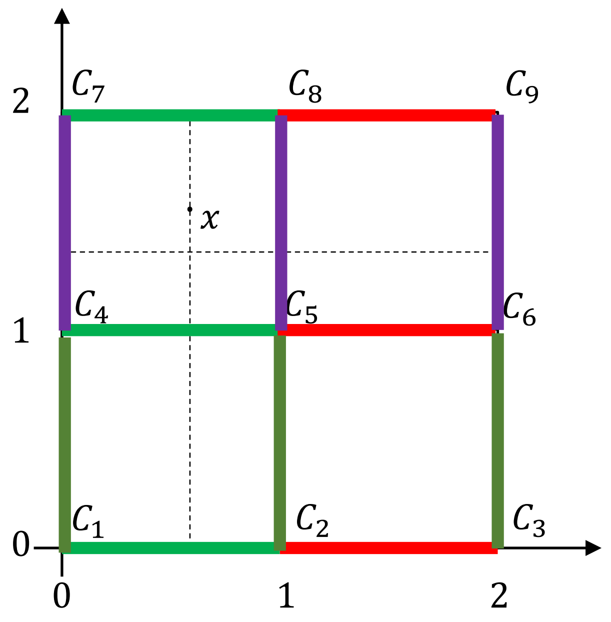
$$\theta, \alpha = \arg\min_{\theta,\alpha} \sum_{i=1}^{n} \left(\left(\sum_{j=1}^{m} \phi(g(x,\alpha))_{ij}\theta_j \right) - y_i \right)^2 + \lambda R(\theta)$$
$$s.t.\ \ A\theta^T \leq 0 \ and \ B\alpha^T \leq 0 \,,$$

Cell-Wise Regularizer

Graph Laplacian:
flatter function

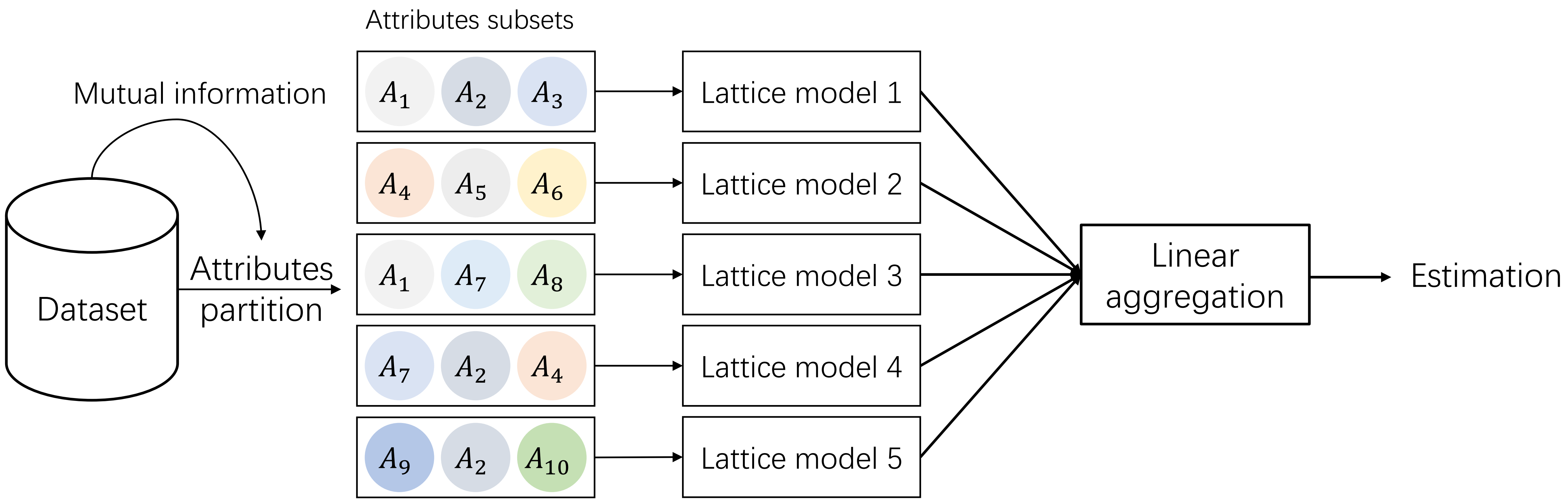
Penalizes:
$(A-C)^2 + (A-B)^2 + (C-D)^2 + (B-D)^2$
$= \Delta_{AC}^2 + \Delta_{AB}^2 + \Delta_{CD}^2 + \Delta_{BD}^2$





$$R(\theta) = \sum_{i=1}^{d} \sum_{\substack{C_r, C_s \text{ such that} \\ C_r \text{ and } C_s \text{ adjacent on dimension } i}} H_i(C_r, C_s)(\theta_r - \theta_s)^2$$

Lattice Ensemble



ACTIVE DATA GENERATOR

- Challenges:
  - Infinite query space (NOT pool based active learning)
  - Regression problem (NO model uncertainty)
- Solution
  - Picking the lattice cells that are most valuable or necessary to optimize
  - Two factors: (1) cell accuracy; (2) cell density
  - Weighted lattice sampling

Weighted Lattice Sampling

Algorithm 2: Active Data Generator

```
input : $X_L$ is the initial labeled data,
        $\theta$ is model weight of CDF learner,
        $\mathcal{T}$ is the cost threshold of data collection,
        $\epsilon$ is the cost function to label a data instance,
        $B$ is the number of data selected in one batch,
        $\mathcal{P}$ is a function to calculate cell sampling weight
output: $X$ is the labeled training data

$X \leftarrow X_L$ // total training set
$t \leftarrow 0$ // initialize total cost
while $t < \mathcal{T}$ do
    $\theta \leftarrow$ TrainModelWith($X$)
    $Error \leftarrow$ Evaluate($\theta, X$)
    $P_{TopError} \leftarrow$ Top($X, Error, K$)
    $\mathcal{P}_c \leftarrow X, P_{TopError}$
    $cells \leftarrow$ WeightedSampling($\mathcal{P}_c, B$)
    $X_A \leftarrow$ RandomPointGenerate($cells$)
    $X_{AL} \leftarrow$ ExecuteQuery($X_A$)
    $t = t + \epsilon(X_A)$
    $X = X \cup X_{AL}$
return $X$
```

$$\mathcal{P}_c = \frac{1 + \omega k_c}{1 + M_c} ,$$

M_c: points in cell C

k_c: k points of C are in the TOP-K worst estimation

Accuracy

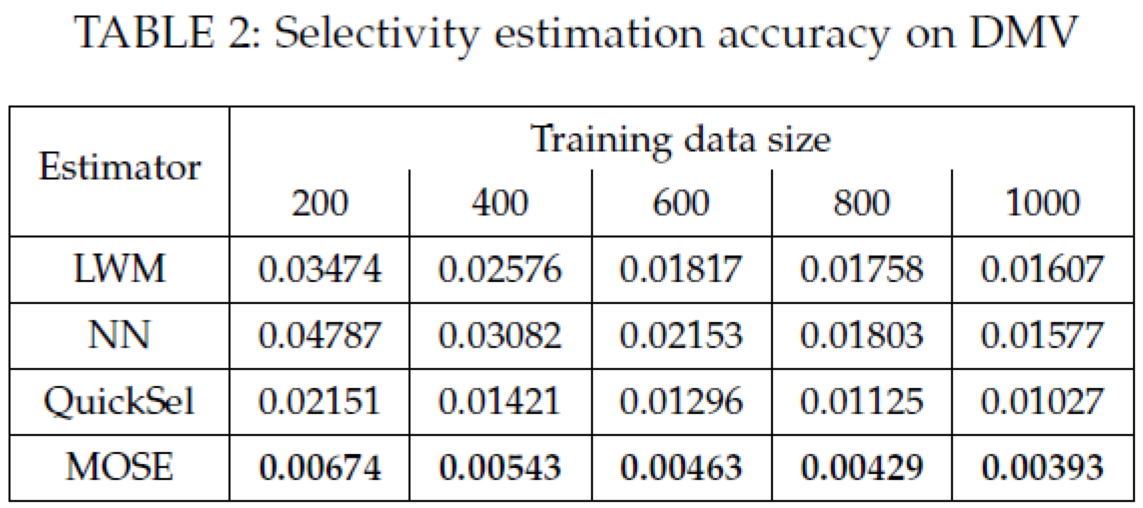


TABLE 2: Selectivity estimation accuracy on DMV

Estimator	Training data size				
	200	400	600	800	1000
LWM	0.03474	0.02576	0.01817	0.01758	0.01607
NN	0.04787	0.03082	0.02153	0.01803	0.01577
QuickSel	0.02151	0.01421	0.01296	0.01125	0.01027
MOSE	**0.00674**	**0.00543**	**0.00463**	**0.00429**	**0.00393**

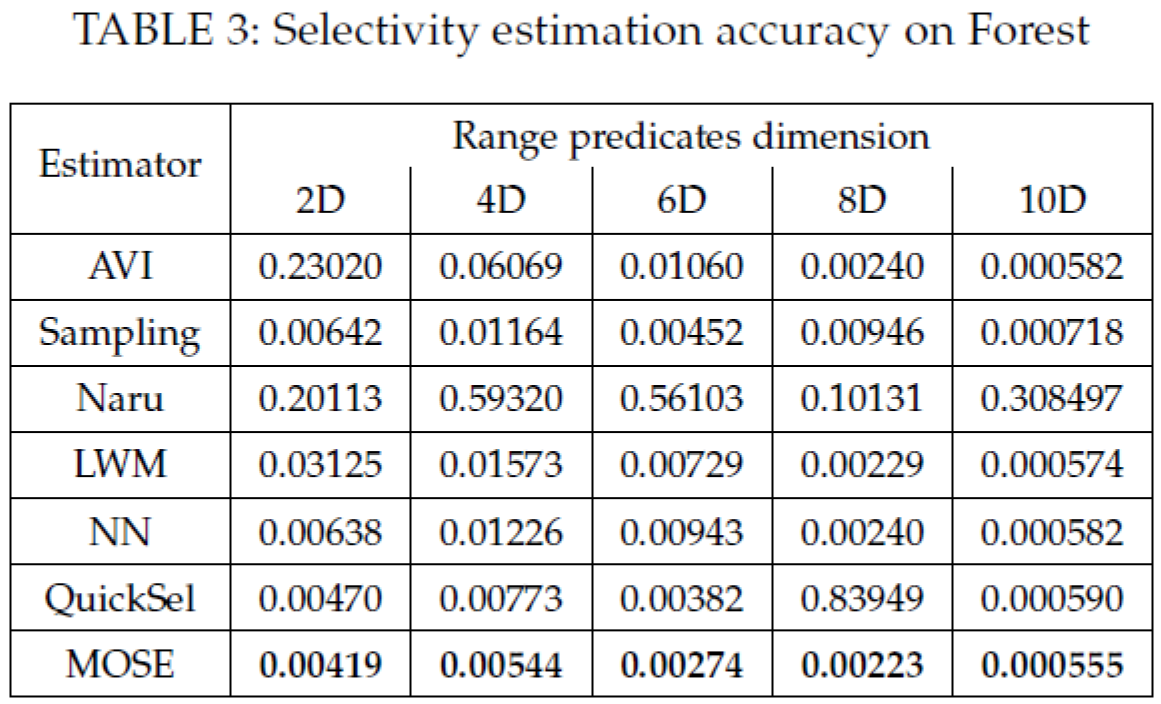
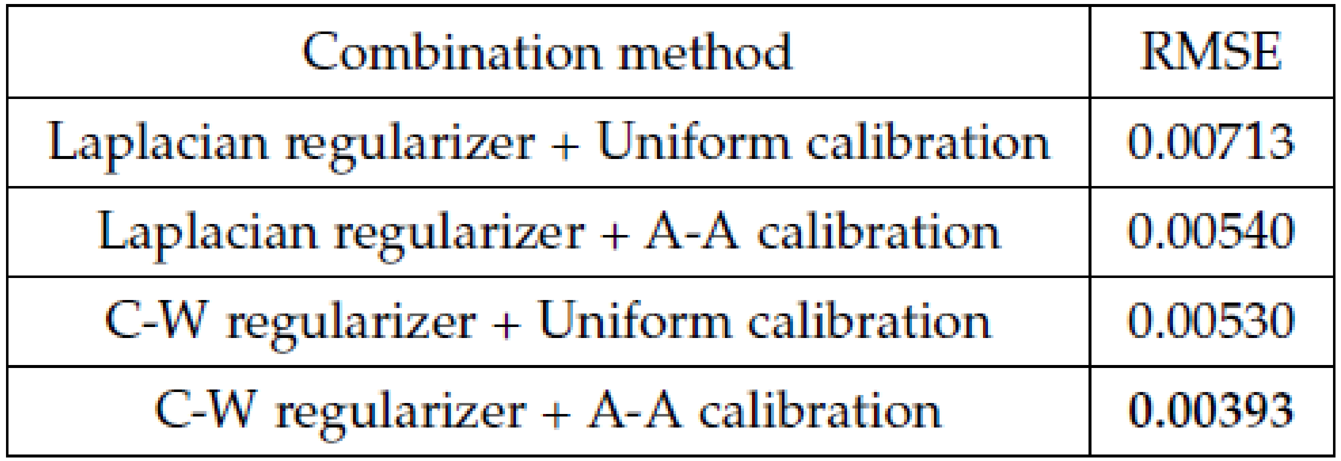


TABLE 3: Selectivity estimation accuracy on Forest

Estimator	Range predicates dimension				
	2D	4D	6D	8D	10D
AVI	0.23020	0.06069	0.01060	0.00240	0.000582
Sampling	0.00642	0.01164	0.00452	0.00946	0.000718
Naru	0.20113	0.59320	0.56103	0.10131	0.308497
LWM	0.03125	0.01573	0.00729	0.00229	0.000574
NN	0.00638	0.01226	0.00943	0.00240	0.000582
QuickSel	0.00470	0.00773	0.00382	0.83949	0.000590
MOSE	**0.00419**	**0.00544**	**0.00274**	**0.00223**	**0.000555**

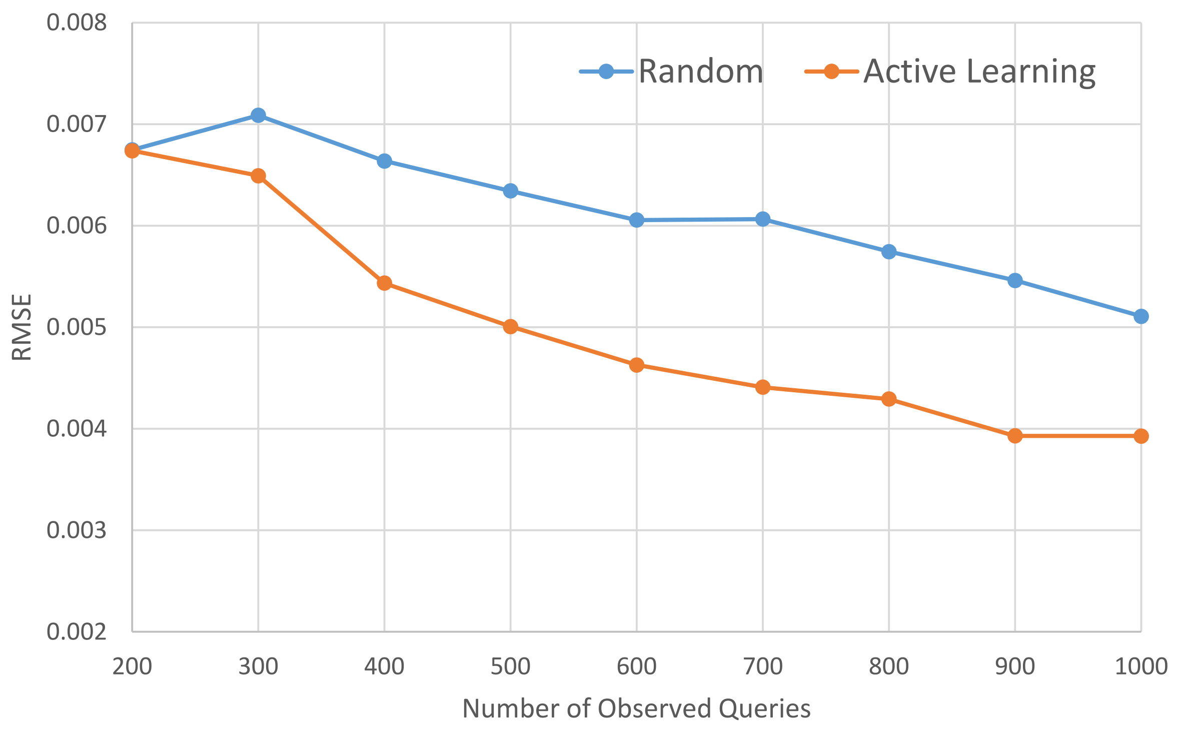
Ablation Experiments

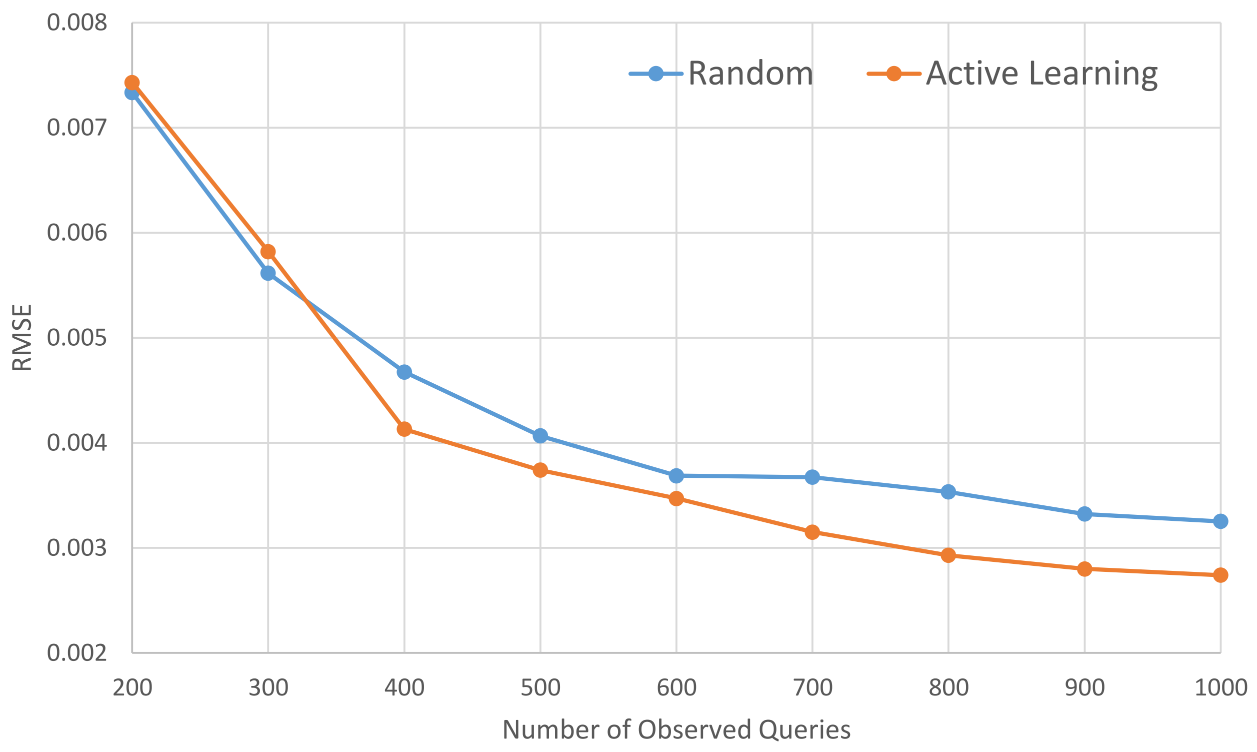
TABLE 4: Combination of calibration and regularizer



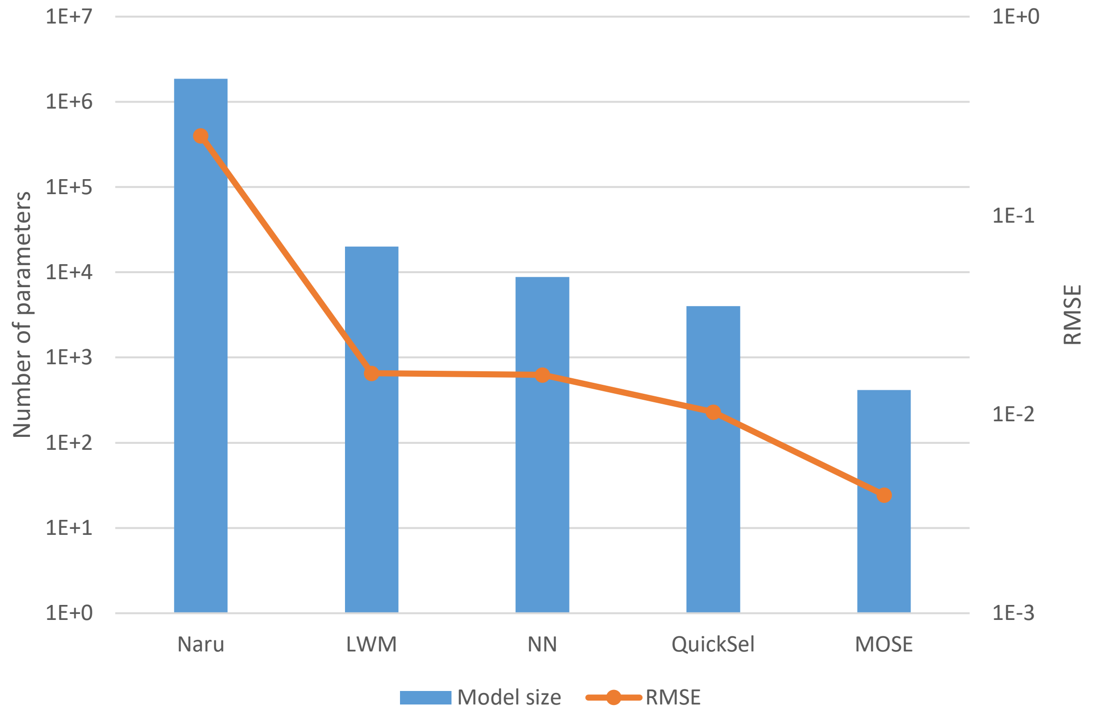
Combination method	RMSE
Laplacian regularizer + Uniform calibration	0.00713
Laplacian regularizer + A-A calibration	0.00540
C-W regularizer + Uniform calibration	0.00530
C-W regularizer + A-A calibration	**0.00393**

Active Learning

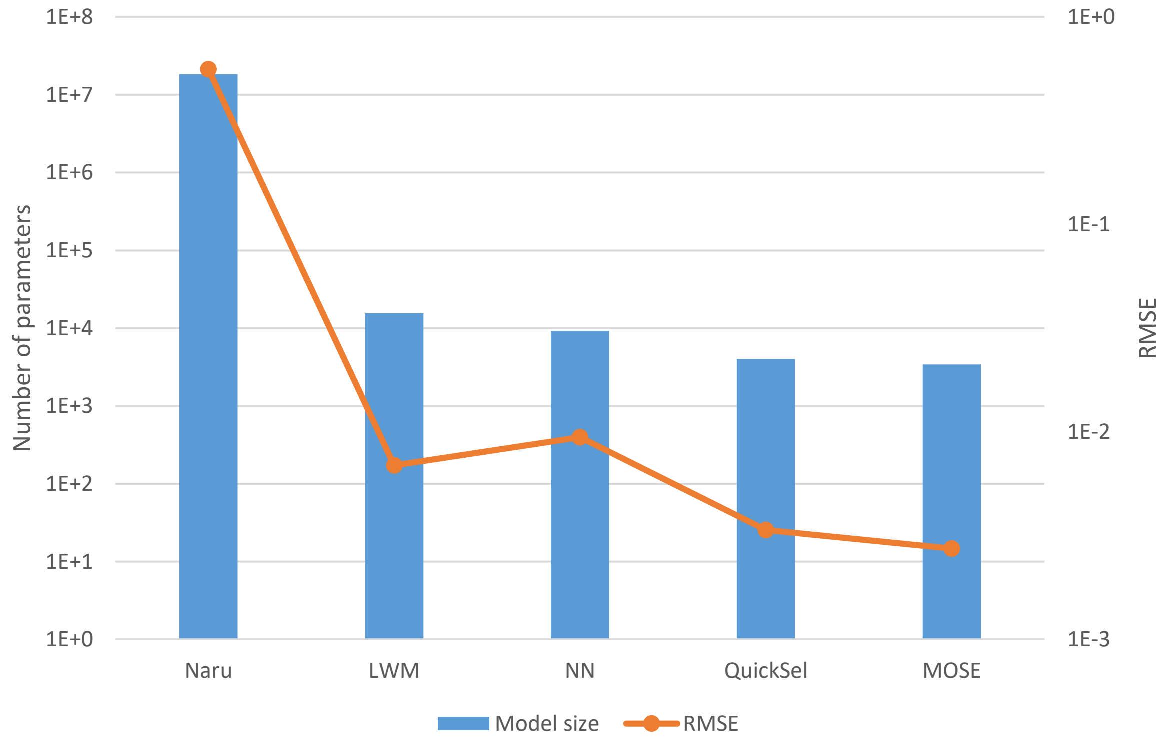




Model Size



1E+7
1E+6
1E+5
1E+4
1E+3
1E+2
1E+1
1E+0
Number of parameters
1E+0
1E-1
1E-2
1E-3
RMSE
Naru
LWM
NN
QuickSel
MOSE
Model size
RMSE



1E+8
1E+7
1E+6
1E+5
1E+4
1E+3
1E+2
1E+1
1E+0
Number of parameters
1E+0
1E-1
1E-2
1E-3
RMSE
Naru
LWM
NN
QuickSel
MOSE
Model size
RMSE

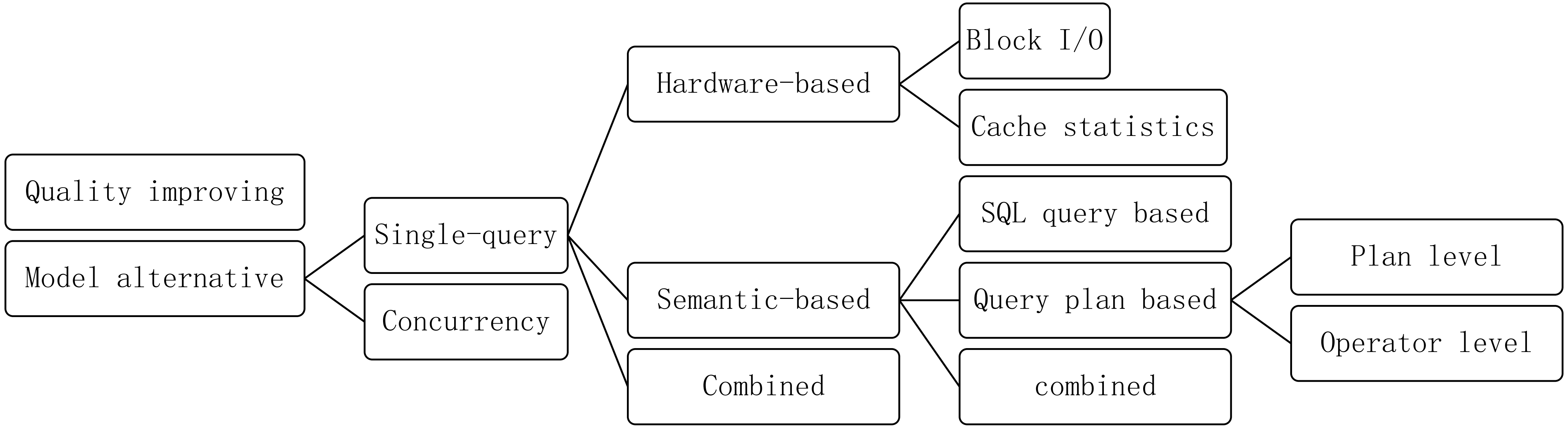
Takeaways

- Learning-based methods can be accurate, efficient, lightweight
- But
  - Have longer-training time
  - May fail to catch up fast data change
  - Hard to trust
  - Are not practical in production

https://www.cs.sfu.ca/~jnwang/ppt/learned-cardinality-estimation.pdf

Cost Estimation

Learning-Based Cost Estimation



Cost model in PostgreSQL

1. ***seq_page_cost***(c_s): the cost of a sequential disk page fetch
2. ***random_page_cost*** (c_r): the cost of fetching a disk page randomly
3. ***cpu_tuple_cost*** (c_t): the cost of processing a tuple
4. ***cpu_index_tuple_cost***(c_i): the cost of processing an index entry during an index scan
5. ***cpu_operator_cost***(c_o): the cost of performing an operation

The cost of an operator, C_o is calculated as [8]:

$$C_o = n_s.c_s + n_r.c_r + n_t.c_t + n_i.c_i + n_o.c_o \tag{2.1}$$

where

n_s: number of disk pages fetched sequentially

n_r: number of disk pages fetched randomly

n_t: number of tuples processed

n_i: number of index entries processed during an index scan

n_o: number of operations performed

Plan-structured cost learner

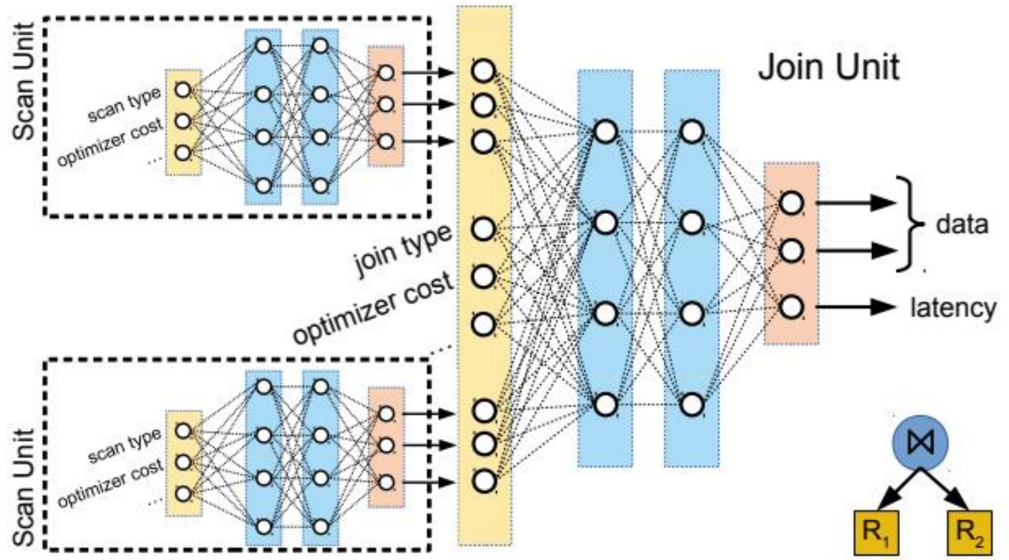


Figure 3: A neural network for a simple join query

Plan-Structured Deep Neural Network Models for Query Performance Prediction (VLDB 2019)

Plan-structured cost learner

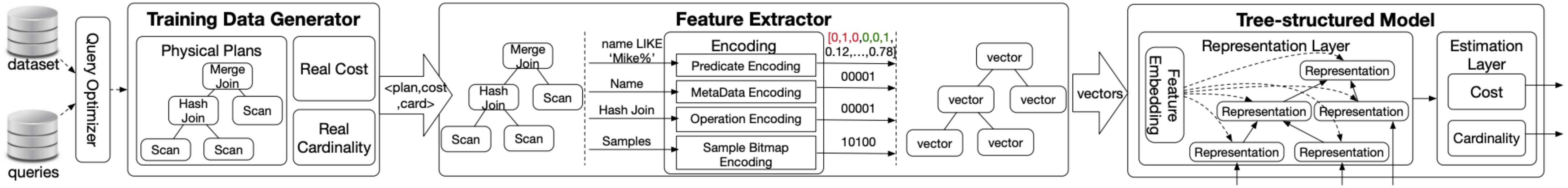


Figure 2: Architecture of learning-based cost estimator

An End-to-End Learning-based Cost Estimator (VLDB 2019)

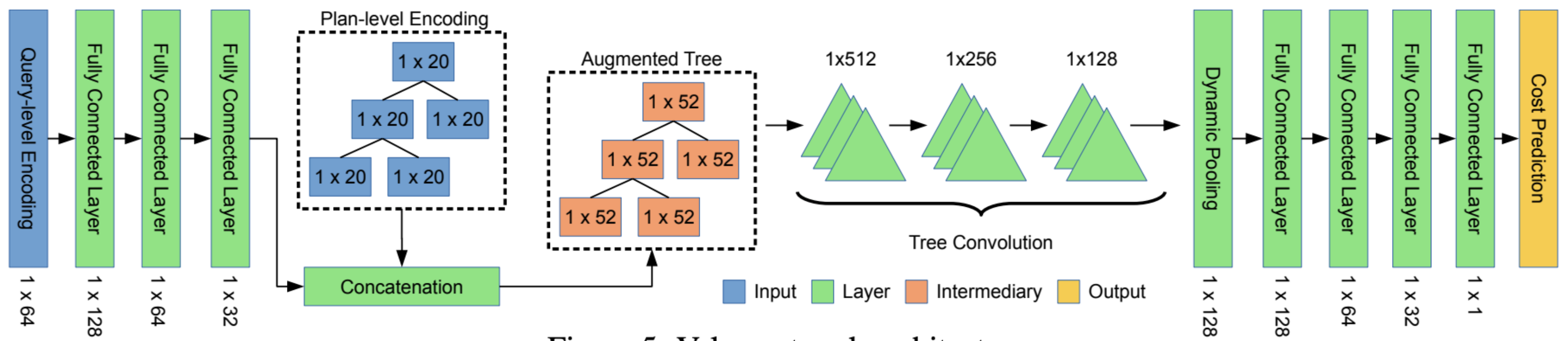


Figure 5: Value network architecture

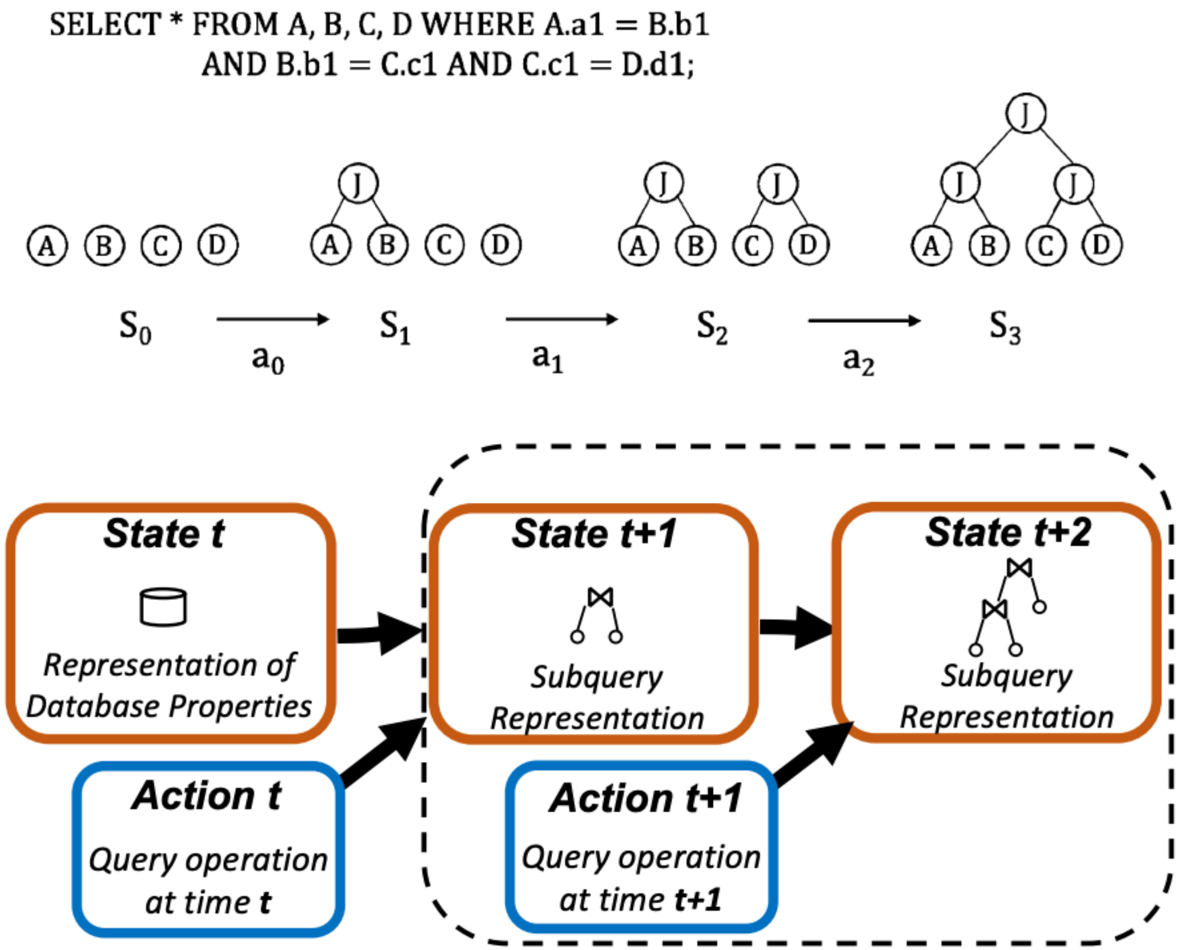
Neo: A Learned Query Optimizer (VLDB 2019)

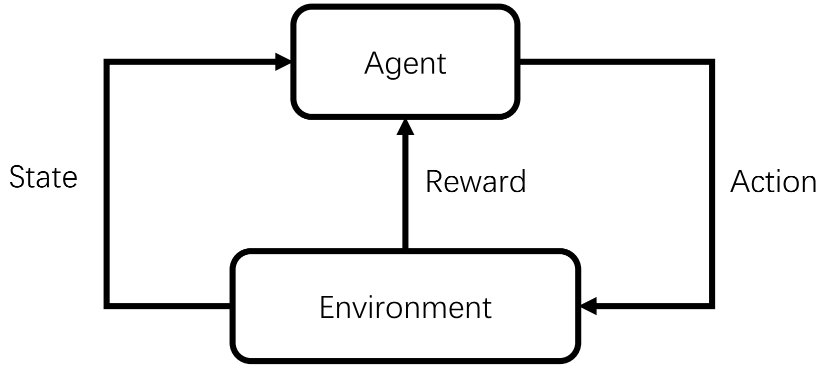
Takeaways

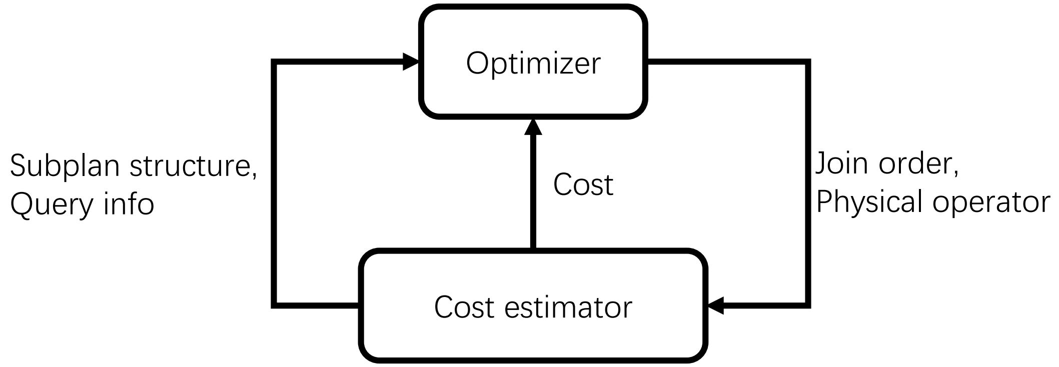
- Learning-based cost estimation are accurate but not so efficient.
- Some learning-based cost estimation methods are used for scheduling not query optimization.
- Learning-based cost estimation (especially tree-structured) are not so practical in traditional cost-based query optimization.

Plan enumeration

Query Plan Optimization


SELECT * FROM A, B, C, D WHERE A.a1 = B.b1
AND B.b1 = C.c1 AND C.c1 = D.d1;
J
A
B
C
D
S_0
a_0
S_1
a_1
S_2
a_2
S_3
State t
Representation of Database Properties
State t+1
Subquery Representation
State t+2
Subquery Representation
Action t
Query operation at time t
Action t+1
Query operation at time t+1





Query Plan Optimization

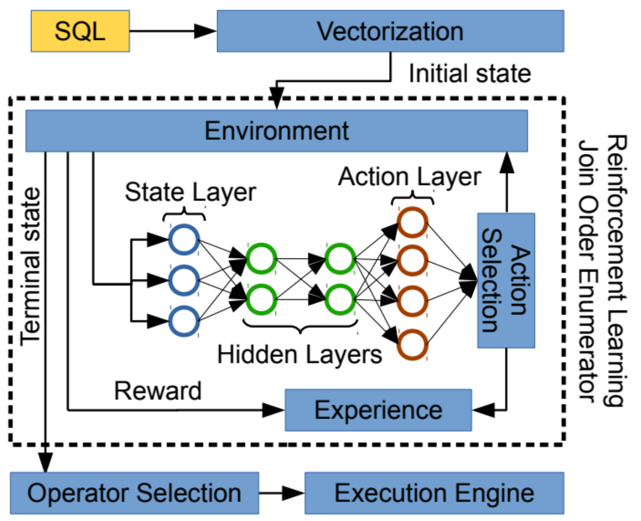


Figure 2: The ReJOIN Framework

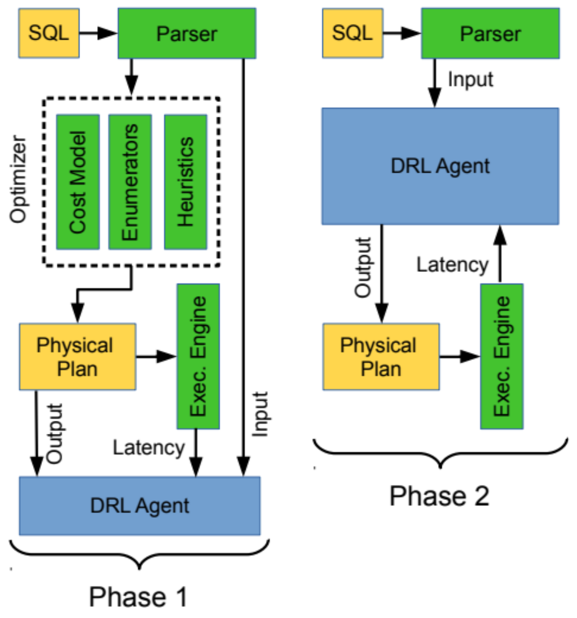


Figure 4: Learning from demonstration

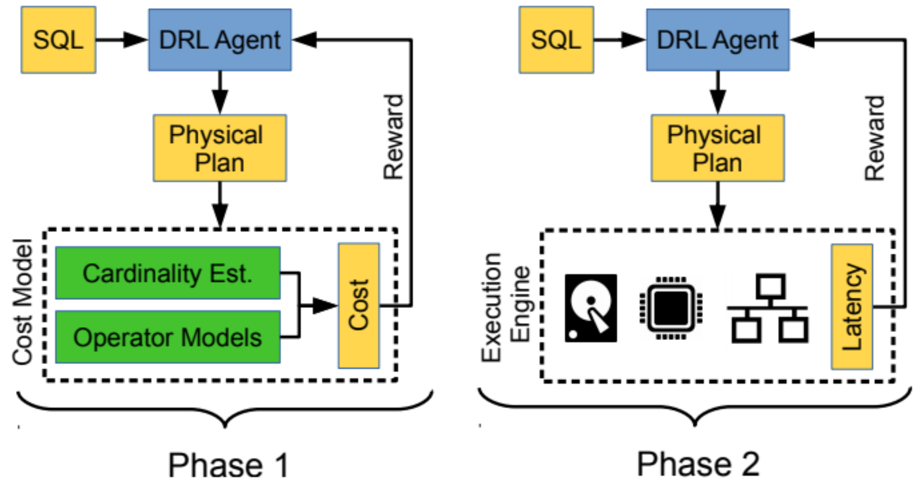


Figure 5: Cost Model Bootstrapping

1. Krishnan, Sanjay, et al. "Learning to optimize join queries with deep reinforcement learning." arXiv preprint arXiv:1808.03196 (2018).
2. Marcus, Ryan, and Olga Papaemmanouil. "Deep reinforcement learning for join order enumeration." Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management (2018): 3.
3. Ortiz, Jennifer, et al. "Learning state representations for query optimization with deep reinforcement learning." arXiv preprint arXiv:1803.08604 (2018).
4. Marcus, Ryan and Olga Papaemmanouil. "Towards a hands-free query optimizer through deep learning. " (CIDR 2019).
5. Marcus, Ryan, et al. "Neo: A learned query optimizer." (VLDB 2019).

Shortcomings

- Long training time
- Inability to adjust to data and workload changes
- Tail catastrophe
- Black-box decisions
- Integration cost

Bao: Making Learned query Optimization Practical (SIGMOD 2021 Best Paper)

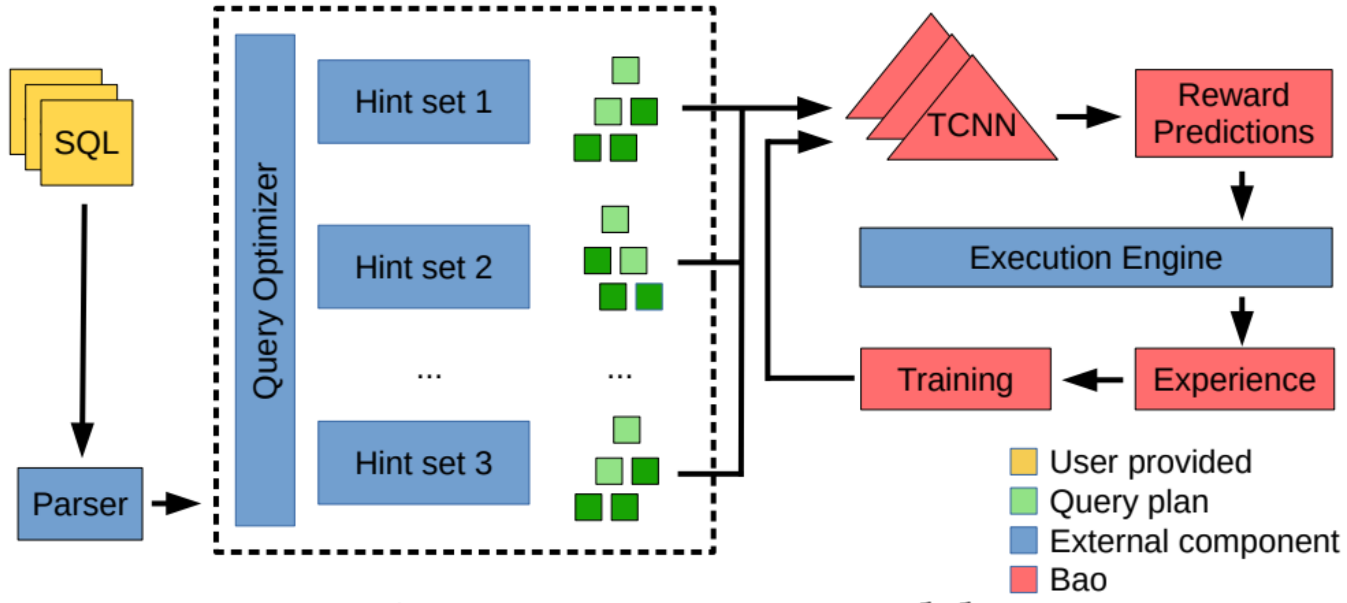
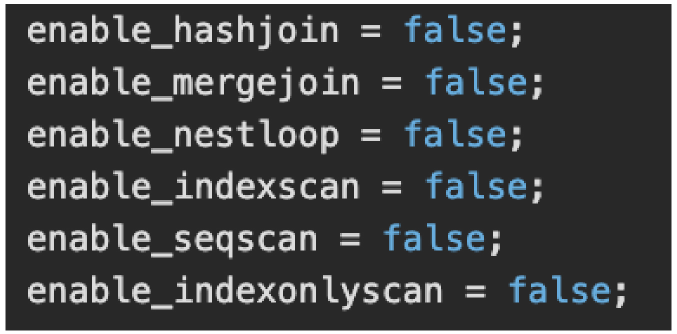
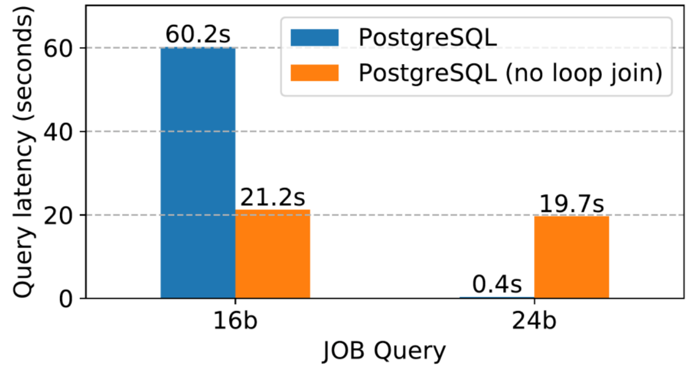


Figure 2: Bao system model



```
enable_hashjoin = false;
enable_mergejoin = false;
enable_nestloop = false;
enable_indexscan = false;
enable_seqscan = false;
enable_indexonlyscan = false;
```



Bao: Making Learned query Optimization Practical (SIGMOD 2021 Best Paper)

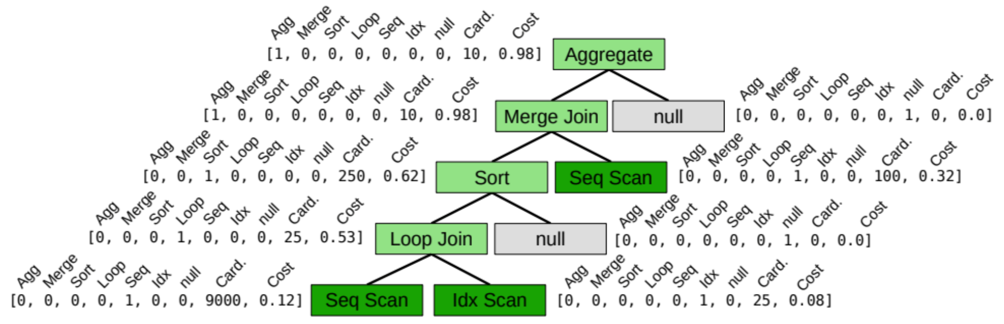

Agg Merge Sort Loop Seq Idx null Card. Cost
[1, 0, 0, 0, 0, 0, 0, 10, 0.98]
Aggregate
Agg Merge Sort Loop Seq Idx null Card. Cost
[1, 0, 0, 0, 0, 0, 0, 10, 0.98]
Merge Join
null
Agg Merge Sort Loop Seq Idx null Card. Cost
[0, 0, 0, 0, 0, 0, 1, 0, 0.0]
Agg Merge Sort Loop Seq Idx null Card. Cost
[0, 0, 1, 0, 0, 0, 0, 250, 0.62]
Sort
Seq Scan
Agg Merge Sort Loop Seq Idx null Card. Cost
[0, 0, 0, 0, 1, 0, 0, 100, 0.32]
Agg Merge Sort Loop Seq Idx null Card. Cost
[0, 0, 0, 1, 0, 0, 0, 25, 0.53]
Loop Join
null
Agg Merge Sort Loop Seq Idx null Card. Cost
[0, 0, 0, 0, 0, 0, 1, 0, 0.0]
Agg Merge Sort Loop Seq Idx null Card. Cost
[0, 0, 0, 0, 1, 0, 0, 9000, 0.12]
Seq Scan
Idx Scan
Agg Merge Sort Loop Seq Idx null Card. Cost
[0, 0, 0, 0, 0, 1, 0, 25, 0.08]

Figure 4: Vectorized query plan tree (vector tree)

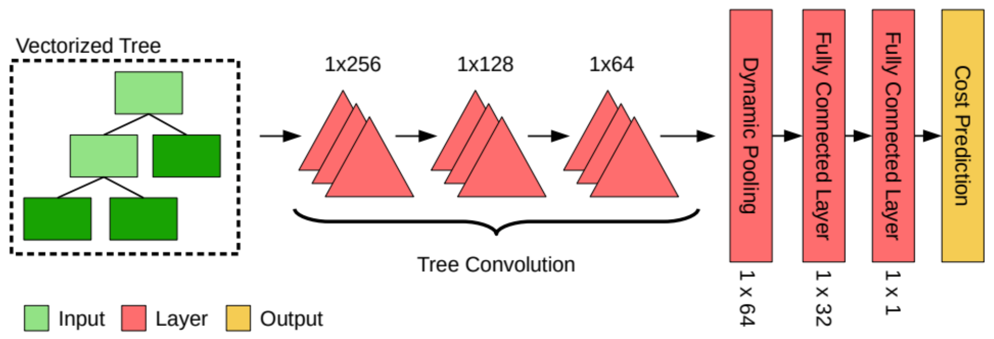
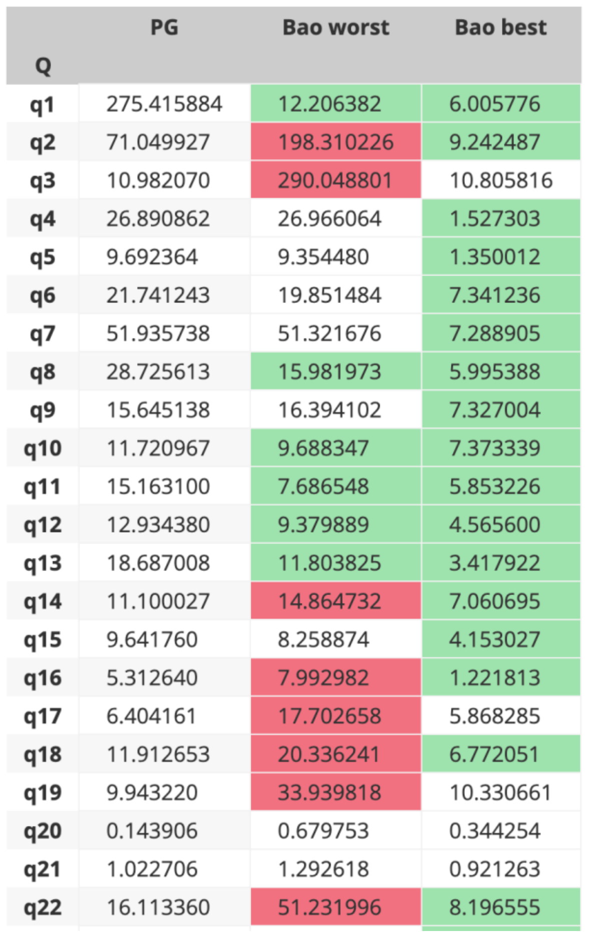


Figure 5: Bao prediction model architecture



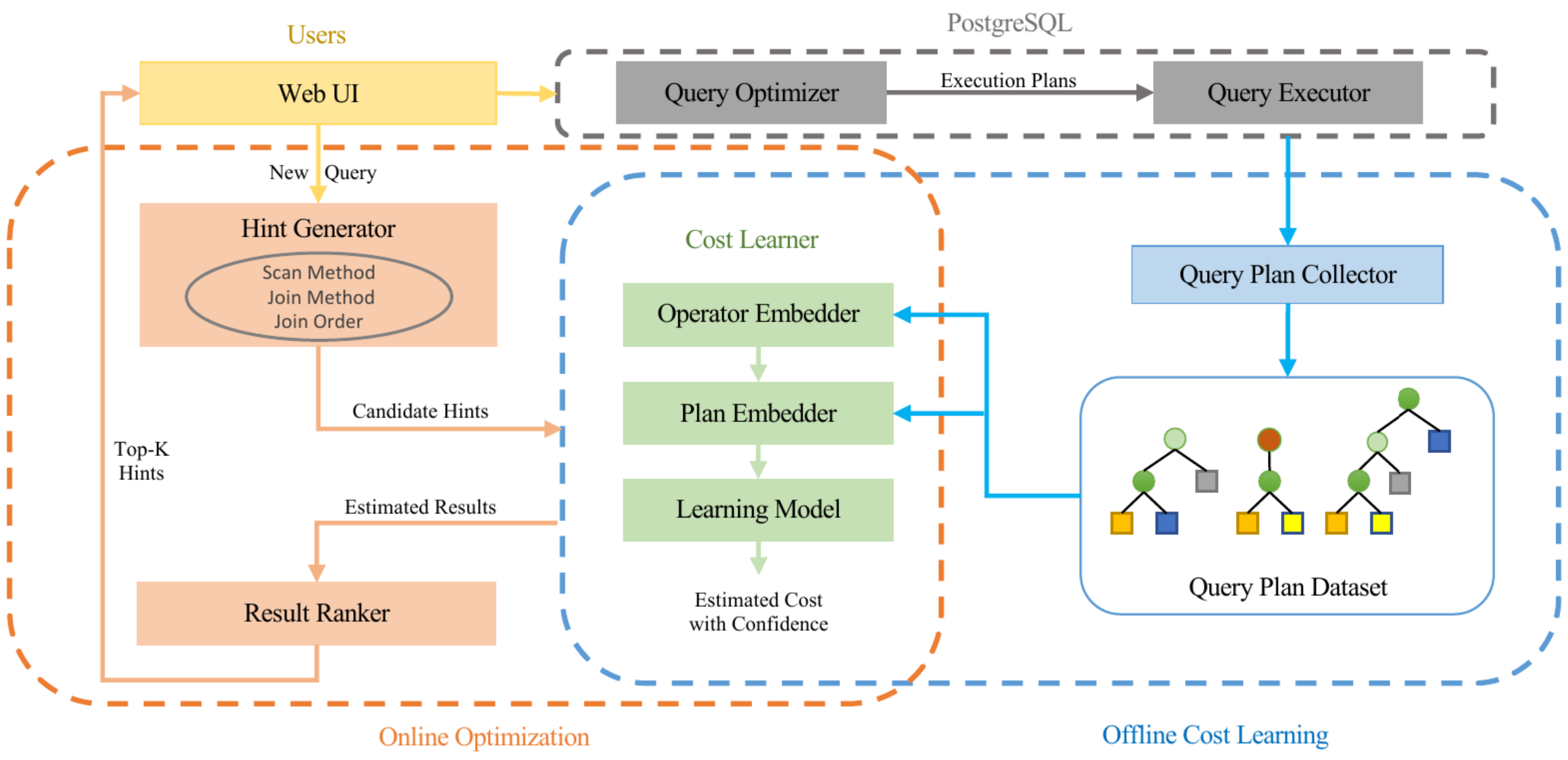
Q	PG	Bao worst	Bao best
q1	275.415884	12.206382	6.005776
q2	71.049927	198.310226	9.242487
q3	10.982070	290.048801	10.805816
q4	26.890862	26.966064	1.527303
q5	9.692364	9.354480	1.350012
q6	21.741243	19.851484	7.341236
q7	51.935738	51.321676	7.288905
q8	28.725613	15.981973	5.995388
q9	15.645138	16.394102	7.327004
q10	11.720967	9.688347	7.373339
q11	15.163100	7.686548	5.853226
q12	12.934380	9.379889	4.565600
q13	18.687008	11.803825	3.417922
q14	11.100027	14.864732	7.060695
q15	9.641760	8.258874	4.153027
q16	5.312640	7.992982	1.221813
q17	6.404161	17.702658	5.868285
q18	11.912653	20.336241	6.772051
q19	9.943220	33.939818	10.330661
q20	0.143906	0.679753	0.344254
q21	1.022706	1.292618	0.921263
q22	16.113360	51.231996	8.196555

Deep A Learned Query Optimizer

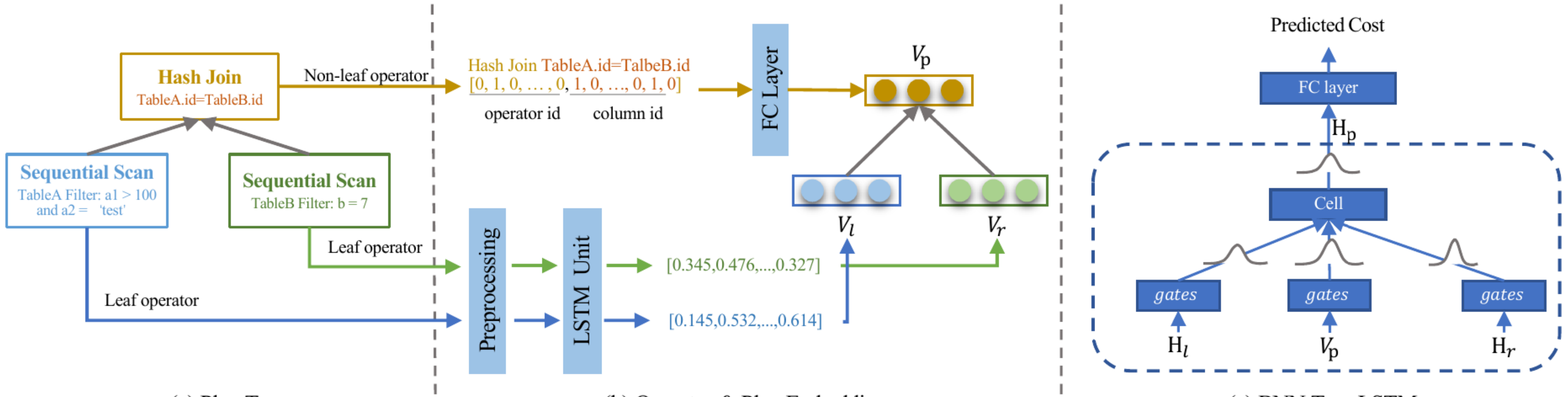
Luming Sun, Tao Ji, Cuiping Li, Hong Chen

SIGMOD 2022 Demo Track

System Architecture



Cost Learner Intervals



(a) Plan Tree (b) Operator & Plan Embedding (c) BNN Tree-LSTM

Contributions

- Practical
  - DeepO is integrated with real DBMS (PostgreSQL).
- Novel
  - DeepO adopts BNN-based Tree-LSTM network.
  - DeepO offers fine-grained query optimization.
- User-friendly
  - DeepO has a web UI for interactive operation.
- Effective
  - DeepO can optimize queries.

Takeaways

- Reinforcement learning is suitable for the task.
- Join order optimization (query plan optimization) is a very difficult task.
- Challenges and opportunities coexist, as do hardships and hopes.

Thanks!

sunluming@ruc.edu.cn
https://lumingsun.github.io/