Introduction
We are entering an era of data engineering, and machine learning algorithms are widely used and adopted. This brings new opportunities as well as new challenges to software engineering research:
- The output of a machine learning program is a model. A model cannot be simply recognize as correct or incorrect. Instead of debugging a model, we can only optimize a model.
- The performance of the model is related to all components in the data engineering process.
  - Implementation bugs, data bugs, data collection and cleaning, constant numbers, model selection, underfitting, overfitting etc.
- Dependency tracking is very expensive and inaccurate.
  - Most machine learning algorithms are highly iterative, especially for graph models.
  - Dependencies should be properly measured to fully understand the machine learning process.

<table>
<thead>
<tr>
<th>Graph</th>
<th>#Node</th>
<th>#Edge</th>
<th>#Dep avg</th>
<th>#Dep max</th>
<th>CPU</th>
<th>Mem</th>
</tr>
</thead>
</table>
| Stanford  | 1,000 | 10,948| 723      | 1,000    | 825% | 33.25%
| Google    | 34,546| 421,578| 6,194    | 21,349   | 1103%| 42.19%
| TencentUser | 18,970| 170,327| 1,293    | 8,386    | 1023%| 48.24%
| TencentMsg | 83,360| 2,516,122| 12,342   | 82,194   | 1345%| 39.25%
| Twitter   | 96,401| 482,834| 32,593   | 72,294   | 1483%| 29.38%

**LAMP**
- A data provenance system for machine learning programs.
  - Currently, we focus on graph models.
- LMAP uses the Jacobian matrix as the provenance.
  - Partial derivatives are used to measure the sensitiveness of an output variable to an input variable.
  - The value is ignored if it is too small. This keeps the highly weighted dependencies and reduces unimportant dependencies.
- LAMP is inspired by automatic differentiation (AD).
  - Beyond AD, we handle control flows.
- LAMP is faster and more accurate comparing with traditional dependency tracking (for machine learning algorithms).
  - Only important dependencies are kept.
  - Dependencies are measured by partial derivatives.

Collect runtime information for derivative computation, and compute the Jacobian matrix when needed.
- Runtime information, e.g. the Product Rule.
- Computing the Jacobian matrix needs a second run. This avoids unnecessary computation.
- Partial derivatives can be commuted by using the Chain Rule.
- The whole program is viewed as a sequences of computations.
- When the program potentially can switch branches, we compute the derivatives by using the raw definition (with a new thread).

\[ m = \frac{\text{change in } y}{\text{change in } x} = \frac{\Delta y}{\Delta x} \]

**Evaluation**
- 12 algorithms with 100+ Python implementations, 35 datasets,
  - Github, MovieLens, Stanford Large Network Dataset Collection
- Runtime overhead
  - Original run with logging: <1%
  - Derivative computation: >100%
- Memory Consumption
  - The size of the Jacobian matrix (unoptimized), dense matrix
  - <1% of the Jacobian matrix size (optimized), sparse matrix
  - Grows fast at the beginning, and keeps the same after a short period
- Logging overhead
  - <7 KB for all cases
- Comparing with tainting, average numbers
  - Runtime overhead: 1133.17% vs. 150.16%
  - Space overhead: 41.16% vs. 19.71%
- Optimization
  - 99.8% partial derivative values are smaller than 10⁻⁵

**Case Study**
- PageRank
  - The Tencent Weibo network
    - Accounts, messages, likes etc.
- Rank accounts
  - Popular accounts, e.g. Kai-Fu Lee
    - Followed by other high-rank accounts, and a lot of followers
- Unexpected high-rank account: Yangqi
  - No popular followers, a relatively small number of followers
  - Highly affected by another account: Qing Fei (not high-rank)
- Edges with negative values
  \[ PR(u) = 1 - \frac{d - \sum_{j \in BU} PR(j) * K_j}{|V|} \]

**Model optimization**
- 6 different models
- Improve the accuracy from 80% to 90%+
- Identify problems in data cleaning, constant numbers etc.
- Data bugs
  - 12 data bugs (e.g. missing, incomplete, wrong values etc.)
  - Identify the wrong values and their propagations
- Implementation bugs
  - 35 bugs from Github, StackOverflow including unknown bugs

**Contacts**
Email: ma229@purdue.edu