A Sampling–based Learning Framework for Big Databases

Jingtian Zhang, Sai Wu, Junbo Zhao, Zhongle Xie, Feifei Li, Yusong Gao, Gang Chen

Alibaba / Zhejiang University (ZJU)
Autonomous Database

• To get rid of the work from DBAs.
• Tune the database performance fully automatically.
• RL (Reinforcement learning) is a promising direction to go.
• Core challenge: the sample complexity (AlphaGo)
Reinforcement Learning & Large-scale Databases

• Too expensive → How to reduce the overhead?

(a) 10-million database

(b) 1-billion database
Questions at core of this paper

- What is the difference between the neural model trained in the sampled database and original database?
- How can we guarantee that the model trained in the sampled database is well-adopted to the original database?
- If model transfer incurs a high precision loss, how can we address the problem?
Motivation
Sampling

• How to reduce the overhead?
  → What if train the policy network on a sampled database
  → How do we do the sampling: towards mitigating un-ignorable noises and prediction drifting

• How to adapt the model to the original database
  → Transfer the model to the original database
A Transferable Sampling-based Framework

1. Train a model on an unbiased sampled database
2. Transfer the trained model to the original database
Method
Taking the index recommendation task as an example
Initial Training Phase -- Q function

- A workload feature network.
  -> consider select, join and etc.
- An index feature network.
- A fusion network merging the representations.
Continuous Learning Phase

- Step 1: Locate the outliers for the transferred model
- Step 2: Tune the model with a new branch
Lower and Upper Bound --- Robustness (noises)

Suppose the correct class for an input $\tilde{x}$ on $D'$ is $i^*$ ($i^* = 0$ or $1$), we define the margin function for a neural model $f(x)$ as:

$$g(x) = f_{i^*}(x) - f_i(x)$$

$$\Pr(g(x) \geq 0) \geq 1 - \epsilon$$

**THEOREM 1.** Let $f(x)$ be a K-class neural classifier and $x_0$ is its input. We define the noise $\delta$ as $||x - x_0||_p \leq \delta$ for $p \geq 1$. Let $g(x)$ be the margin function. Suppose the input vector $X$ follows some given distribution $\mathcal{D}$ with mean $x_0$. For a constant $a \geq 0$, there exists a lower bound $\mathcal{L}$ and upper bound $\mathcal{U}$ for the probability $\mathcal{L} \leq \Pr(g(x) \geq a) \leq \mathcal{U}$, where

$$\mathcal{L} = 1 - F_{g^L}(x)(a)$$

and

$$\mathcal{U} = 1 - F_{g^U}(x)(a)$$

$F_Z(z)$ is the cumulative distribution function (CDF) of the random variable $Z$.

$$g^L(x) = A^L x + b^L$$

$$g^U(x) = A^U x + b^U$$

Note that if $X$ follows a normal distribution with a mean $\mu$ and variance $\Sigma$, the linear combination $Z = wX + v$ also follows the normal distribution with a mean $\mu_z = w\mu + v$ and variance $\sigma_z = w\Sigma w^T$. The CDF of $Z$ can be estimated as:

$$\frac{1}{2}(1 + \text{erf}(\frac{Z - \mu_z}{\sigma_z \sqrt{2}}))$$

where $\text{erf}$ represents the Gauss error function defined as:

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

Merging the definitions for CDF of $Z$, $\mu_L$, $\mu_U$, $\sigma_L$ and $\sigma_U$. We have

$$\mathcal{L} \approx \frac{1}{2} - \frac{1}{2} \text{erf}(\frac{a - \mu_L}{\sigma_L \sqrt{2}})$$

$$\mathcal{U} \approx \frac{1}{2} - \frac{1}{2} \text{erf}(\frac{a - \mu_U}{\sigma_U \sqrt{2}})$$

$$\delta = \arg\max_{i=0}^m \left| \sum_j \log \left( \frac{1}{\text{Sel}(Q_j, C_i)} \right)^2 - \sum_j \log \left( \frac{1}{\text{Sel'}(Q_j, C_i)} \right)^2 \right|$$
Normalization of Rewards — Robustness

• To guarantee the model returns the consistent prediction results for both the original database and any sampled database.

\[ r_{\text{norm}}(q_i, s_i) = \frac{R(q_i, 1)}{R(q_i, \rho_i)} \times RT(q_i, I) \]
Experiments
Setting

- Benchmark
  TPC–H benchmark, JOBbenchmark
- Server
  Intel Xeon Processor E5 2660 v2 (25M Cache, 2.20 GHz)
- Database
  PolarDB, Intel Xeon Platinum 8163 CPU (25M Cache, 2.50GHz)
- GPU
  V100

- Notation
  Default: on the whole database
  Percona: a RL baseline
  Lift: a RL baseline
  Random: random sampled
  IR–Mirror–NR:
    - reward normalization
    - continuous learning techniques
  IR–OR: no mirror
  IR–Mirror: without the optimizations
  Mirror–CL: all the optimizations
Main Result

<table>
<thead>
<tr>
<th>Data Size</th>
<th>IR-Mirror-NR (per V100)</th>
<th>Lift (mins per V100)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 Million (0.8%)</td>
<td>41 mins</td>
<td>77 mins</td>
</tr>
<tr>
<td>16 Million (1.6%)</td>
<td>119 mins</td>
<td>186 mins</td>
</tr>
<tr>
<td>32 Million (3.2%)</td>
<td>314 mins</td>
<td>386 mins</td>
</tr>
<tr>
<td>1 Billion (100%)</td>
<td>45 days</td>
<td>Not Converge</td>
</tr>
</tbody>
</table>

Table 1: Training Time with Varied Sampling Rate

(a) Overall Performance

(b) Index number (Multi-column)

(c) Index number (Multi-table Join)
Conclusion
Contribution

- Mirror reduces the training overhead by transferring a trained policy network.

- Based on the theoretic bounds, Mirror adopts a continuous learning technique to refine the model on the original database.

- Experiments demonstrate promising results.
Thanks! & QA