Special Topics: CSci 8980
Machine Learning in Computer Systems

Jon B. Weissman (jon@cs.umn.edu)

Department of Computer Science
University of Minnesota
Introduction

• Introductions - all

• Who are you?

• What interests you and why are you here?
Introduction (cont’d)

• What is this course about?

  – machine learning
    • Interpreted broadly: learning from data to improve ...

  – computer systems
    • Interpreted broadly: compilers, databases, networks, OS, mobile, security, ... (not finding a boat in an image)
Confession

• If you took a ML course, you know more than me about it

• Interestingly ...
  – Took an AI course from Geoff Hinton
  – Did an M.S. on neural networks eons ago
Web Site

Technical Course Goals

• Learn a “little” about ML and DL techniques
  – Understand their scope of applicability

• Learn about one or more areas of computer systems in more detail

• Learn how ML/DL can benefit computer systems
Non-Technical Course Goals

• Learn how to write critiques (blogs)
• Learn how to present papers and lead discussions
• Do a team research project
  – Idea formation
  – Writeup
  – Experiment
  – Present
  – (fingers-crossed) publish a (workshop) paper
Major Topics

- Machine learning Introduction
- Databases
- Networking
- Scheduling
- Power management
- Storage
- Compilers/Architecture
- Fault tolerance
- IOT/mobile
Course structure

• Grading ...
  – Presentations: 2 (1 big, 1 small) of them (10% each)
  – Take-home mid-term: 20%
  – Final project: 30%
  – Written critiques (blogging): 10%
    • Approximately 2 of these per person
  – Discussions: 20%
Presentations

• Two presentations
  – Presentation = 1 long paper; 1 short paper
• Give paper’s context and background
• Key technical ideas
  – Briefly explain the ML technique used
• It’s relation to other papers or ideas
• Positive/Negative points (and why)
• long: 30 minutes max to leave time for discussion
• short: 15 minutes
• Keep it interesting!
  – tough job: don’t want gory paper details nor total fluff
  – audience: smart CS/EE students and faculty
Presentations (cont’d)

• Research/Discussion questions
  – go beyond the claims in the paper
  – limitations, extensions, improvements
  – “bring up” any blog discussions

• You may find .ppt online BUT
  – put it in your own words
  – understand everything you are presenting
Critiques/Blogging

• Brief overview
• Positives and negatives
  – Hint: only one of these will be in the abstract 😊
• Discussion points
• Due before paper is presented so presenter has a chance to see it
Projects

• Talk about ideas in a few weeks ...  
  – present a list of things that are useful, open to other ideas
• Work in a team of 2 or 3
• Large groups are fine  
  – Plan C could be an issue
• Risk encouraged ... and rewarded (even if you fall short)
Projects (cont’d)

• Implementation project
  – Applying ML technique(s) to any systems area

• 1 page proposals will be due in early March

• Will present final results at the end
Near-term Schedule

• [web site](#)

• Next three lectures+
  – I will present, no blogging necessary

• Need volunteers for upcoming papers (see ? next to papers on the website)
  – I will hand-pick “volunteers” if necessary 😊
  – I will pick bloggers
Admin Questions?
Inspiration

• Jeff Dean’s NIPS 2017 keynote
Next two lectures

• Basics of ML/DL
  – See website for reading
Machine Learning for Systems and Systems for Machine Learning

Jeff Dean
Google Brain team

Presenting the work of many people at Google
Machine Learning for Systems
Learning Should Be Used Throughout our Computing Systems

Traditional low-level systems code (operating systems, compilers, storage systems) does not make extensive use of machine learning today

This should change!

A few examples and some opportunities...
Machine Learning for Higher Performance Machine Learning Models
For large models, model parallelism is important
For large models, model parallelism is important.

But getting good performance given multiple computing devices is non-trivial and non-obvious.
Reinforcement Learning for Higher Performance Machine Learning Models

Device Placement Optimization with Reinforcement Learning,
Azalia Mirhoseini, Hieu Pham, Quoc Le, Mohammad Norouzi, Samy Bengio, Benoit Steiner, Yuefeng Zhou, Naveen Kumar, Rasmus Larsen, and Jeff Dean, ICML 2017, arxiv.org/abs/1706.04972
Reinforcement Learning for Higher Performance Machine Learning Models

Placement model (trained via RL) gets graph as input + set of devices, outputs device placement for each graph node.

Device Placement Optimization with Reinforcement Learning, Azalia Mirhoseini, Hieu Pham, Quoc Le, Mohammad Norouzi, Samy Bengio, Benoit Steiner, Yuefeng Zhou, Naveen Kumar, Rasmus Larsen, and Jeff Dean, ICML 2017, arxiv.org/abs/1706.04972
Reinforcement Learning for Higher Performance Machine Learning Models

Placement model (trained via RL) gets graph as input + set of devices, outputs device placement for each graph node.

Measured time per step gives RL reward signal.

Device Placement Optimization with Reinforcement Learning, Azalia Mirhoseini, Hieu Pham, Quoc Le, Mohammad Norouzi, Samy Bengio, Benoit Steiner, Yuefeng Zhou, Naveen Kumar, Rasmus Larsen, and Jeff Dean, ICML 2017, arxiv.org/abs/1706.04972
Device Placement with Reinforcement Learning

Placement model (trained via RL) gets graph as input + set of devices, outputs device placement for each graph node

Measured time per step gives RL reward signal

+19.3% faster vs. expert human for neural translation model

+19.7% faster vs. expert human for InceptionV3 image model

Device Placement Optimization with Reinforcement Learning,
Azalia Mirhoseini, Hieu Pham, Quoc Le, Mohammad Norouzi, Samy Bengio, Benoit Steiner, Yuefeng Zhou, Naveen Kumar, Rasmus Larsen, and Jeff Dean, ICML 2017, arxiv.org/abs/1706.04972
Device Placement with Reinforcement Learning

Placement model (trained via RL) gets graph as input + set of devices, outputs device placement for each graph node.

Measured time per step gives RL reward signal.

Plug: Come see Azalia Mirhoseini’s talk on “Learning Device Placement” tomorrow at 1:30 PM in the Deep Learning at Supercomputing Scale workshop in 101B.

+19.3% faster vs. expert human for neural translation model

+19.7% faster vs. expert human for InceptionV3 image model

Device Placement Optimization with Reinforcement Learning,
Azalia Mirhoseini, Hieu Pham, Quoc Le, Mohammad Norouzi, Samy Bengio, Benoit Steiner, Yuefeng Zhou, Naveen Kumar, Rasmus Larsen, and Jeff Dean, ICML 2017, arxiv.org/abs/1706.04972
Learned Index Structures
not
Conventional Index Structures
B-Trees are Models

(a) B-Tree Index

Key

BTree

pos

pos - 0 pos + pagezise

... ...

(b) Learned Index

Key

Model (e.g., NN)

pos

pos - min_err pos + max_err

... ...

The Case for Learned Index Structures, Tim Kraska, Alex Beutel, Ed Chi, Jeffrey Dean & Neoklis Polyzotis, arxiv.org/abs/1712.01208
Indices as CDFs

The Case for Learned Index Structures, TimKraska, Alex Beutel, Ed Chi, Jeffrey Dean & Neoklis Polyzotis, arxiv.org/abs/1712.01208
## Does it Work?

Index of 200M web service log records

<table>
<thead>
<tr>
<th>Type</th>
<th>Config</th>
<th>Lookup time</th>
<th>Speedup vs. Btree</th>
<th>Size (MB)</th>
<th>Size vs. Btree</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTree</td>
<td>page size: 128</td>
<td>260 ns</td>
<td>1.0X</td>
<td>12.98 MB</td>
<td>1.0X</td>
</tr>
<tr>
<td>Learned index</td>
<td>2nd stage size: 10000</td>
<td>222 ns</td>
<td>1.17X</td>
<td>0.15 MB</td>
<td>0.01X</td>
</tr>
<tr>
<td>Learned index</td>
<td>2nd stage size: 50000</td>
<td>162 ns</td>
<td>1.60X</td>
<td>0.76 MB</td>
<td>0.05X</td>
</tr>
<tr>
<td>Learned index</td>
<td>2nd stage size: 100000</td>
<td>144 ns</td>
<td>1.67X</td>
<td>1.53 MB</td>
<td>0.12X</td>
</tr>
<tr>
<td>Learned index</td>
<td>2nd stage size: 200000</td>
<td>126 ns</td>
<td>2.06X</td>
<td>3.05 MB</td>
<td>0.23X</td>
</tr>
</tbody>
</table>

Hash Tables

(a) Traditional Hash-Map

(b) Learned Hash-Map

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Slots</th>
<th>Hash Type</th>
<th>Search Time (ns)</th>
<th>Empty Slots</th>
<th>Space Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>75%</td>
<td>Model Hash</td>
<td>67</td>
<td>0.63GB (05%)</td>
<td>-20%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Random Hash</td>
<td>52</td>
<td>0.80GB (25%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>Model Hash</td>
<td>53</td>
<td>1.10GB (08%)</td>
<td>-27%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Random Hash</td>
<td>48</td>
<td>1.50GB (35%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>125%</td>
<td>Model Hash</td>
<td>64</td>
<td>2.16GB (26%)</td>
<td>-6%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Random Hash</td>
<td>49</td>
<td>2.31GB (43%)</td>
<td></td>
</tr>
<tr>
<td>Web Log</td>
<td>75%</td>
<td>Model Hash</td>
<td>78</td>
<td>0.18GB (19%)</td>
<td>-78%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Random Hash</td>
<td>53</td>
<td>0.84GB (25%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>Model Hash</td>
<td>63</td>
<td>0.35GB (25%)</td>
<td>-78%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Random Hash</td>
<td>50</td>
<td>1.58GB (35%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>125%</td>
<td>Model Hash</td>
<td>77</td>
<td>1.47GB (40%)</td>
<td>-39%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Random Hash</td>
<td>50</td>
<td>2.43GB (43%)</td>
<td></td>
</tr>
<tr>
<td>Log Normal</td>
<td>75%</td>
<td>Model Hash</td>
<td>79</td>
<td>0.63GB (20%)</td>
<td>-22%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Random Hash</td>
<td>52</td>
<td>0.80GB (25%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>Model Hash</td>
<td>66</td>
<td>1.10GB (26%)</td>
<td>-30%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Random Hash</td>
<td>46</td>
<td>1.50GB (35%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>125%</td>
<td>Model Hash</td>
<td>77</td>
<td>2.16GB (41%)</td>
<td>-9%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Random Hash</td>
<td>46</td>
<td>2.31GB (44%)</td>
<td></td>
</tr>
</tbody>
</table>

The Case for Learned Index Structures, Tim Kraska, Alex Beutel, Ed Chi, Jeffrey Dean & Neoklis Polyzotis, arxiv.org/abs/1712.01208
Bloom Filters

Model is simple RNN
$W$ is number of units in RNN layer
$E$ is width of character embedding

$\sim 2X$ space improvement over Bloom Filter at same false positive rate

The Case for Learned Index Structures, Tim Kraska, Alex Beutel, Ed Chi, Jeffrey Dean & Neoklis Polyzotis, arxiv.org/abs/1712.01208
Machine Learning for Improving Datacenter Efficiency
Machine Learning to Reduce Cooling Cost in Datacenters

Collaboration between DeepMind and Google Datacenter operations teams.
See https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/
Where Else Could We Use Learning?
Computer Systems are Filled With Heuristics

Compilers, Networking code, Operating Systems, ...

Heuristics have to work well “in general case”

Generally don’t adapt to actual pattern of usage

Generally don’t take into account available context
Anywhere We’re Using Heuristics To Make a Decision!

**Compilers**: instruction scheduling, register allocation, loop nest parallelization strategies, ...

**Networking**: TCP window size decisions, backoff for retransmits, data compression, ...

**Operating systems**: process scheduling, buffer cache insertion/replacement, file system prefetching, ...

**Job scheduling systems**: which tasks/VMs to co-locate on same machine, which tasks to pre-empt, ...

**ASIC design**: physical circuit layout, test case selection, ...
Anywhere We’ve Punted to a User-Tunable Performance Option!

Many programs have huge numbers of tunable command-line flags, usually not changed from their defaults

--eventmanager_threads=16
--bigtable_scheduler_batch_size=8
--mapreduce_merge_memory=134217728
--lexicon_cache_size=1048576
--storage_server_rpc_freelist_size=128
...
Meta-learn everything

ML:

- learning placement decisions
- learning fast kernel implementations
- learning optimization update rules
- learning input preprocessing pipeline steps
- learning activation functions
- learning model architectures for specific device types, or that are fast for inference on mobile device X, learning which pre-trained components to reuse, ...

Computer architecture/datacenter networking design:

- learning best design properties by exploring design space automatically (via simulator)
Keys for Success in These Settings

(1) Having a numeric metric to measure and optimize
(2) Having a clean interface to easily integrate learning into all of these kinds of systems

Current work: exploring APIs and implementations
Basic ideas:
Make a sequence of choices in some context
Eventually get feedback about those choices
Make this all work with very low overhead, even in distributed settings
Support many implementations of core interfaces
Conclusions

ML hardware is at its infancy. Even faster systems and wider deployment will lead to many more breakthroughs across a wide range of domains.

Learning in the core of all of our computer systems will make them better/more adaptive. There are many opportunities for this.

More info about our work at g.co/brain