Building a Platform for Efficient Collaborative Research

Percy Liang

Microsoft Faculty Summit
July 16, 2013

Objective

To create a collaborative ecosystem for conducting computational research in an efficient and reproducible manner.

Lack of reuse

The current research process

Step 1: come up with a good idea

Step 2:
- Find data, clean it, convert between formats
- Find code, compile it, email authors, reimplement
- Run experiments, keep track of multiple versions

Non-exhaustive comparisons

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Previous method</th>
<th>Our method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1</td>
<td>88% accuracy</td>
<td>92% accuracy</td>
</tr>
<tr>
<td>Dataset 2</td>
<td>72% accuracy</td>
<td>77% accuracy</td>
</tr>
<tr>
<td>Dataset 3</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Dataset 4</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Dataset 5</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Dataset 6</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>...</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Uncontrolled comparisons

<table>
<thead>
<tr>
<th>Previous method</th>
<th>Our method</th>
</tr>
</thead>
<tbody>
<tr>
<td>88% accuracy</td>
<td>92% accuracy</td>
</tr>
<tr>
<td>using sampling</td>
<td>using optimization</td>
</tr>
<tr>
<td>$L_2$ regularization</td>
<td>$L_1$ regularization</td>
</tr>
<tr>
<td>5-fold cross-validation</td>
<td>10-fold cross-validation</td>
</tr>
<tr>
<td>one set of bugs</td>
<td>another set of bugs</td>
</tr>
</tbody>
</table>
Lack of good broad overview

Question: Which algorithms work well on what types of datasets?

An outsider's perspective

Difficult to understand the problems:
- classification
- regression
- ranking
- structured prediction
- statistical relational learning
- ...

Difficult to find reliable solutions:
- logistic regression
- kernel methods
- topic models
- conditional random fields
- hidden Markov models
- ...

Outline

MLcomp: code, data, comparison

CodaLab: complex workflows

A meeting place

People with programs:

How well does my method work compared to others?

People with datasets:

What is the best method for my problem?

Components

Programs:
- SVMlight - C implementation of support vector machines for classification by Thorsten Joachims.

Datasets:
- thyroid - Task is to predict whether a patient has thyroid disease given attributes (age, gender, I131 treatment, etc.)

Runs:
- Program: SVMlight
- Dataset: thyroid
- Error: 2.6%
- Time: 1 second
Usage

- Users upload **programs**
- Users upload **datasets**
- System runs programs on datasets

Everything happens in a distributed/asynchronous manner.

---

Generalization

\[
\begin{array}{ccccccc}
D & P & D & D & D & P & D & D
\end{array}
\]

**Evaluation:**

- Program $P$ run on dataset $D$, get some accuracy
- Only meaningful if $D$ is **independent** of $P$
- In papers, this is never true!

**In MLcomp:**

- People upload program $P$
- People upload new dataset $D$ **afterwards**
- Guarantees that $P$ is not overfit to $D$

---

Result

### Datasets

<table>
<thead>
<tr>
<th>Programs</th>
<th>Datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.6</td>
<td>6.2</td>
</tr>
<tr>
<td>1.1</td>
<td>1.4</td>
</tr>
<tr>
<td>5.5</td>
<td>4.5</td>
</tr>
<tr>
<td>0.4</td>
<td>1.5</td>
</tr>
<tr>
<td>9.0</td>
<td>8.9</td>
</tr>
<tr>
<td>3.9</td>
<td>5.2</td>
</tr>
<tr>
<td>1.5</td>
<td>7.4</td>
</tr>
<tr>
<td>3.3</td>
<td>9.5</td>
</tr>
</tbody>
</table>

---

Design decisions

- Program is an arbitrary Linux binary (support C++, Java, Python, R, etc.)
- User-uploaded programs and datasets conform to standard interfaces/formats
- All runs executed on Amazon EC2 (initiated by user or system)
- Users can download any programs/datasets/runs (not marked by user as restricted)

---

Related projects

- **Code repositories (mloss.org):** only code
- **Data repositories (UCI, mldata.org):** only data
- **Machine learning as a service (BigML, Google Prediction API):** provide fixed set of programs, people submit (private) data; doesn't encourage development of new methods
- **Competitions (Kaggle):** provide fixed set of datasets, people submit predictions; doesn't promote general/clean solutions

---

Status

- Development started in 2009 [with Jake Abernethy, Alex Simma, Ariel Kleiner]
- Today: 2129 users, 686 datasets, 390 programs, 19083 runs
- Website: mlcomp.org
- Open-source on GitHub: https://github.com/percyliang/mlcomp
Delving deeper

MLcomp's primitive:

\[ \text{accuracy} = \text{run(program, dataset)} \]

Want fuller analysis:
- Hyperparameter tuning / sensitivity analysis
- Learning curves (varying amounts of data)
- Error analysis: ROC, confusion matrices, predictions
- Visualization: plot all these statistics

Text understanding

Peter Houser, a 68-year-old man with a history of hypertension, has an aortic aneurysm repaired. On the first postoperative day, he is transferred from the intensive care unit (ICU) to the medical-surgical unit. Mr. Houser has a midline incision, a nasogastric tube connected to low intermittent suction, and a left subclavian triple lumen catheter. On receiving Mr. Houser from the ICU, the nurse notes that he has edema of both lower extremities, and his pedal pulses are not palpable in either foot. Which of these actions should the nurse take?

System

Use a hand-held Doppler ultrasound device to reassess his pulses.

Stages of an NLP workflow

- tokenization
- part-of-speech tagging
- named-entity recognition
- syntactic parsing
- coreference resolution
- relation extraction
- clustering
- gazetteer
- Google n-grams
- Wikipedia
- raw text
- ...
**Principle 1: modularity**
AI problems require efforts of entire community
People specialize, contribute in decentralized way

**Intermediate tasks**
- **Old**: use intermediate metrics, rhetoric
- **New**: plug in and see ramifications automatically

**A collaborative ecosystem**
Modules interoperate via standard interfaces (Internet)

**Individual benefits:**
- Avoid duplicate work
- Publicize tools/datasets

**Community benefits:**
- Serial combination of components
- Parallel ensembles for better predictions (Netflix Prize)

**Principle 2: immutability**

**Inspiration**: Git version control system

- All programs/datasets/runs are write-once
- Enable collaboration without chaos
- Capture the research process in a reproducible way

**CodaLab worksheet**

*We now train the classifier with more data.*

<table>
<thead>
<tr>
<th>Program</th>
<th>SVMlight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments</td>
<td>-n 2000</td>
</tr>
<tr>
<td>Dataset</td>
<td>thyroid</td>
</tr>
<tr>
<td>Error</td>
<td>2.6%</td>
</tr>
<tr>
<td>Time</td>
<td>1 second</td>
</tr>
</tbody>
</table>

Notice that the error remains the same, suggesting that we've saturated our model.

**Use cases:**
- Informal blog posts
- Formal executable papers

**Principle 3: literacy**
MLcomp is about truth; what about interpretation?

**Inspiration:**
- Mathematica notebook, IPython notebook: interleave code with text descriptions
Related projects

- runmycode.org, myexperiment.org, Weka require specific formats
- Matlab/R/Perl/Python/Ruby provide code modules, but no data; data is a resource

Challenges

Inertia:
- **Problem**: People have personal setup, takes effort to port to foreign environment
- **Solution**: Easy to contribute, benefits of online sharing (Dropbox + execution)

Search:
- **Problem**: CodaLab is general repository, how to search?
- **Solution**: Smart autocomplete, ranking, recommendation

Status

Collaboration with Microsoft Research Connections
Project hosted by the Outercurve Foundation

People:
- Development: Christophe Poulain, Beau Hargis, Justin Carden, Dan O’Donnell
- Program/community: Evelyne Viegas, Erick Watson, Lori Ada Kilty, Ivan Judson, Simon Mercer
- Design: Chrisopher Rampey

Final remarks

To create a **collaborative** ecosystem for conducting computational research in an **efficient** and **reproducible** manner.

Questions/feedback?