18-847F: Special Topics in Computer Systems

Foundations of Cloud and Machine Learning Infrastructure
Lecture 1: Overview and Logistics

Foundations of Cloud and Machine Learning Infrastructure
Course Enrollment and Waitlist

Capacity of the class is now increased to 35 (instead of 25)

Hopefully, a majority of waitlisted students will be cleared

Please contact Megan Oliver (moliver) regarding waitlists
Graduate Seminar Class

A Few Lectures

Concept-Check Homeworks

Class Presentations and Discussion

In-class Quizzes
Learning Objectives

- Know the state-of-the-art frameworks in cloud and machine learning and their theoretical foundations
- Read and understand latest research papers
- Present to an audience, and answer their questions
Why study Cloud and ML infrastructure?

What are the largest words after ‘Big Data’?
Big Data Gold Rush

Who got rich in the California gold rush?
Big Data Gold Rush

Who got rich in the California gold rush?

In the Big Data rush, it’s the infrastructure companies.
Topics Covered

Cloud Computing/Storage

Machine Learning

PARAMETER SERVER

\[ w' = w - \alpha \Delta w \]
Topics Covered

- Scheduling in Parallel Computing
- MapReduce, Straggler Replication
- Task Replication in Queueing Systems
- Erasure Coding for Locality/Repair
Topics Covered

- SGD and its convergence
- Asynchronous and Local-Update SGD
- Gradient and Model Compression
- Hyper-parameter tuning
Class Staff and Office Hours

Instructor: Gauri Joshi (gaurij)

Office Location: CIC 4105

Office Hours: Right after class or by appointment

TAs: Abhishek Sawarkar (asawarka), Jianyu Wang (jianyuw1)

Office Hours: TBD
Class Hours and Website(s)

- When: Mon, Wed 4:30-6:00 pm
- Where: Scaife Hall 222
- Class Website (Readings, Schedule):
  https://www.andrew.cmu.edu/course/18-847F/
- Gradescope for Homework submissions
Graduate Seminar Class

A Few Lectures

Concept-check Homeworks

Class Presentations and Discussion

In-class Quizzes
Lectures

- This Week: Probability and Linear Algebra Review

- Next week: Queueing and Scheduling

- Guest lectures during the semester by authors of papers relevant to this class
Graduate Seminar Class

A Few Lectures

Concept-check Homeworks

Class Presentations and Discussion

In-class Quizzes
Homeworks (45%)

- Concept-Check questions about the papers
- Programming questions
- Collaboration is okay but write your own answers, and list the collaborators
Reading Material

Papers will be posted on the class website
- Book chapters
- Survey papers
- Theory papers (Scheduling, Queuing, Coding, Optimization)
- Systems papers (Cloud, Machine Learning)

Additional reference books listed in the syllabus
Graduate Seminar Class

A Few Lectures

Concept-check Homeworks

Class Presentations and Discussion

In-class Quizzes
Class Presentations (10%)

- Sign up for presentation at least 2 weeks in advance
- Each student will present 1-2 times in the semester
- 25 min presentation, followed by 20 min discussion
  - Motivation and Related work
  - Summary of main results
  - Your views on the paper
Tentative Grading Rubric (Total: 10 pts)

- Motivation (2 pts)
- Clarity (2 pts)
- Correctness (3 pts)
- Engaging the audience (2 pts)
- Extra research, going beyond the paper (1 pts)

Sept 9th: Workshop on Effective Presentations

Presentations will be graded for following the guidelines given in the workshop
Class Participation (10%)

- Participation in Class Discussions
- Attendance and attention
- Insightful Questions/Comments
Graduate Seminar Class

A Few Lectures

Concept-check Homeworks

Class Presentations and Discussion

In-class Quizzes
In-class Quizzes (35%)

- 3 quizzes during the semester
- Checking your understanding of the material
- Can refer to the papers during the quiz
In Summary..

- Paper Reading
- Concept-Check Homeworks
- Class Presentations (1-2 in the semester)
- In-class Quizzes

You will learn..

- Latest cloud and ML infrastructure research
- How to read and critique research papers
- How to present effectively
- Some programming in TensorFlow/PyTorch
TO DO

- Sign-up for presentation
- Start reading the papers
Topics Covered

Cloud Computing/Storage

Machine Learning

PARAMETER SERVER

\[ w' = w - \alpha \Delta w \]

Model replica
Model replica
Model replica
Topics Covered

Cloud Computing/Storage

- Scheduling in Parallel Computing
- MapReduce, Straggler Replication
- Task Replication in Queueing Systems
- Erasure Coding for Locality/Repair
What is the cloud?

A collection of servers that can function as a single computing node, and can be accessed from multiple devices.
1960’s: The Mainframe Era

- Large, expensive machines
- Only one per university/institution

IBM 704 (1964)
1970’s: Virtualization

- IBM released a VM OS that allowed multiple users to share the mainframe computer

IBM 704 (1964)
1980’s-1990’s: Internet and PCs

- PCs become affordable
- Internet connectivity went on improving
- Virtual Private Networks (VPNs)
- Grid Computing: Connect cheap PCs via the Internet
- On the theory side, queueing theory, traditionally used in operations management rebounded
1990’s: Scheduling in Parallel Computing

- **Bin-Packing**
  - Need job size estimates

For references see survey [Weinberg 2008]
1990’s: Scheduling in Parallel Computing

- **Bin-Packing**
  - Need job size estimates

- **Processor Sharing**, i.e. switching b/w threads for different jobs
  - Need processor speed estimates

- **Load-balancing**: Work stealing, Power-of-choice
  - Need queue length estimates
1990’s: Internet and PCs

- PCs become affordable
- Internet connectivity went on improving
- Virtual Private Networks (VPNs)
- Grid Computing: Connect cheap PCs via the Internet
- Many Internet Companies bought their own servers and managed them privately
- But then the Dotcom bubble burst..
2000’s: The Cloud Computing Era

- The idea of a flexible, low-cost, scalable, shared computing environment developed

- Computing became a utility, like electricity
2000’s: The Cloud Computing Era

KEY ISSUE: Job sizes, server speeds & queue lengths are unpredictable

REASON: Large-scale resource sharing □ Variability in service
  • Virtualization, server outages etc.
  • Norm and not an exception [Dean-Barroso 2013]
The Tale of Tails

Tail at Scale: 99%ile latency can be much higher than average
The Tale of Tails

Tail at Scale: 99%ile latency much higher than average
Tale of Tails: Quiz

A server finishes a task in 1 sec with probability 0.9, and 10 sec with probability 0.1

- What is the expected task execution time?

- If 100 tasks are run in parallel of 100 servers, what is the expected time to complete all of them.
A server finishes a task in 1 sec with probability 0.9, and 10 sec with probability 0.1

- What is the expected task execution time? 
  \[ 1 \times 0.9 + 10 \times 0.1 = 1.9 \]

- If 100 tasks are run in parallel of 100 servers, what is the expected time to complete all of them.
Tale of Tails: Quiz

A server finishes a task in 1 sec with probability 0.9, and 10 sec with probability 0.1

- What is the expected task execution time?
  \[1 \times 0.9 + 10 \times 0.1 = 1.9\]

- If 100 tasks are run in parallel of 100 servers, what is the expected time to complete all of them.
  \[1 \times 0.9^{100} + 10 \times (1 - 0.9^{100}) \approx 10\]
Straggler Replication

**PROBLEM:** Slowest tasks become a bottleneck

**SOLUTION:** Replicate the stragglers and wait for one copy

**PARAMETERS**
- p: Frac. of tasks replicated
- r: # additional replicas
- c: kill/keep original task

Eg. MapReduce, Apache Spark launch 1 replica, keep original copy
Straggler Replication Analysis

PARAMETERS
p: Frac. of tasks replicated
r: # additional replicas
c: kill/keep original task

METRICS
E[T] = Time to finish all tasks
E[C] = Total server runtime per task

\[ E[T] = E[X_{(1-p)n:n}] + E[Y_{pn:pn}] \]

Central Value Theorem
\[ F_X^{-1}(1 - p) \]

Extreme Value Theorem
\[ n \to \infty \]

Different behavior for Exponential, Light or Heavy tailed Y

Y is the residual service time after adding replicas
Simulations using Google Cluster Data

Latency-Cost Trade-off

Expected Latency $E[T]$

Expected Cost $E[C]$

Increasing fraction of tasks replicated

- MapReduce setting ($r=1$)
- $r=2$ & keep original copy
- $r=3$ & keep original copy

Careful choice of replication strategy can be better than the default in MapReduce
Task Replication in Queueing Systems
Task Replication in Cloud Computing

IDEA: Assign task to multiple servers and wait for earliest copy

COST

- Additional computing time at servers

Wait for the earliest copy to finish, and cancel the rest
Task Replication in Cloud Computing

**IDEA:** Assign task to multiple servers and wait for earliest copy

**COST**
- Additional computing time at servers
- Increased queuing delay for other tasks

Wait for the earliest copy to finish, and cancel the rest
Design Questions

- How many replicas to launch?
- Which queues to join?
- When to issue and cancel the replicas?
Surprising Insight

In certain regimes, replication could make the whole system faster, and cheaper!

Effective service rate > Sum of individual servers
Distributed Cloud Storage

- Content is replicated on the cloud for reliability
- Can support more users simultaneously
- Replicated used for “hot” data, i.e. more frequent accessed
- Any 1 out of 3 copies is sufficient
Erasure Coded Storage

- With an (n,k) MDS code, any k out of n chunks are sufficient
  - Facebook, Google, Microsoft use (14,10) or (7,4) codes
  - Currently used for cold data, increasing for hot data

Any k=2 out of n=3 are sufficient
(n,k) Reed-Solomon Codes: 1960

- Data: $d_1, d_2, d_3, \ldots d_k$
- Polynomial: $d_1 + d_2 x + d_3 x^2 + \ldots + d_k x^{k-1}$
- Parity bits: Evaluate at n-k points:
  
  - $x=1$: $d_1 + d_2 + d_3 + d_4$
  - $x=2$: $d_1 + 2d_2 + 4d_3 + 8d_4$
  - $x=3$: $\ldots$
  - $x=4$: $\ldots$
  - $x=n$: $\ldots$

- Can solve for the coefficients from any k coded symbols
Example: (4,2) Reed-Solomon Code

- Data: $d_1$, $d_2$  Polynomial: $d_1 + d_2 x + d_3 x^2 + \ldots + d_k x^{k-1}$

- Can solve for the coefficients from any $k$ coded symbols
- Microsoft uses (7, 4) code
- Facebook uses (14,10) code
Locality and Repair Issues

- Repairing failed nodes is hard with Reed-Solomon Codes.

- If we lose 1 node:
  - Need to contact k other nodes
  - Need to download k times the lost data
Solution: Locally Repairable Codes

- Codes designed to minimize:
  - Repair Bandwidth
  - Number of nodes contacted
Coded Computing and ML

- So far: Coding for storage
- Codes can also speed up computing and machine learning!
- Example: Matrix-Vector Multiplication
Coded Computing and ML

- So far: coding for storage
- Codes can also speed up computing and machine learning!
- Example: Matrix-Vector Multiplication
Coded Computing and ML

- So far: coding for storage
- Codes can also speed up computing and machine learning!
- Example: Matrix-Vector Multiplication

\[ \begin{align*}
A_1 & \times X \\
A_1 & \times X \\
A_1 + A_2 & \times X \\
\end{align*} \]

Need only 2 out of 3 to finish
Topics Covered

- SGD and its convergence
- Asynchronous and Local-Update SGD
- Gradient and Model Compression
- Hyper-parameter tuning
The unprecedented ML boom

NeurIPS Growth

Total Registrations 3755

- Tutorials (2,584)
- Conference (3,262)
- Workshops (3,006)
The Origins: 1950

Alan Turing

Can a computer talk like a human?
Neural Networks: Perceptron 1957

Original Perceptron


Simplified model:

Frank Rosenblatt (1928-1971)
Learning representations by back-propagating errors

DAVID E. RUMELHART\(^1\), GEOFFREY E. HINTON\(^2\) & RONALD J. WILLIAMS\(^3\)

\(^1\)Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
\(^2\)Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA
\(^3\)To whom correspondence should be addressed.

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure replaces the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure\(^1\).

References

MNIST (LeCun et al 1998)
ImageNet and ILSVRC (2012)

Fei-Fei Li, Stanford
ImageNet and ILSVRC

ImageNet Classification top-5 error (%)

<table>
<thead>
<tr>
<th>Year</th>
<th>Method</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>NEC America</td>
<td>28.2</td>
</tr>
<tr>
<td>2011</td>
<td>Xerox</td>
<td>25.8</td>
</tr>
<tr>
<td>2012</td>
<td>AlexNet</td>
<td>16.4</td>
</tr>
<tr>
<td>2013</td>
<td>Clarif</td>
<td>11.7</td>
</tr>
<tr>
<td>2014</td>
<td>VGG</td>
<td>7.3</td>
</tr>
<tr>
<td>2014</td>
<td>GoogleNet</td>
<td>6.7</td>
</tr>
<tr>
<td>2015</td>
<td>ResNet</td>
<td>3.5</td>
</tr>
</tbody>
</table>
Why the sudden success?

- Availability of massive datasets like Imagenet

- Computing power to train deep neural networks
  - Parallelization
  - GPUs

- Algorithmic advances:
  - Momentum, Adagrad, Adam etc.
Core of ML: Gradient Descent (GD)
Simplest ML example: Regression

Given a big dataset of \((x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), \ldots, (x_N, y_N)\)

Find the optimal weights \(w\)
Core of ML: Gradient Descent (GD)

\[
\min_{\mathbf{w}} F(\mathbf{w}) = \min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} \nabla (y_i - \mathbf{w}^T \mathbf{x})^2
\]
Core of ML: Gradient Descent (GD)

\[ w_{t+1} = w_t - \eta \nabla F(w_t) \]
Exercise: Find the update rule for $w_a$ and $w_b$

Given a big dataset of $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), \ldots,(x_N, y_N)$

Find the optimal weights $\mathbf{w} = (w_a, w_b)$
Gradient Descent (GD)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \frac{1}{N} \sum_{i=1}^{N} \nabla (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

Too expensive for large datasets
Stochastic Gradient Descent (SGD)

\[ w_{t+1} = w_t - \eta \nabla (y_i - w^T x_i)^2 \]

Easy, but possibly too noisy
Mini-batch SGD

$$w_{t+1} = w_t - \eta \frac{1}{m} \sum_{i=1}^{m} \nabla (y_i - w^T x_i)^2$$

Less noisy, but also computationally tractable
Convergence of SGD

\[ \mathbb{E}[F(w_k) - F_*] \leq \frac{\eta LM}{2c} + (1 - \eta c)^{k-1}(F(w_0) - F_* - \frac{\eta LM}{2c}) \]

How does decay rate and error floor change with
- \( \eta \) (Learning Rate) ?
- \( M \) (Second moment of gradient) ?
Many other variants of SGD

- Momentum SGD
- Nesterov Momentum
- AdaGrad
- Adam
- AdaDelta
- RMS prop
Many other variants of SGD
Many other variants of SGD
Given a big dataset of \((x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), \ldots, (x_N, y_N)\)
Find the optimal weights \(\mathbf{w}\)
Distributed Deep Learning

Data Parallelism

$$w' = w - \eta \Delta w$$
Distributed Deep Learning

Model Parallelism
Synchronous SGD

\[ w_{t+1} = w_t - \eta \sum_{k=1}^{K} \frac{1}{K} \nabla F(w_t, \xi_k) \]
Q: What is the convergence rate and error floor?

\[ w_{t+1} = w_t - \eta \sum_{k=1}^{K} \frac{1}{K} \nabla F(w_t, \xi_k) \]
Q: What is the time to complete each iteration?

$$\mathbb{E}[T] = \mathbb{E}[\max(X_1, X_2, \ldots X_K)]$$

- Slowest worker is the bottleneck

- Parameter Server: $w' = w - \eta \Delta w$

- Fully Sync-SGD:
  - PS: $w_0, w_1, w_2$
  - $L_1, L_2, L_3$
Q: How can we reduce it?

\[ \mathbb{E}[T] = \mathbb{E}[\max(X_1, X_2, \ldots X_K)] \]

Slowest Worker is the bottleneck.

Diagram:
- Parameter Server: \( w' = w - \eta \Delta w \)
- Model Replicas
- Data Shards
- Fully Sync-SGD: \( w_0, w_1, w_2 \)
- DGSM: L1, L2, L3
Asynchronous SGD: Don’t wait for all

Asynchronous SGD cuts the latency tail.

But, what effect does it have on the error?
Variants of Distributed SGD

- Synchronous SGD
- Asynchronous SGD
- HogWild
- Elastic-Averaging SGD
- Federated Learning
- Gradient Compression/Quantization
Hyper-Parameter Tuning

Need to choose the right

• Learning rate
• Mini-batch size
• Momentum
• Number of layers
• Number of neurons per layer
Hyper-Parameter Tuning
Hyper-Parameter Tuning

We will discuss

- Multi-armed Bandit based tuning
- Bayesian Optimization Methods