**Background**

**Goal:** Recommend a configuration to achieve the best user objective.

- **Objective Model:**
  - Configuration $\theta^j$: $\theta^j \in \Theta$ for workload $j$.
  - Workload $j$:
    - System hardware (fixed)
    - User objective: latency
  - **Neural Network Regressor (NNR):**
    - [Input] Configuration: easy to represent
    - [Input] Workload: hard to represent
    - [Input] System hardware: fixed, thus ignored in this problem.
    - [Output] Latency: easy to represent

- **Approach:**
  - **Step 1: Use Autoencoder (AE) to encode workload**
    - For a workload $j$ runs in configuration $i$, we collected its trace observation $O^j_i$.
    - Use Observation traces as the input layer of the AE model.
    - The bottleneck layer in the AE model can be the representation of the combination of the workload and configuration.
    - Split the bottleneck layer into $E^j_i$ and $\tilde{E}^j_i$, representing workload and configuration respectively.
    - For a fixed job $j$ with different configuration $i$, the bottleneck layer in the AE model should represent the same workload encoding $W$ and different configuration information.

  - **Step 2: Use Neural Network Regressor to model the user objective**
    - Use the workflow encoding from Step 1 to finish the objective regressor model.

**Overview of tuning**

- **Configuration Search:**
  - **Objective:** $\theta^j$, configuration $i$.

**Experiment**

- **Data sets:**
  - Machine Learning Workloads (~384 "data points")
    - Doing binary classification tasks by using Stochastic Gradient Descent.
    - Totally 12 online workloads (32 configurations per workload)
  - SQL-like workloads (~4400 "data points")
    - Click stream analysis tasks generated by 5 parameterized templates.
    - 6 offline intensive workloads (with an average of 488 configurations)
    - 46 regular online workloads (with an average of 32 configurations)

**Evaluation**

- Using Mean Absolute Percentage Error (MAPE) to measure the accuracy of the predicted latency.

\[
MAPE = \frac{100}{m} \sum_{i \in I} \left| \frac{\hat{y}_i - y_i}{y_i} \right|
\]

where $\hat{y}_i$ denotes the true value of latency, $y_i$ denotes the predicted latency under workload $j$, configuration $i$.

- **T1 error**
  - Use 20% of the current workload observation traces + other workload observation traces as training data to compute MAPE.
  - Show the generalization power over unseen configuration of familiar workloads.

- **T2 error** (unseen workloads)
  - Only use other workload observation traces as training data to compute MAPE.
  - Show the generalization power over unseen workloads.

**Current Result and Future Work**

<table>
<thead>
<tr>
<th>SQL-like workload</th>
<th>Machine Learning Workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1 error</td>
<td>T2 error</td>
</tr>
<tr>
<td>T1 error</td>
<td>T2 error</td>
</tr>
</tbody>
</table>

**Future work**

- Try to train the AE model better by training for configuration in a supervision manner first, then refining tuning the pretrained model by the overall loss function.
- Borrow the information from "similar workload encoding" for a new coming workload.
- Refine the loss function to make sure two similar workload can have similar observation traces under each same configuration.

---

**Tuning Data Analytics Systems Through Deep Learning**

Chenghao Lyu
chenghao@cs.umass.edu