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

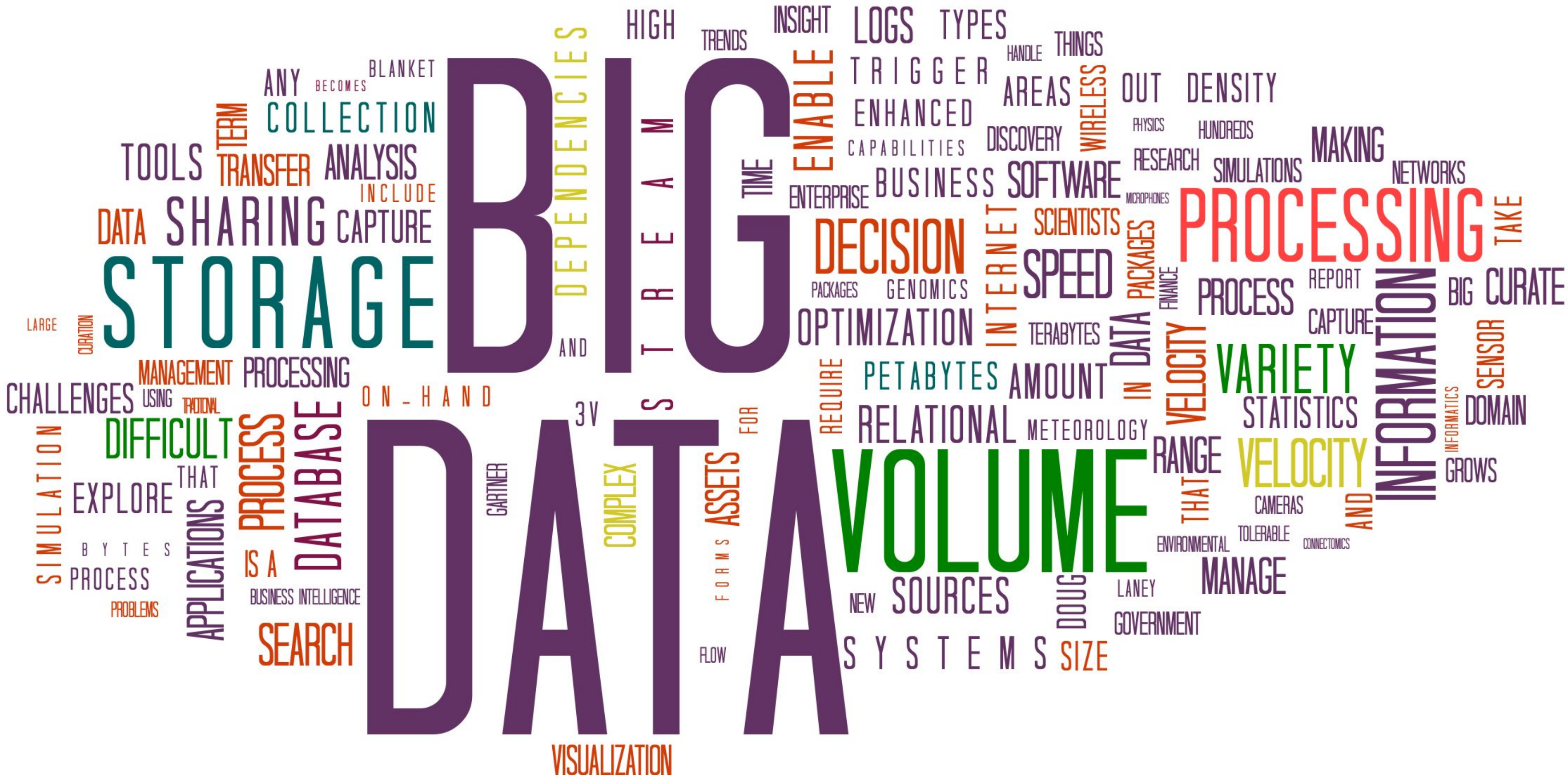
Class Presentations and Discussion

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

Google Compute Engine

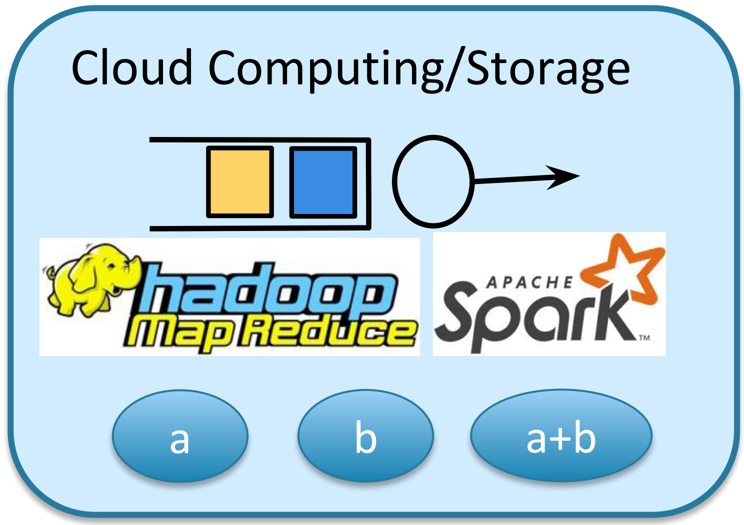
amazon web services

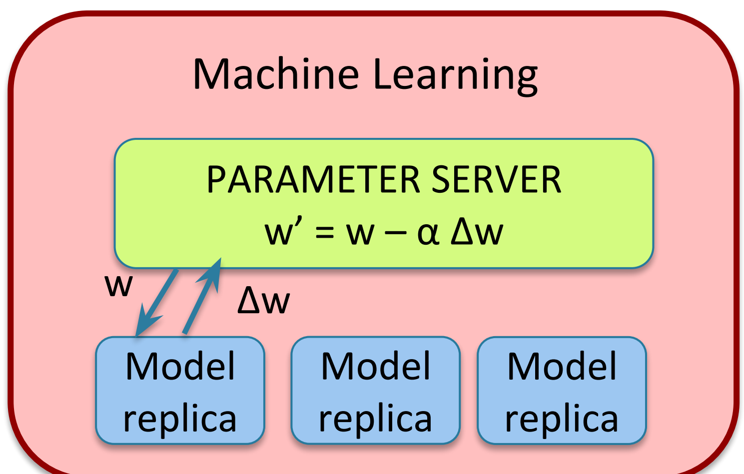
IBM Bluemix™

Who got rich in the California gold rush?

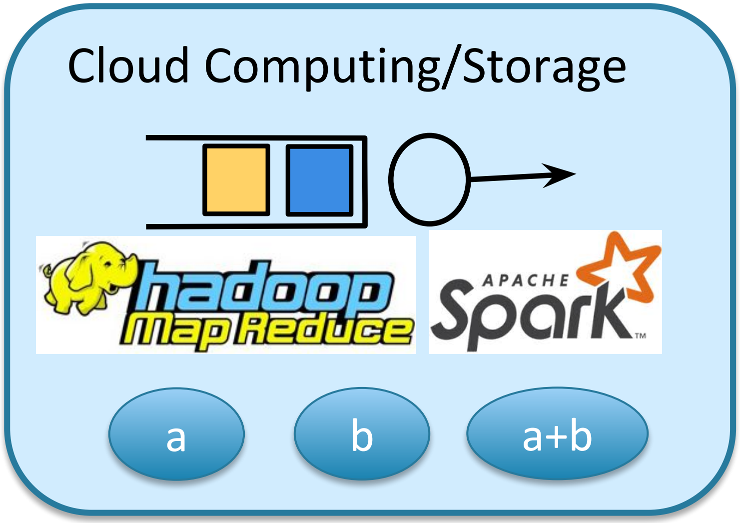
In the Big Data rush, it's the infrastructure companies

Topics Covered





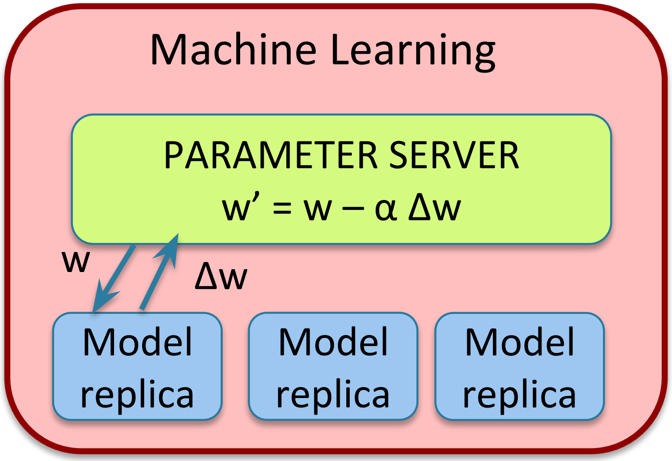
Topics Covered



- 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

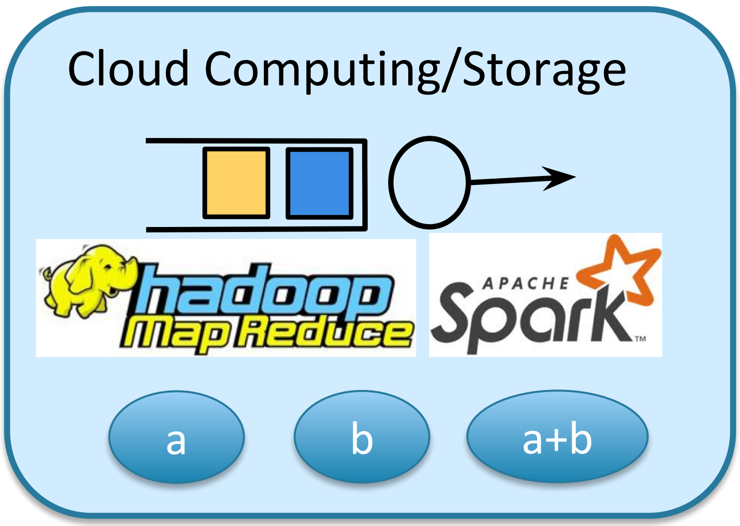
You will learn..

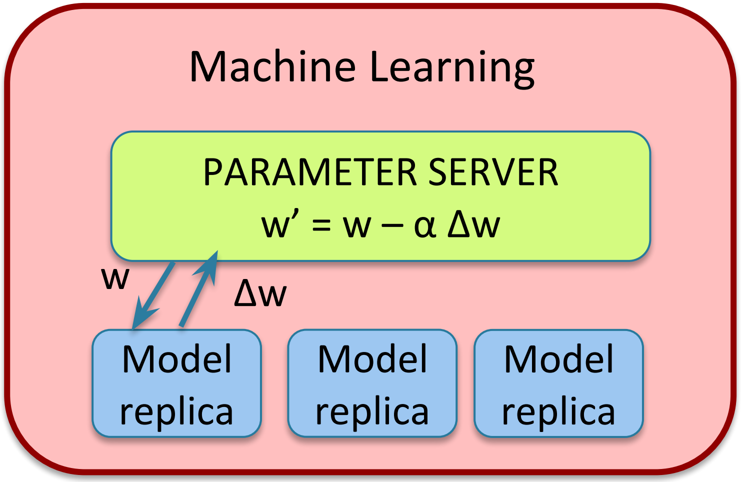
- Latest cloud and ML infrastructure research
- How to read and critique research papers
- How to present effectively
- Some programming in TensorFlow/PyTorch

TO DO

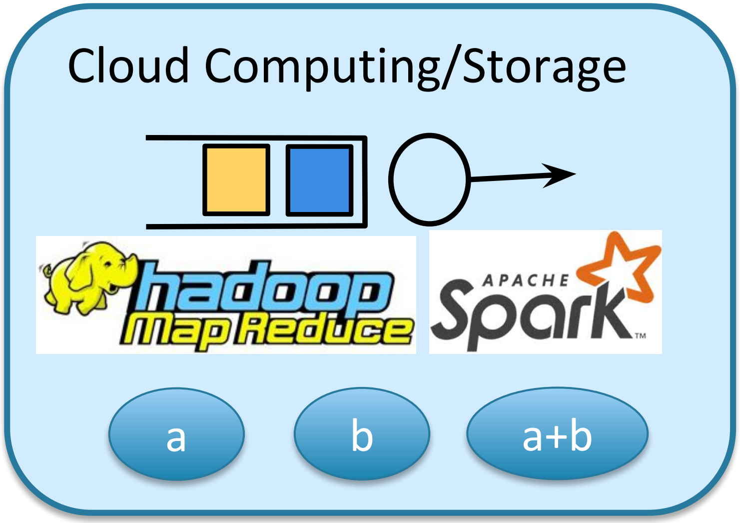
- Sign-up for presentation
- Start reading the papers

Topics Covered





Topics Covered



- 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

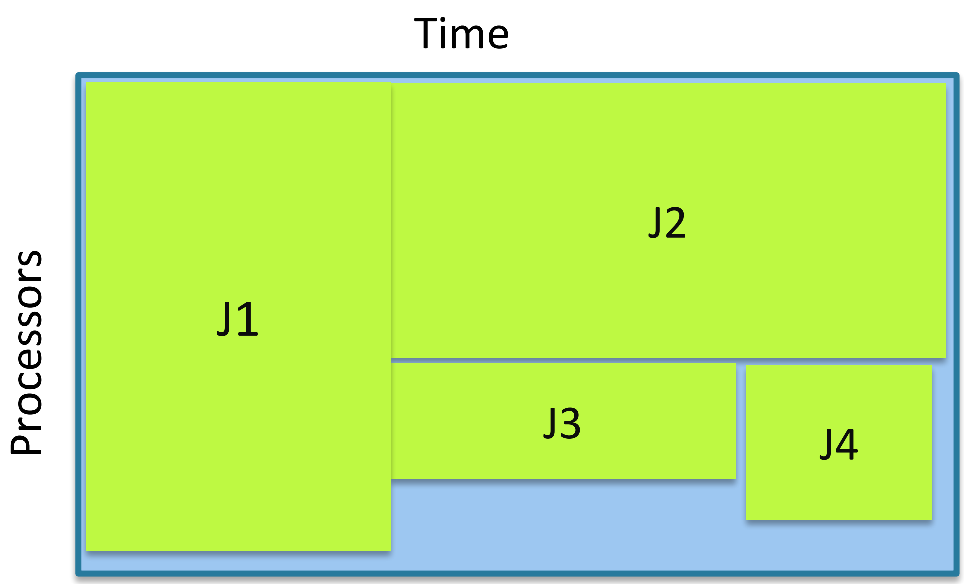
IBM 704 (1964)

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

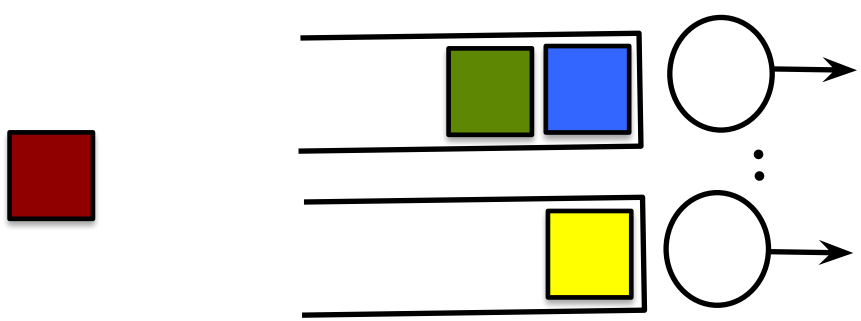
- **Bin-Packing**
  - Need job size estimates



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

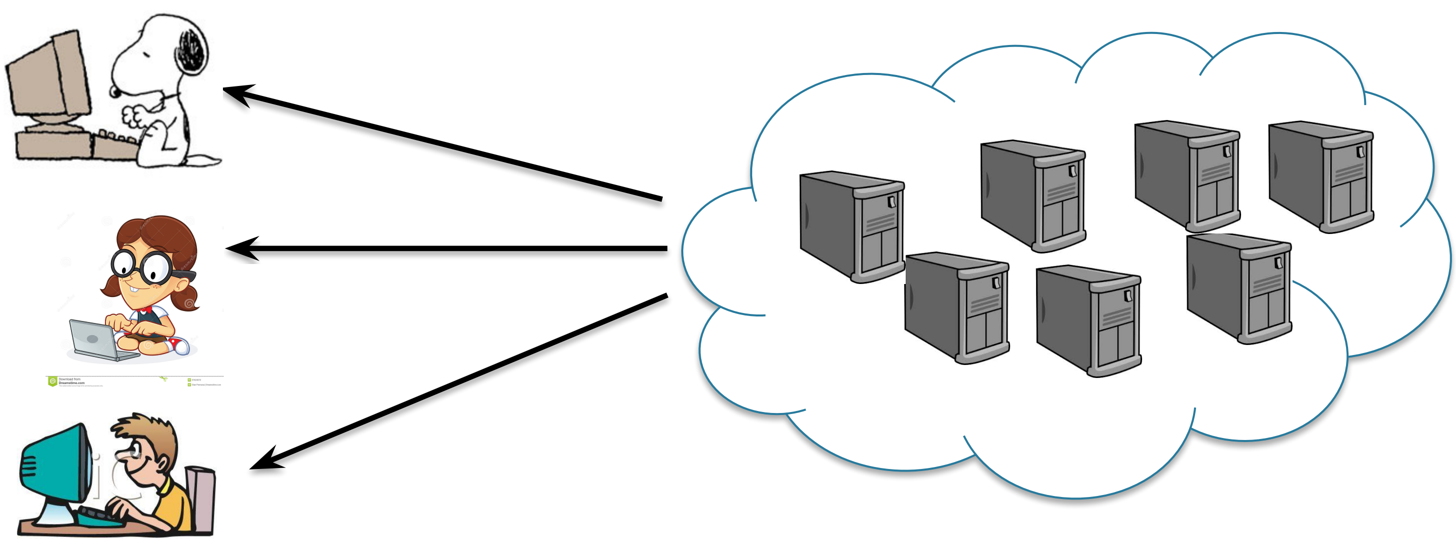
- Computing become a utility, like electricity

2000's: The Cloud Computing Era

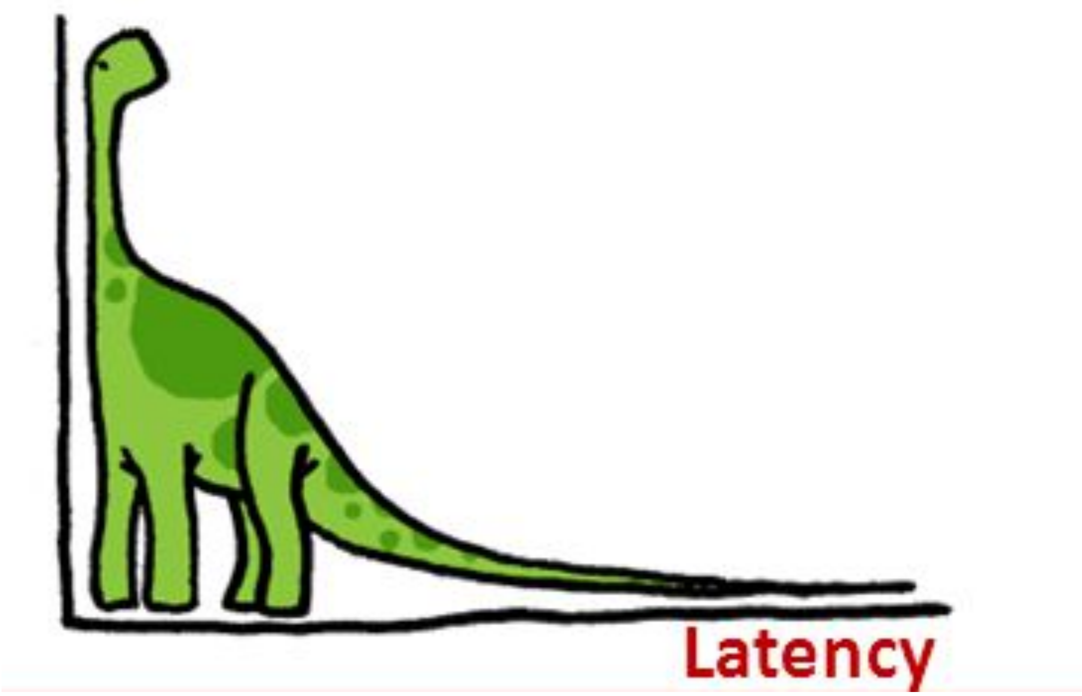
KEY ISSUE: Job sizes, server speeds & queue lengths are unpredictable

REASON: Large-scale resource sharing □ Variability in service

- Virtualization, server outages etc.
- Norm and not an exception [Dean-Barroso 2013]

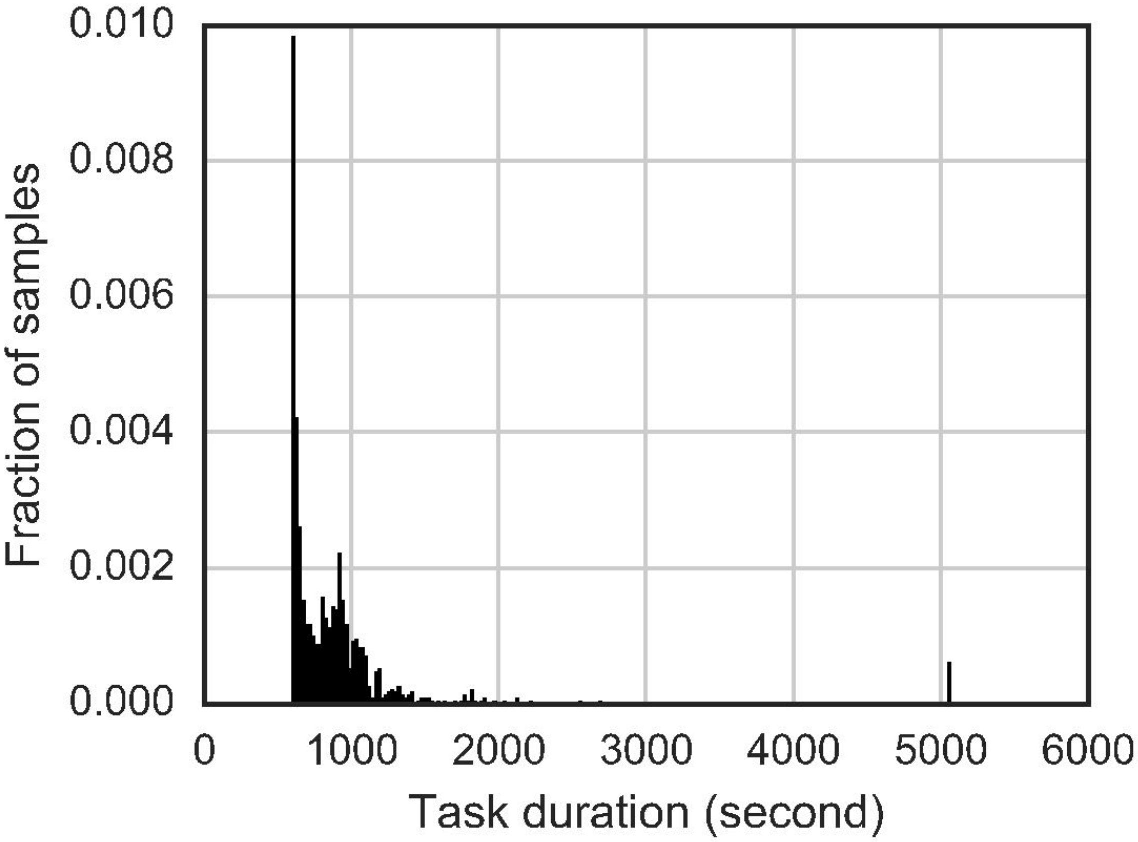


The Tale of Tails



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.

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*0.9 + 10*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?

  $1*0.9 + 10*0.1 = 1.9$

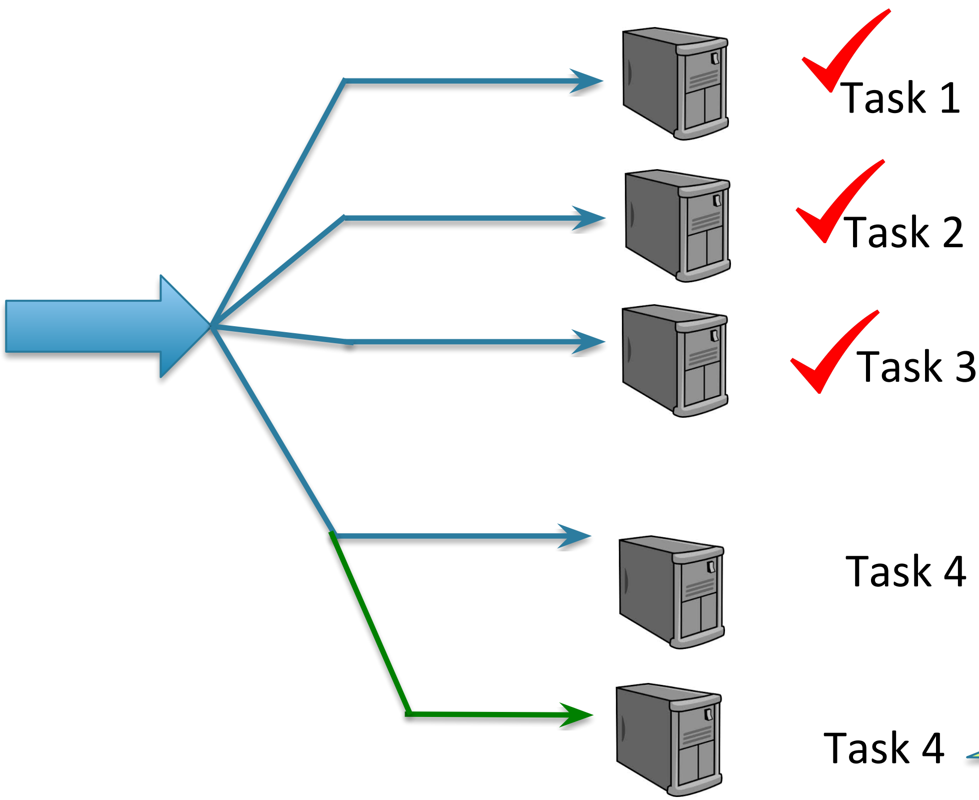
- If 100 tasks are run in parallel of 100 servers, what is the expected time to complete all of them.

  $1*0.9^{100} + 10*(1- 0.9^{100}) \sim 10$

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

[Wang-GJ-Wornell SIGMETRICS 2014, 15]

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

Y is the residual service time after adding replicas

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

Central Value Theorem

n -> ∞

$$F_X^{-1}(1-p)$$

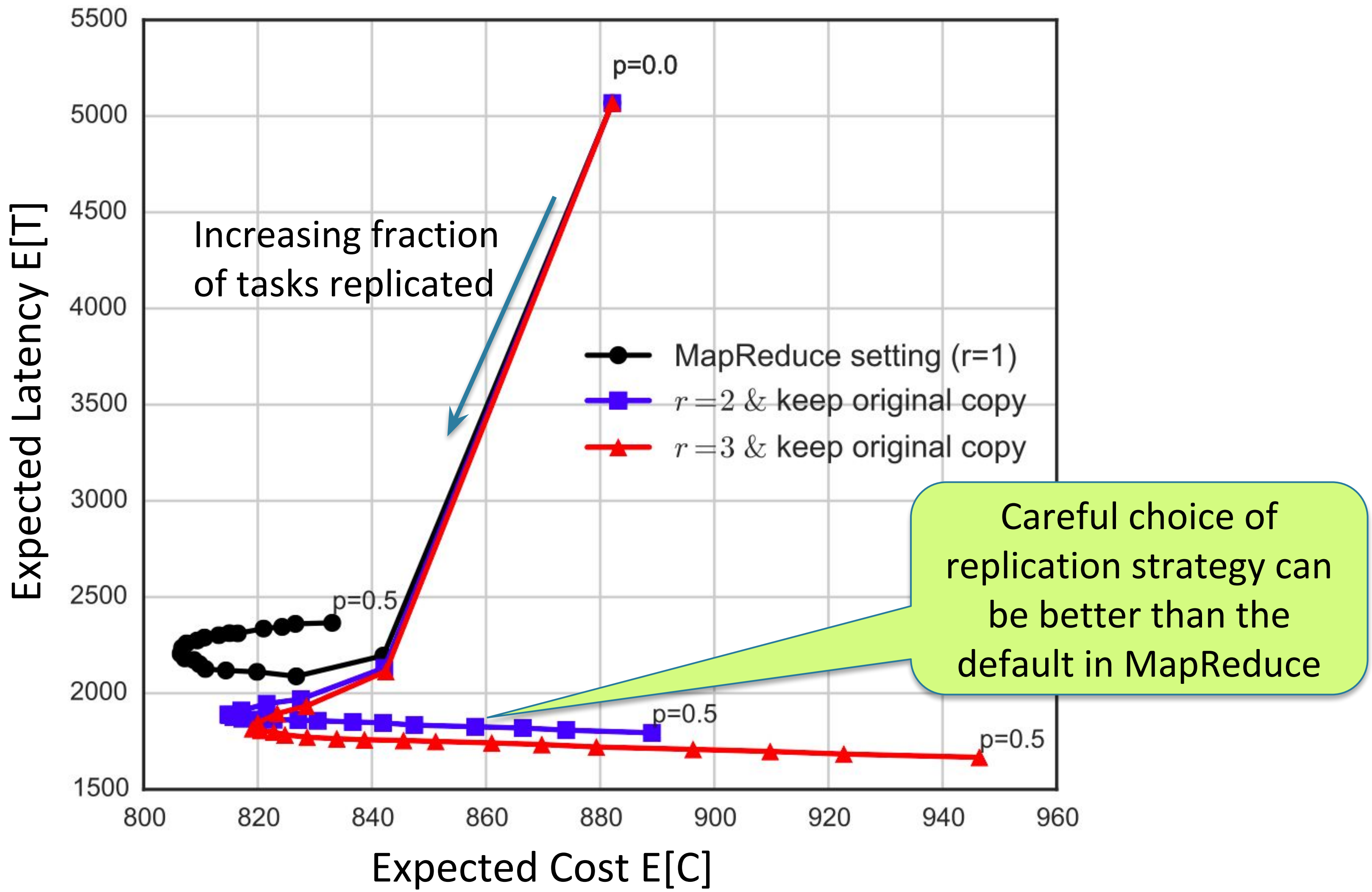
Extreme Value Theorem

n -> ∞

Different behavior for Exponential, Light or Heavy tailed Y

Simulations using Google Cluster Data

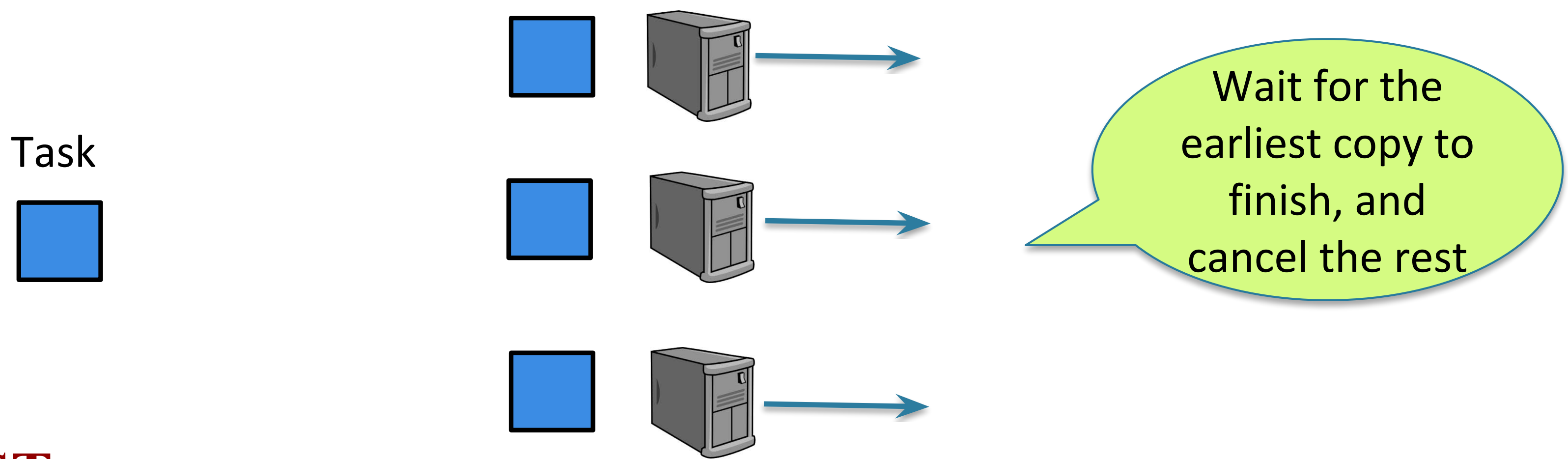
Latency-Cost Trade-off



Task Replication in Queueing Systems

Task Replication in Cloud Computing

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

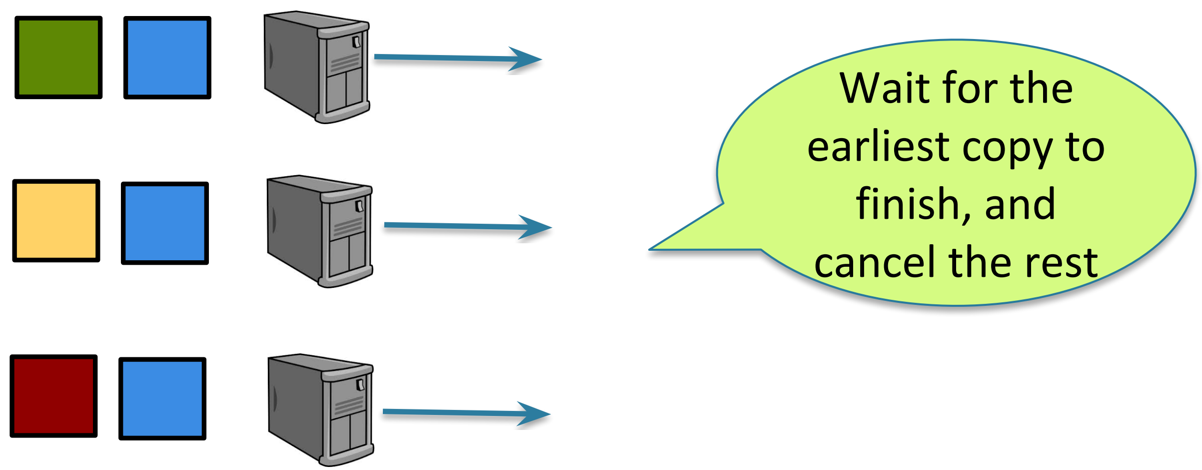


COST

- Additional computing time at servers

Task Replication in Cloud Computing

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

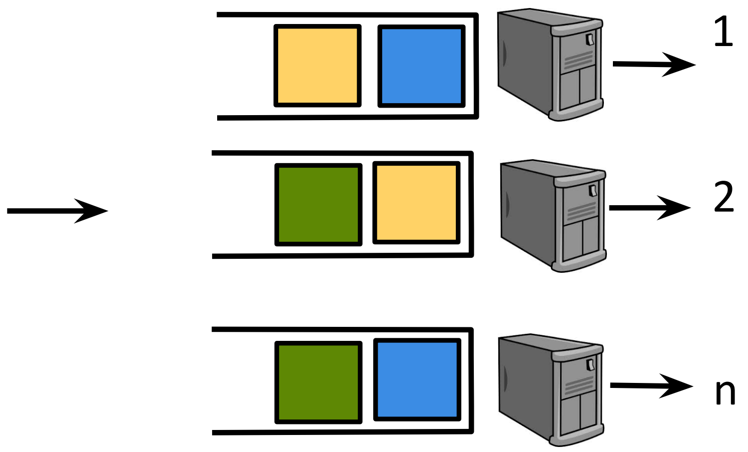


COST

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

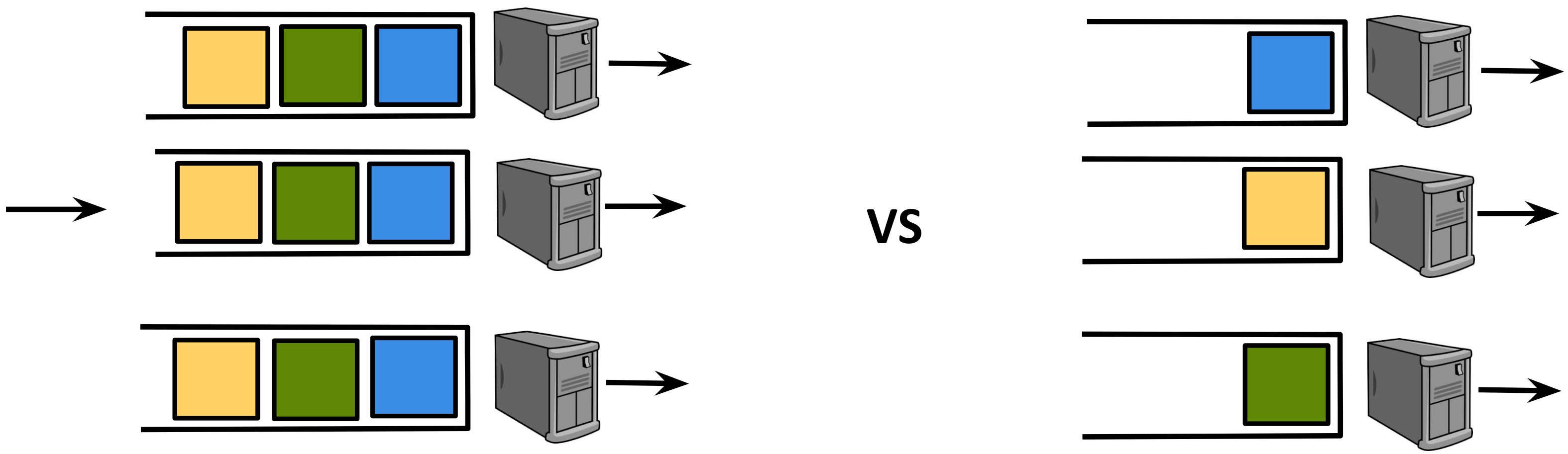
Design Questions

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



Surprising Insight

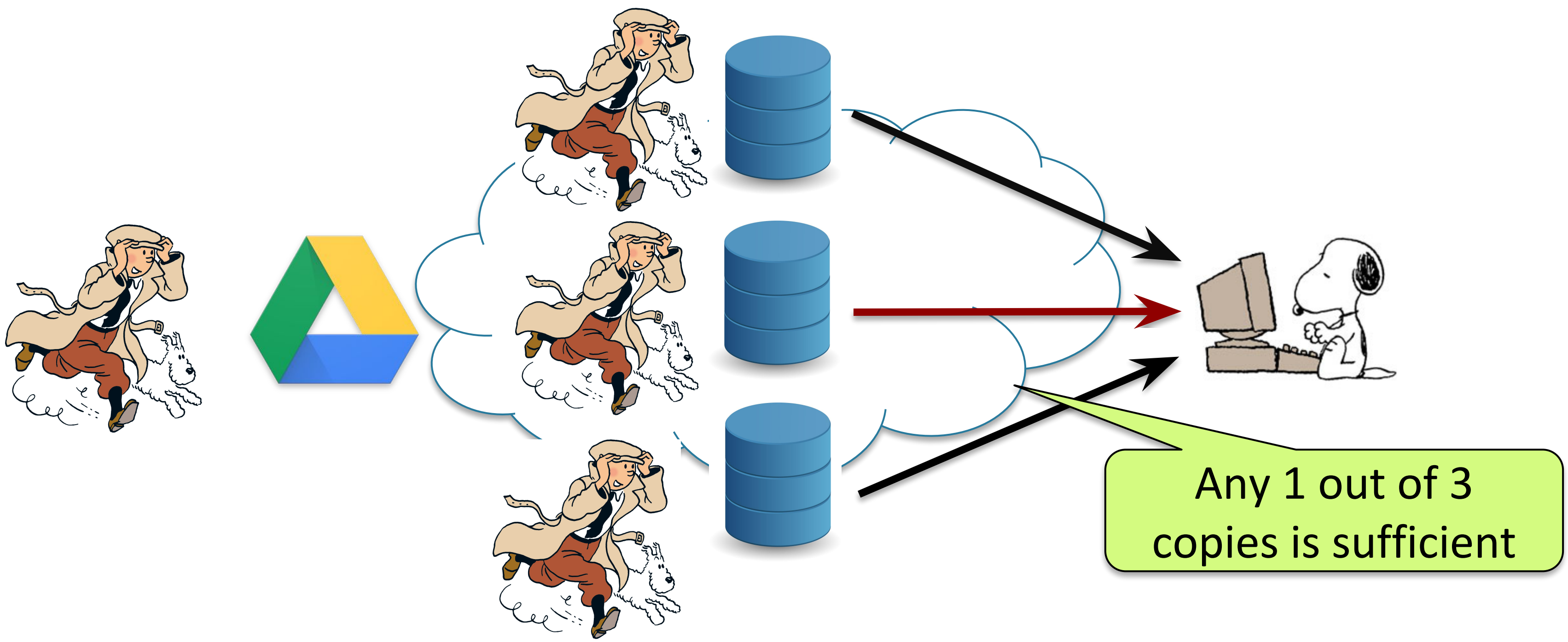
In certain regimes, replication could make the whole system faster, and cheaper!



Effective service rate > Sum of individual servers

Distributed Cloud Storage

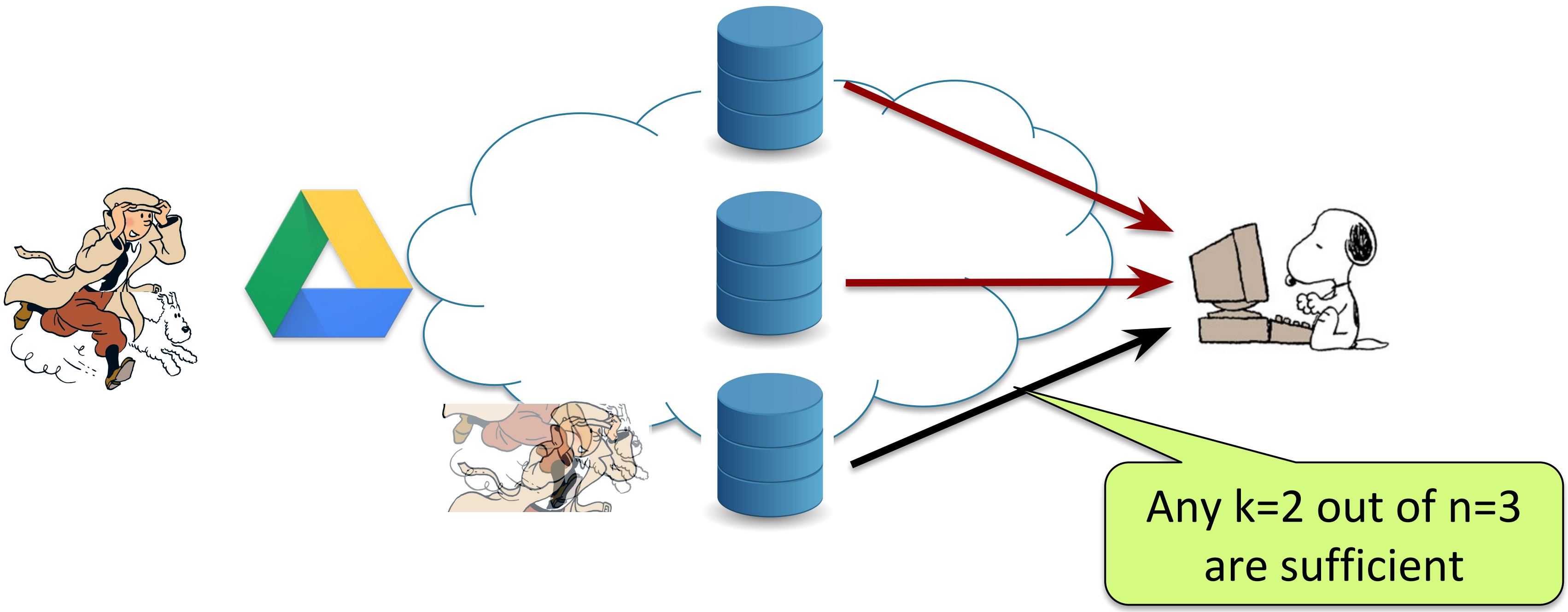
- Content is replicated on the cloud for reliability



- Can support more users simultaneously
- Replicated used for “hot” data, i.e. more frequent accessed

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



(n,k) Reed-Solomon Codes: 1960

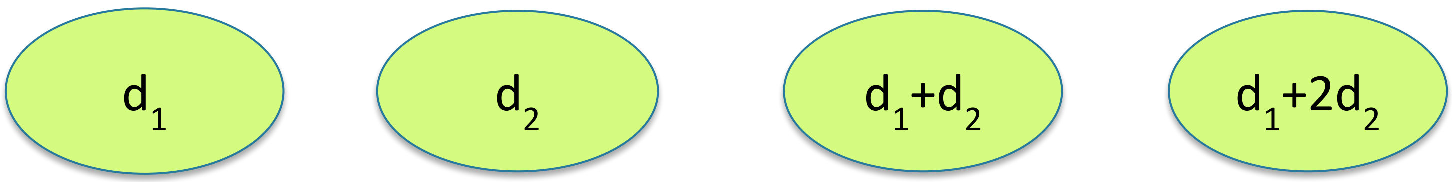
- Data: $d_1, d_2, d_3, \ldots d_k$
- Polynomial: $d_1 + d_2\, x + d_3\, x^2 + \ldots\, d_k\, x^{k-1}$
- Parity bits: Evaluate at n-k points:

x=1:	$d_1 + d_2 + d_3 + d_4$
x=2:	$d_1 + 2\, d_2 + 4\, d_3 + 8\, d_4$
x=3 :	….
x=4:	….
x=n:	…

- Can solve for the coefficients from any k coded symbols

Example: (4,2) Reed-Solomon Code

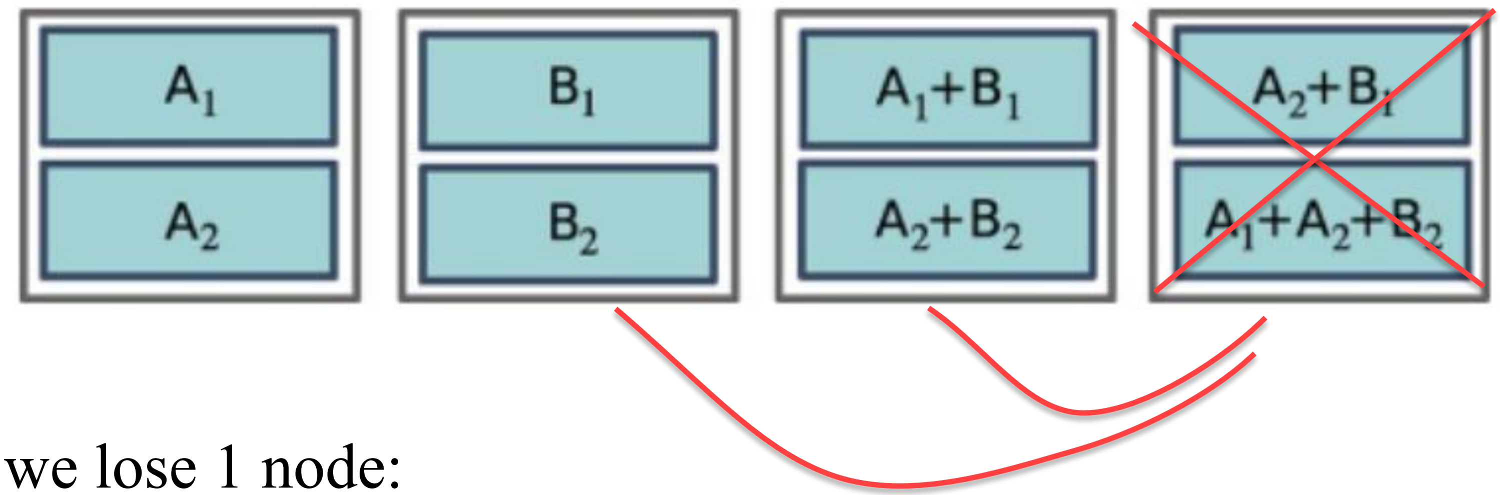
- Data: d_1, d_2 □ Polynomial: $d_1 + d_2\, x + d_3\, x^{2\,+} \ldots\, d_k\, x^{k-1}$



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

Locality and Repair Issues

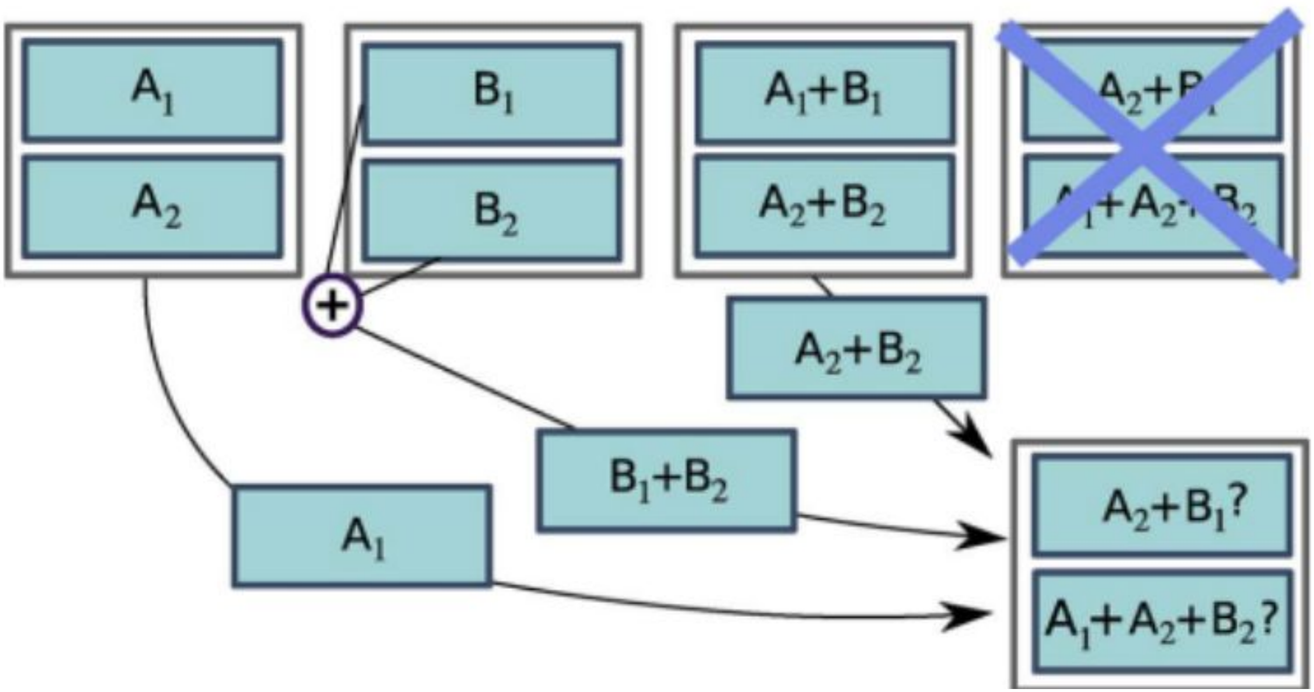
- Repairing failed nodes is hard with Reed-Solomon Codes..



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

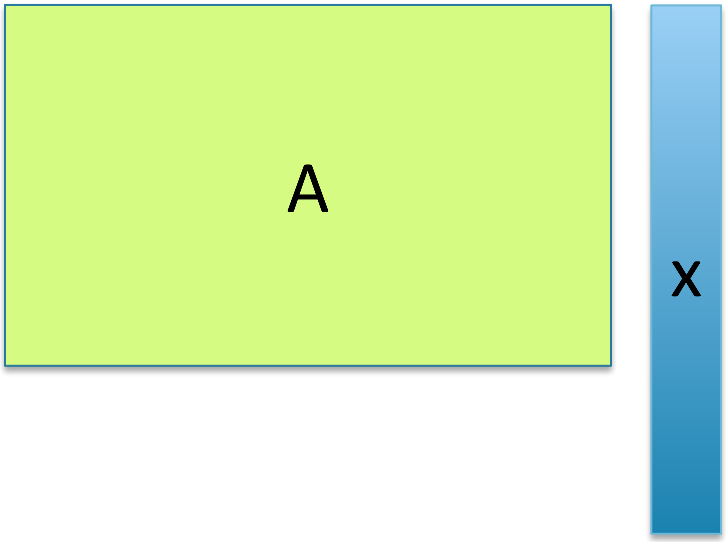
Solution: Locally Repairable Codes

- Codes designed to minimize:
  - Repair Bandwidth
  - Number of nodes contacted



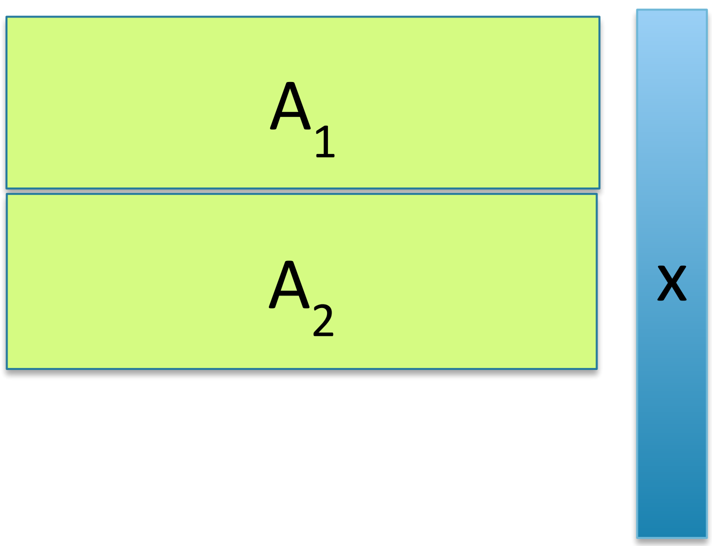
Coded Computing and ML

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



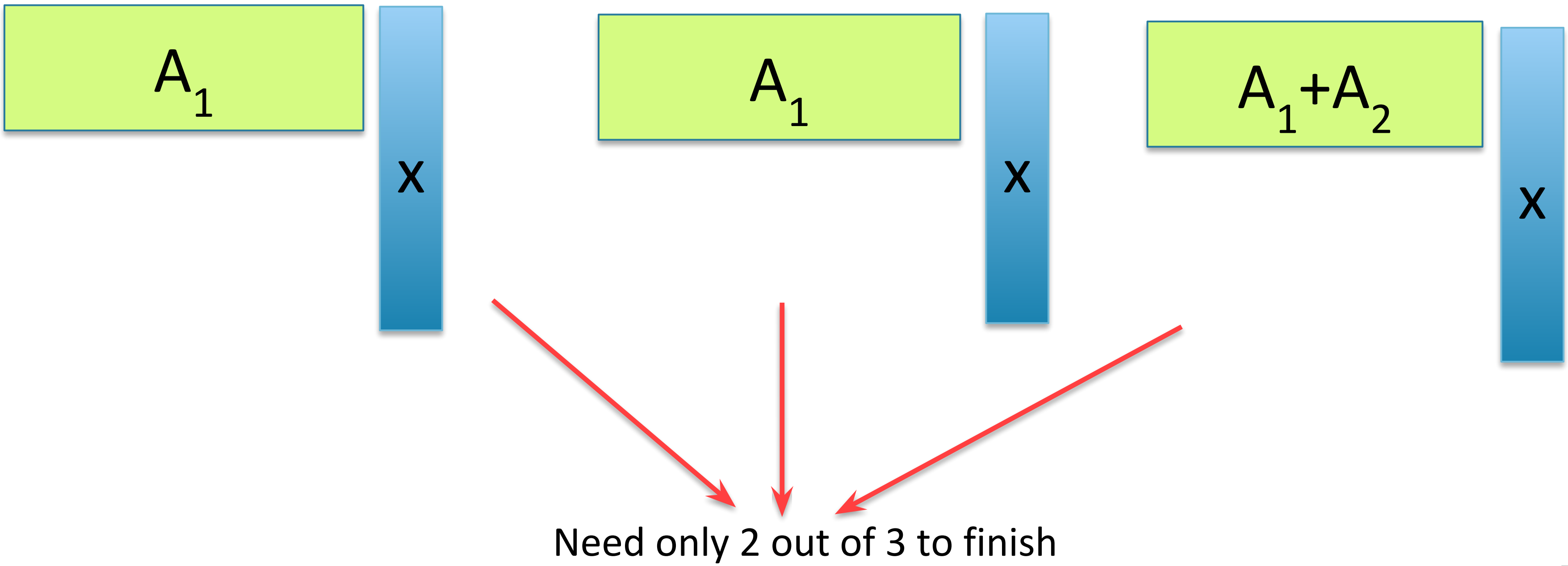
Coded Computing and ML

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



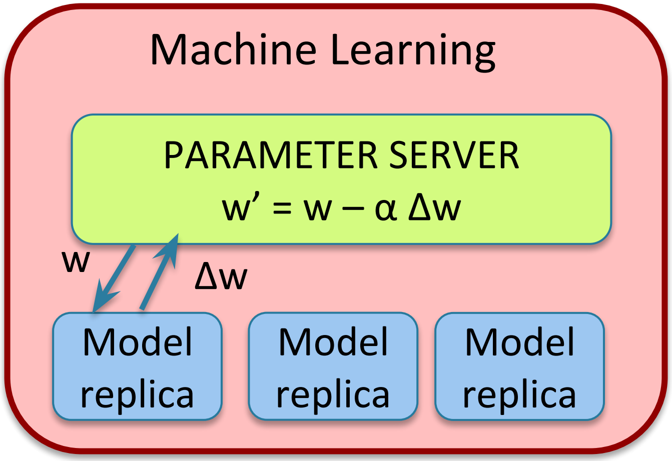
Coded Computing and ML

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

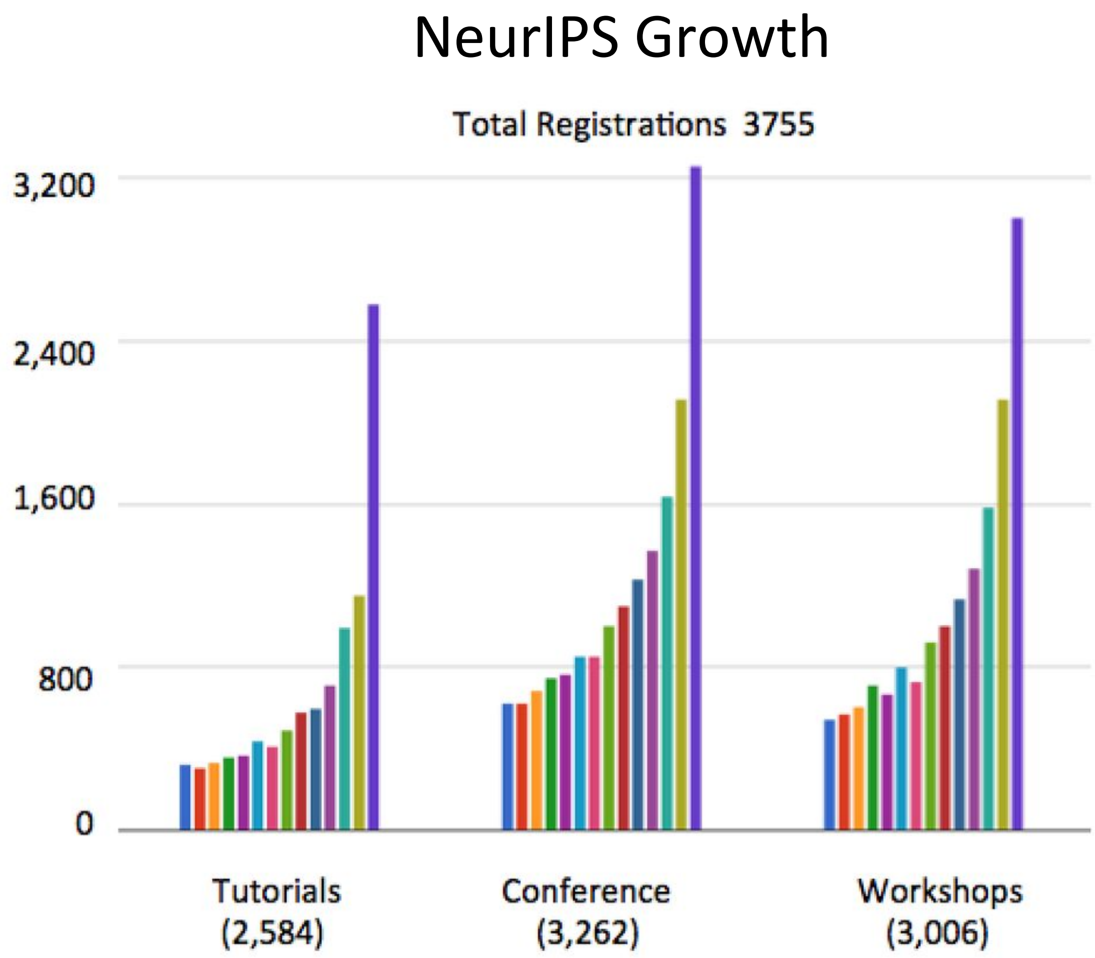


Topics Covered

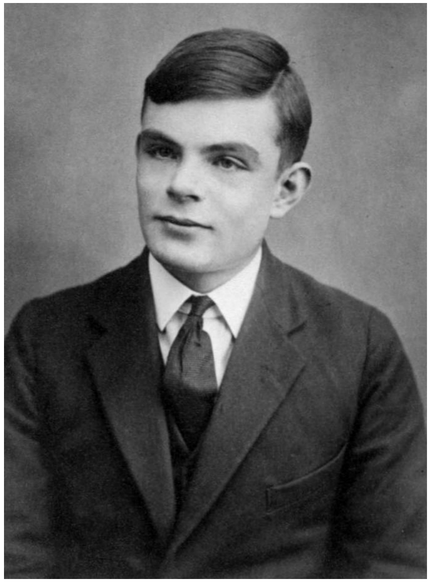
- SGD and its convergence
- Asynchronous and Local-Update SGD
- Gradient and Model Compression
- Hyper-parameter tuning



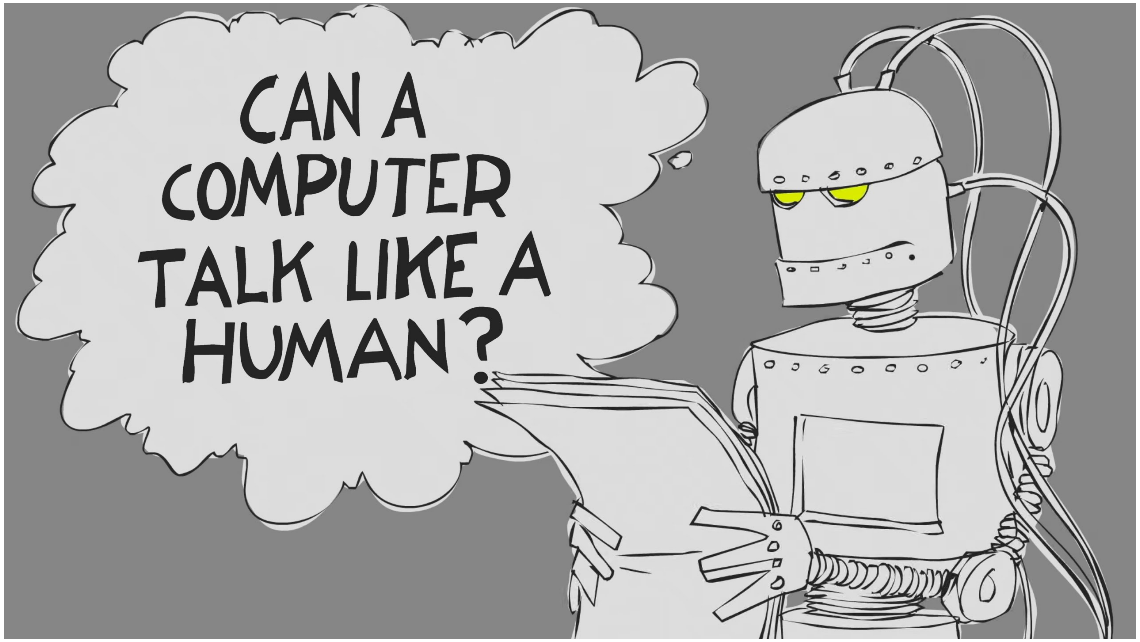
The unprecedented ML boom



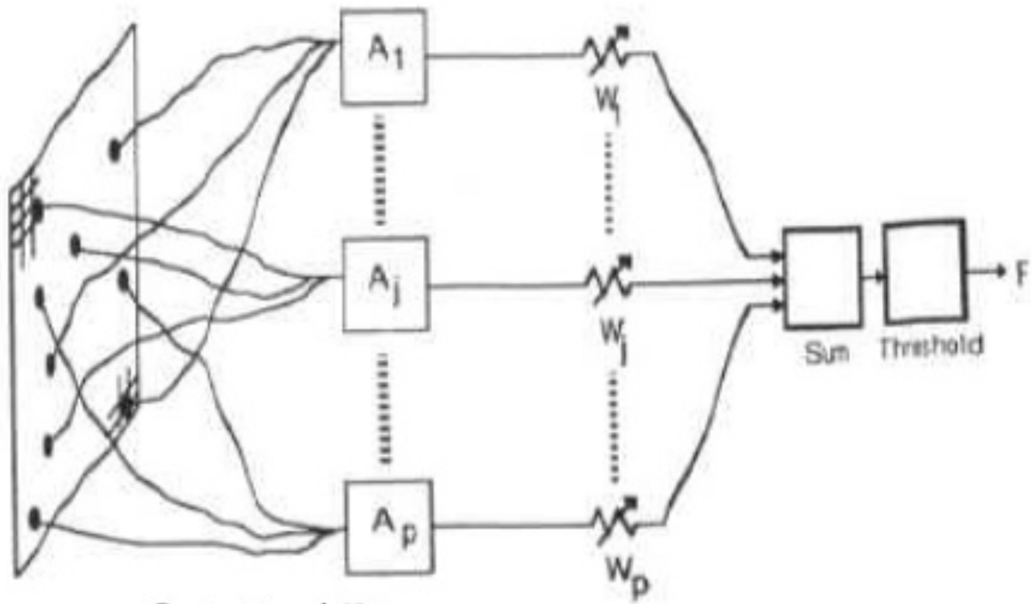
The Origins: 1950



Alan Turing



Neural Networks: Perceptron 1957

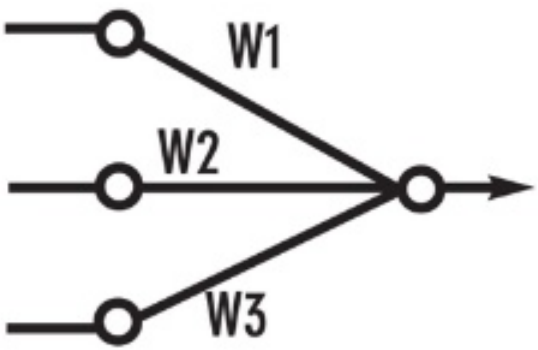


Original Perceptron

(From Perceptrons by M. L Minsky and S. Papert, 1969, Cambridge, MA: MIT Press. Copyright 1969 by MIT Press.

Frank Rosenblatt
(1928-1971)

Simplified model:



Back-propagation Algorithm

Geoff Hinton (U. Toronto, Google)

nature.com | about npg | news@nature.com | naturejobs | natureevents | help | site index

nature

my account | e-alerts | subscribe | register

SEARCH JOURNAL Go

Sunday 22 October 2017

Journal Home
Current Issue
AOP
Archive

THIS ARTICLE
Download PDF
References
Export citation
Export references
Send to a friend
More articles like this
Table of Contents
< Previous | Next >

letters to nature

Nature **323**, 533 - 536 (09 October 1986); doi:10.1038/323533a0

Learning representations by back-propagating errors

DAVID E. RUMELHART[*], GEOFFREY E. HINTON[†] & RONALD J. WILLIAMS[*]

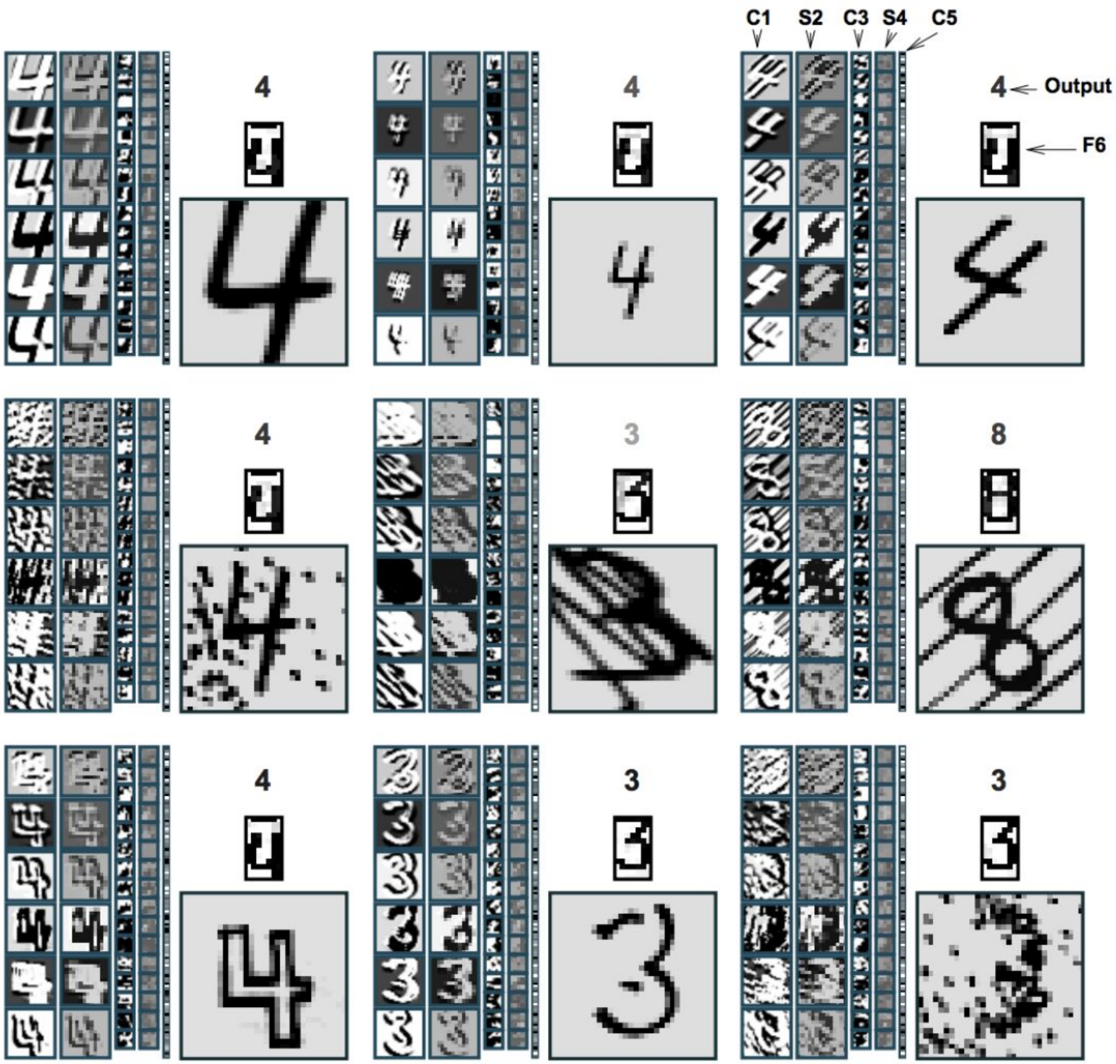
[*]Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
[†]Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA
[†]To whom correspondence should be addressed.

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repe Alan_Turing_Aged_16.jpg **ights 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

1. Rosenblatt, F. *Principles of Neurodynamics* (Spartan, Washington, DC, 1961).

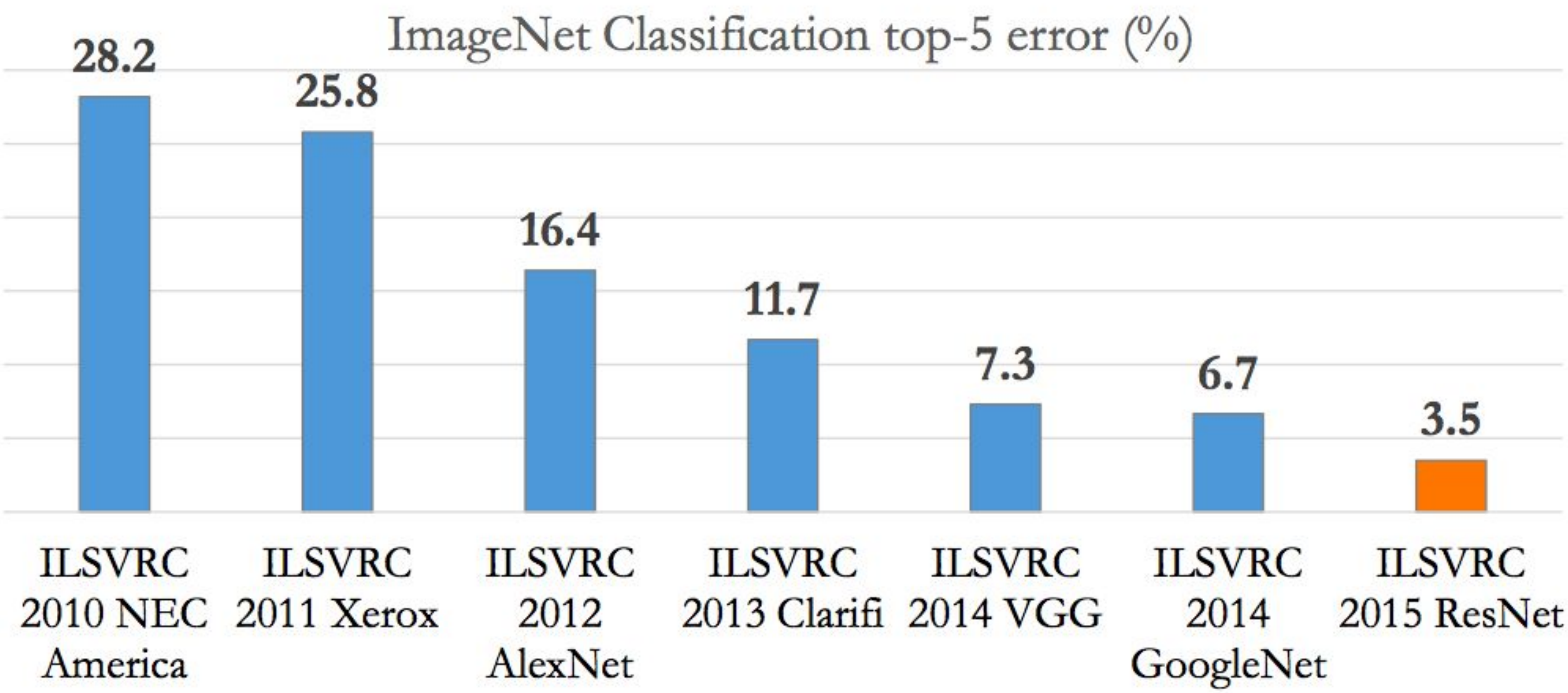
MNIST (LeCun et al 1998)



ImageNet and ILSVRC (2012)

Fei-Fei Li, Stanford

ImageNet and ILSVRC



Why the sudden success?

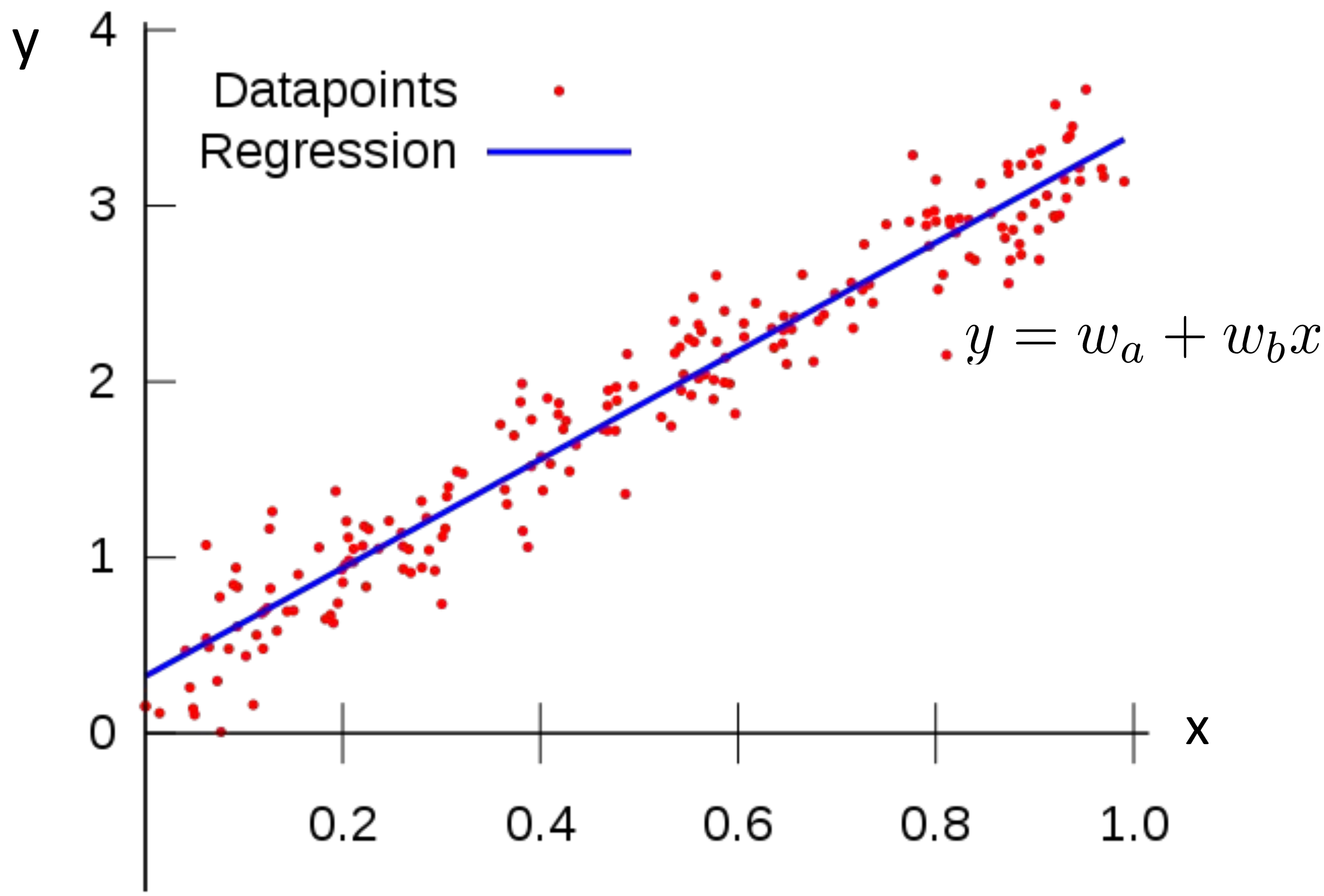
- Availability of massive datasets like Imagenet

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

- Algorithmic advances:
  - Momentum, Adagrad, Adam etc.

Core of ML: Gradient Descent (GD)

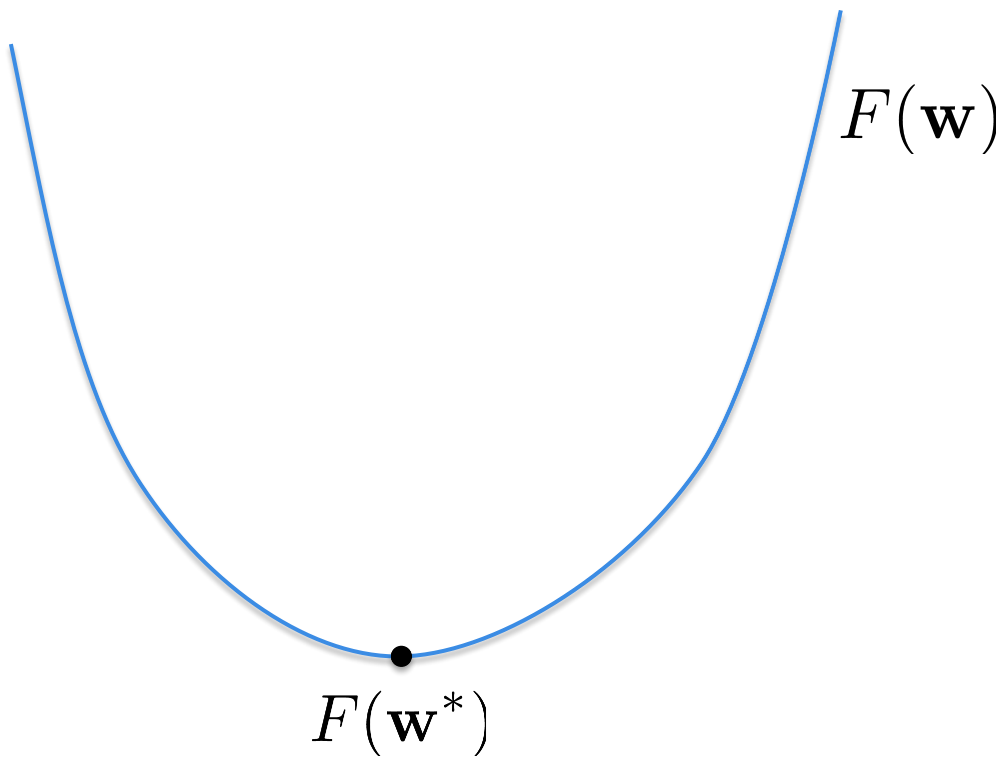
Simplest ML example: Regression



Given a big dataset of ($\mathbf{x_1}$, y_1), ($\mathbf{x_2}$, y_2), ($\mathbf{x_3}$, y_3), ($\mathbf{x_4}$, y_4), ….($\mathbf{x_N}$, y_N)
Find the optimal weights **w**

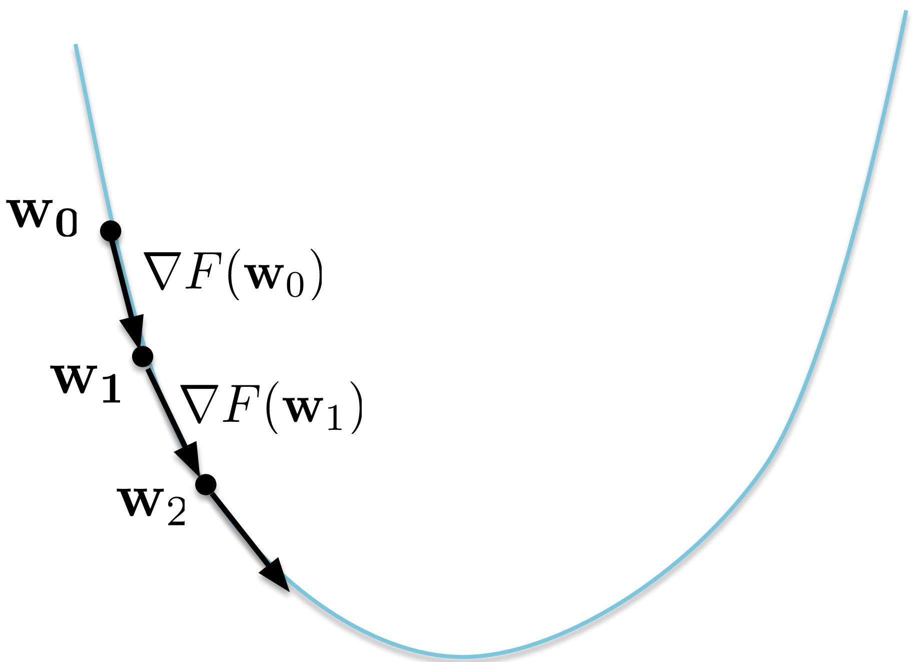
Core of ML: Gradient Descent (GD)

$$\min_{\mathbf{w}} F(\mathbf{w}) = \min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} \nabla (y_i - \mathbf{w}^T \mathbf{x})^2$$

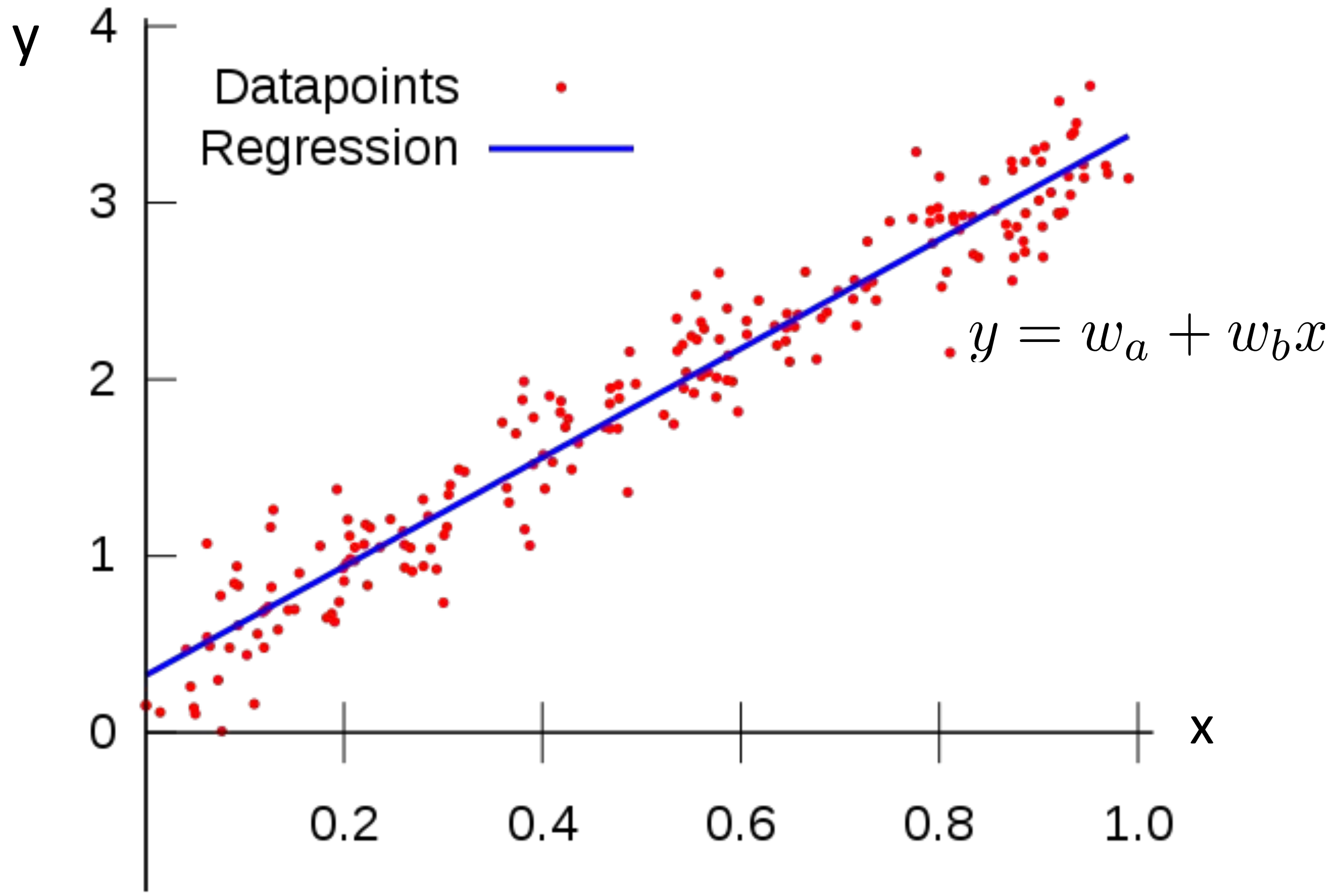


Core of ML: Gradient Descent (GD)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla F(\mathbf{w_t})$$



Exercise: Find the update rule for w_a and w_b



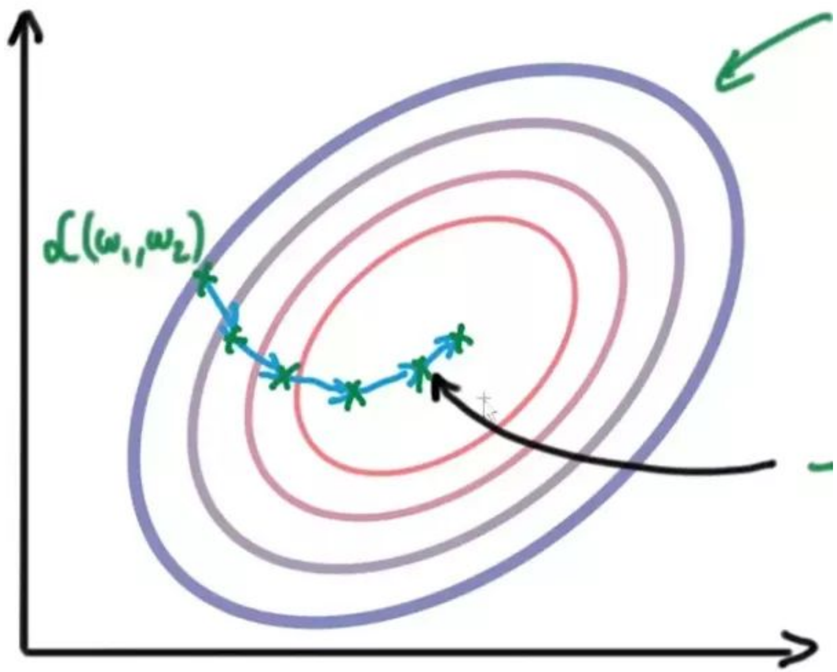
Given a big dataset of $(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), (\mathbf{x_3}, y_3), (\mathbf{x_4}, y_4), \ldots.(\mathbf{x_N}, y_N)$

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

Gradient Descent (GD)

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

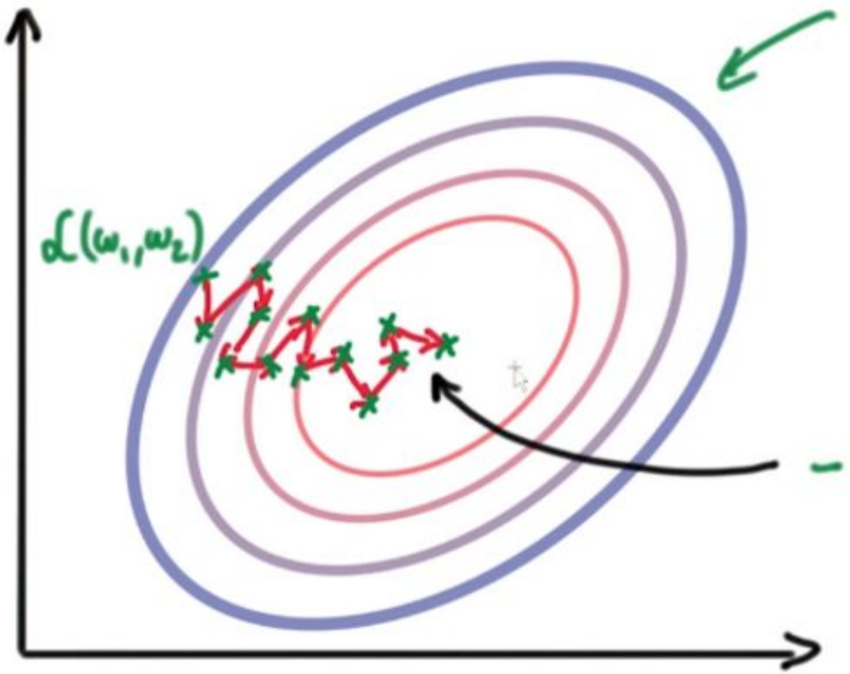
Too expensive for large datasets



Stochastic Gradient Descent (SGD)

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

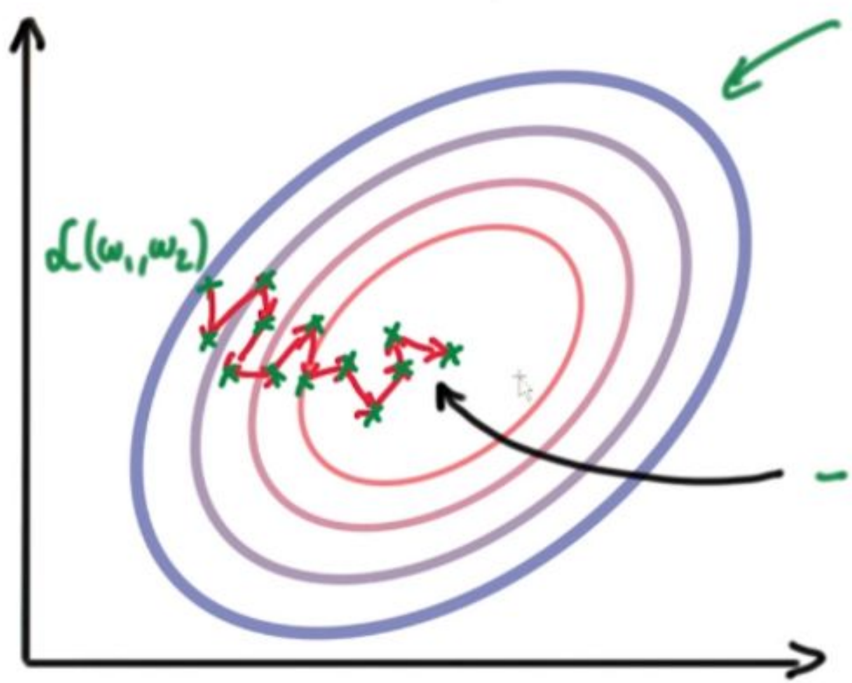
Easy, but possibly too noisy



Mini-batch SGD

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

Less noisy, but also computationally tractable



Convergence of SGD

$$\mathbb{E}[F(\mathbf{w}_k) - F_*] \leq \underbrace{\boxed{\frac{\eta L M}{2c}}}_{\text{Error Floor}} + \underbrace{\boxed{(1 - \eta c)}}_{\text{Decay Rate}}{}^{k-1}\left(F(\mathbf{w}_0) - F_* - \frac{\eta L M}{2c}\right)$$

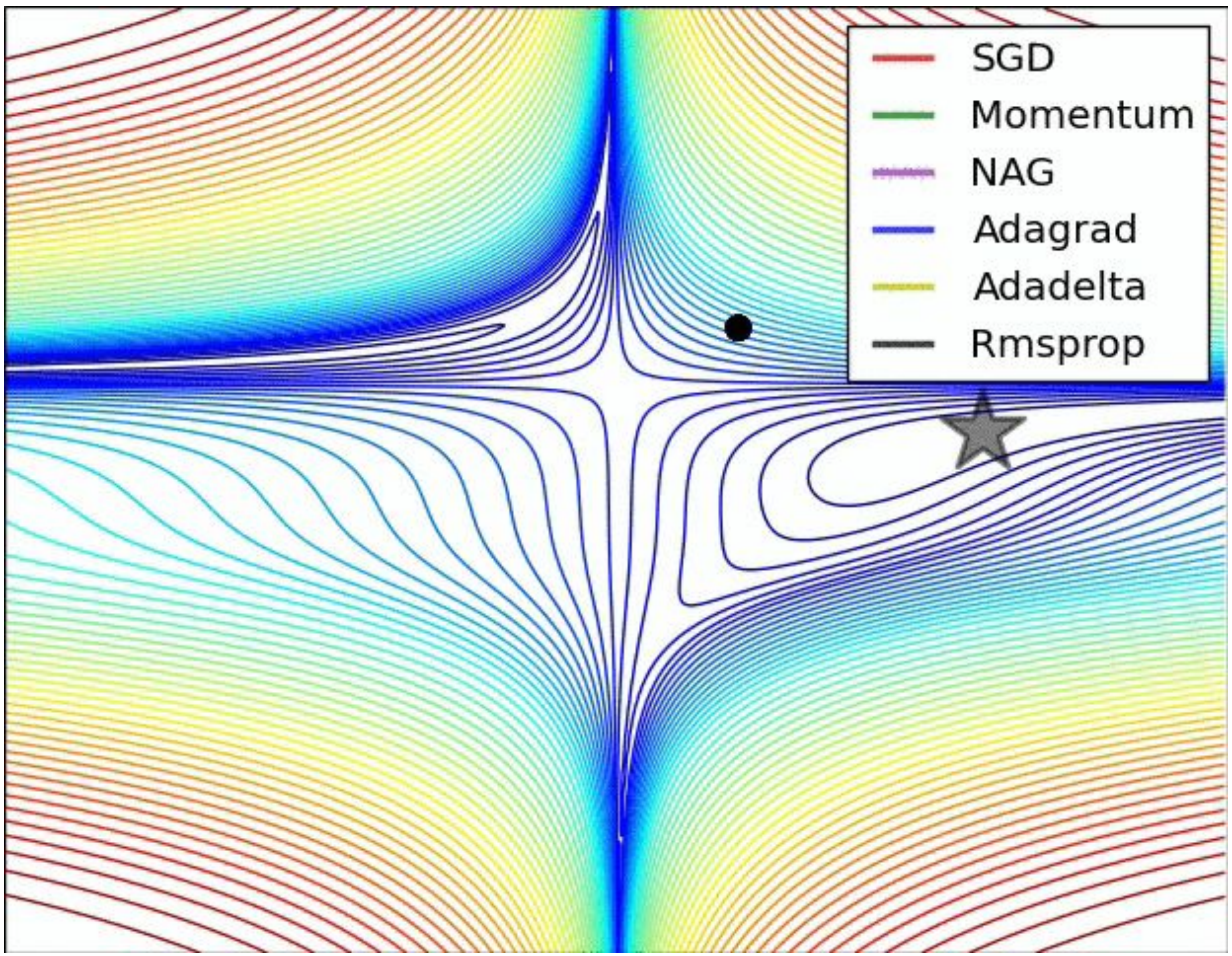
How does decay rate and error floor change with

- η (Learning Rate) ?
- M (Second moment of gradient) ?

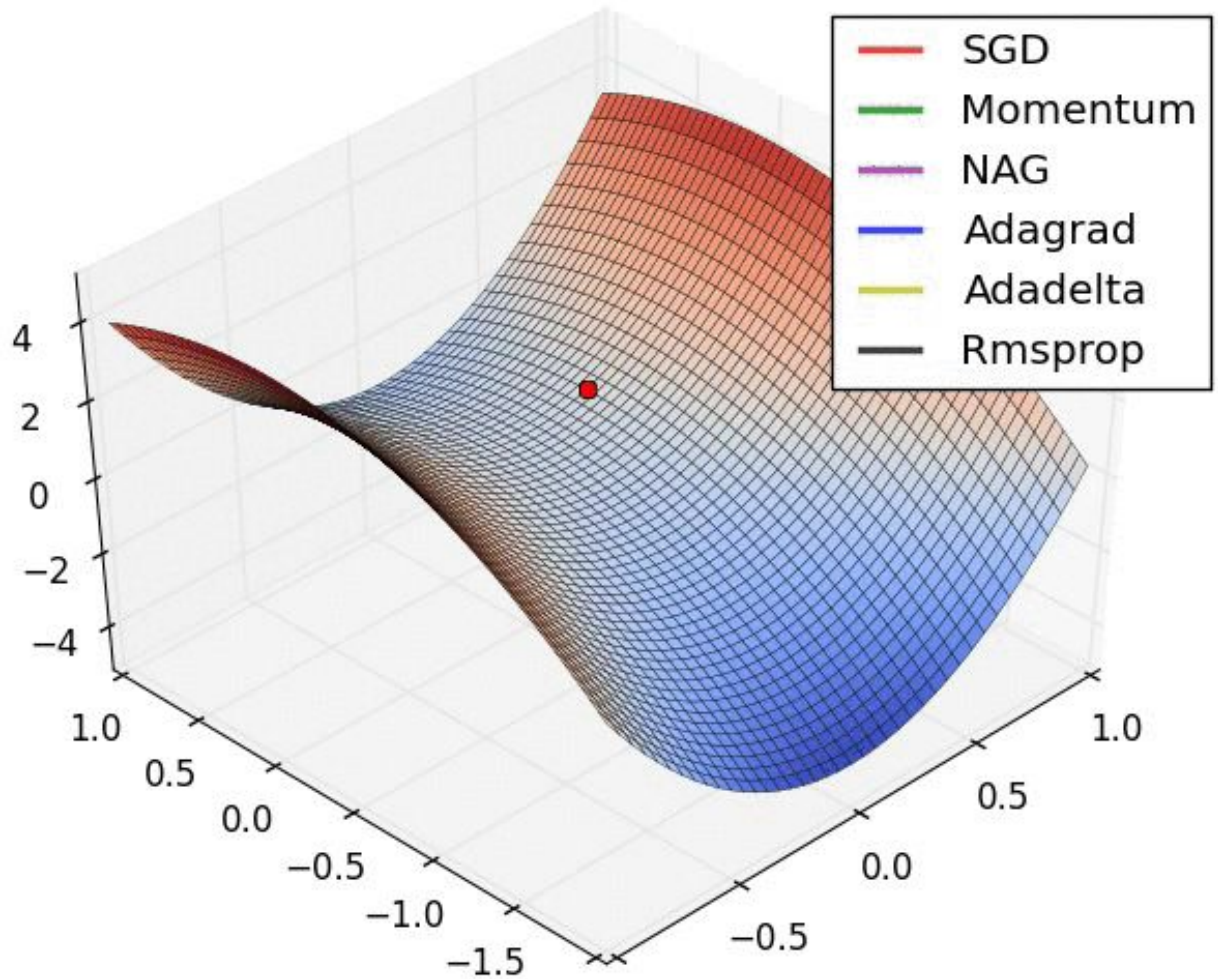
Many other variants of SGD

- Momentum SGD
- Nesterov Momentum
- AdaGrad
- Adam
- AdaDelta
- RMS prop

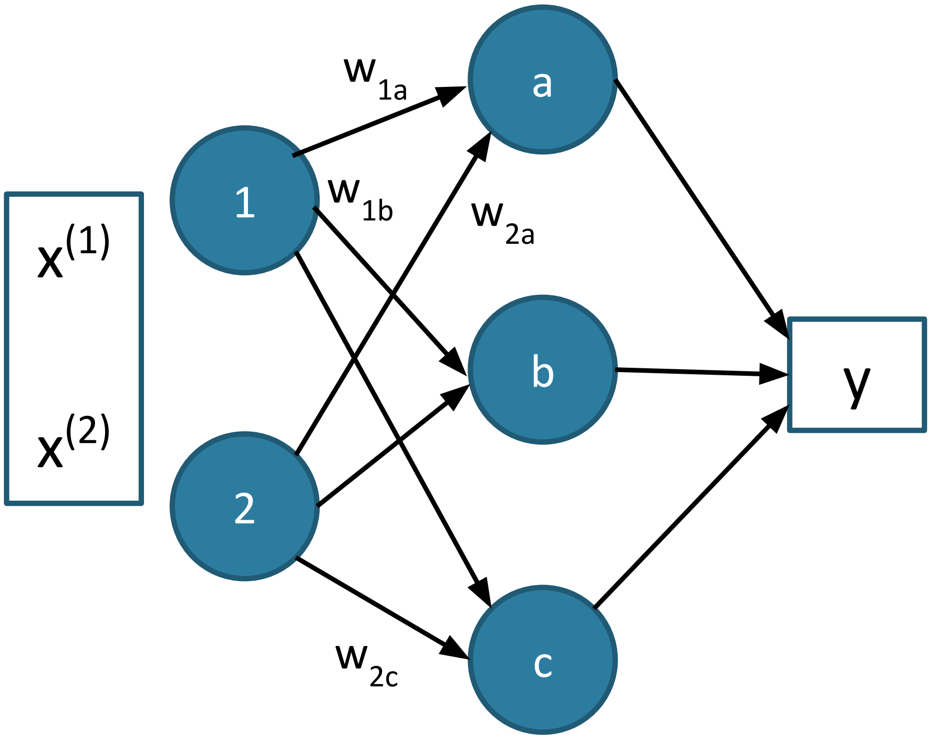
Many other variants of SGD



Many other variants of SGD



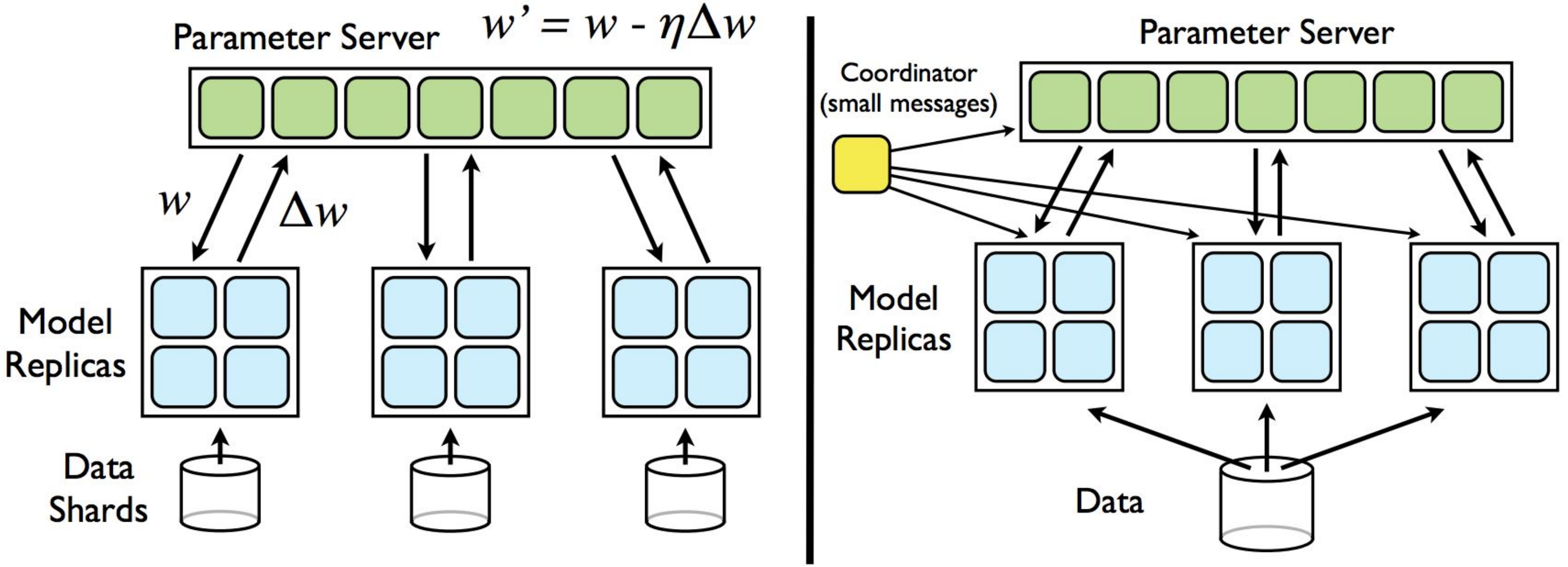
SGD and Backpropagation



Given a big dataset of $(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), (\mathbf{x_3}, y_3), (\mathbf{x_4}, y_4), \ldots.(\mathbf{x_N}, y_N)$
Find the optimal weights **w**

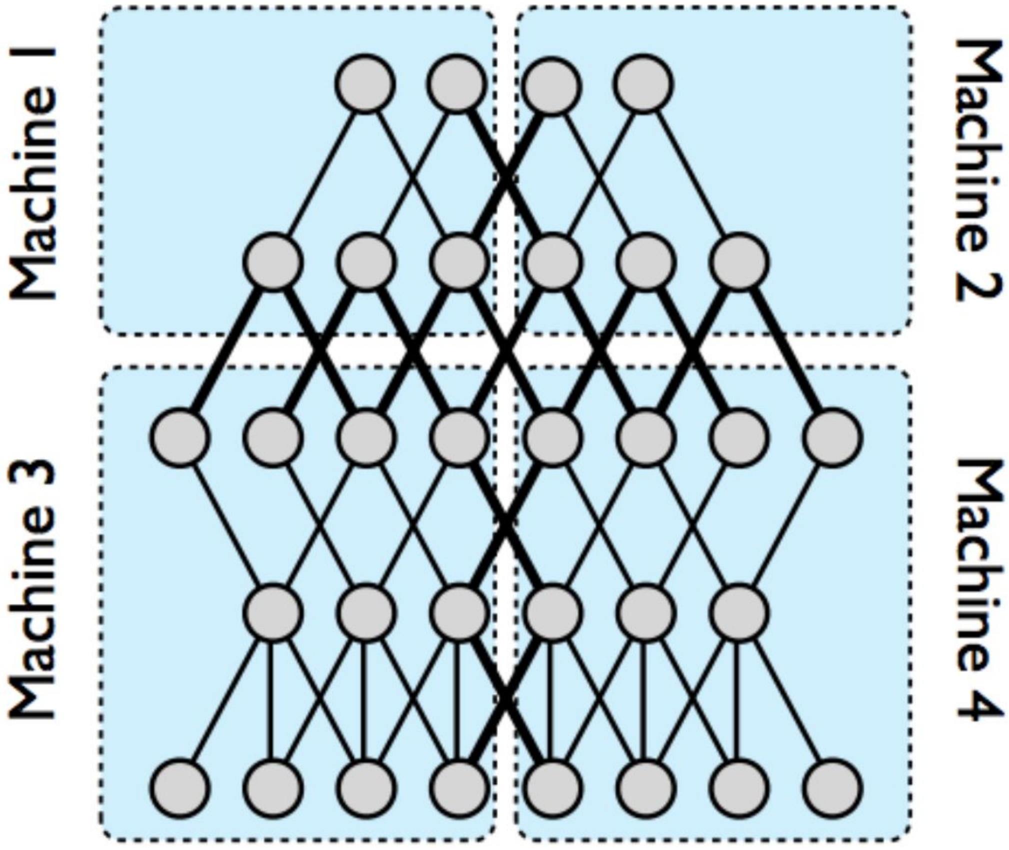
Distributed Deep Learning

Data Parallelism



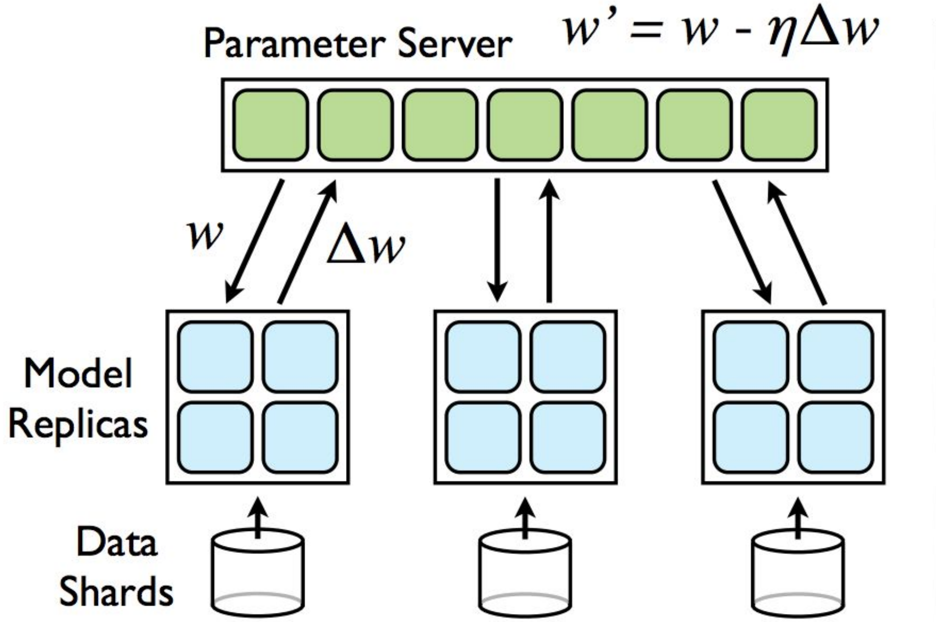
Distributed Deep Learning

Model Parallelism



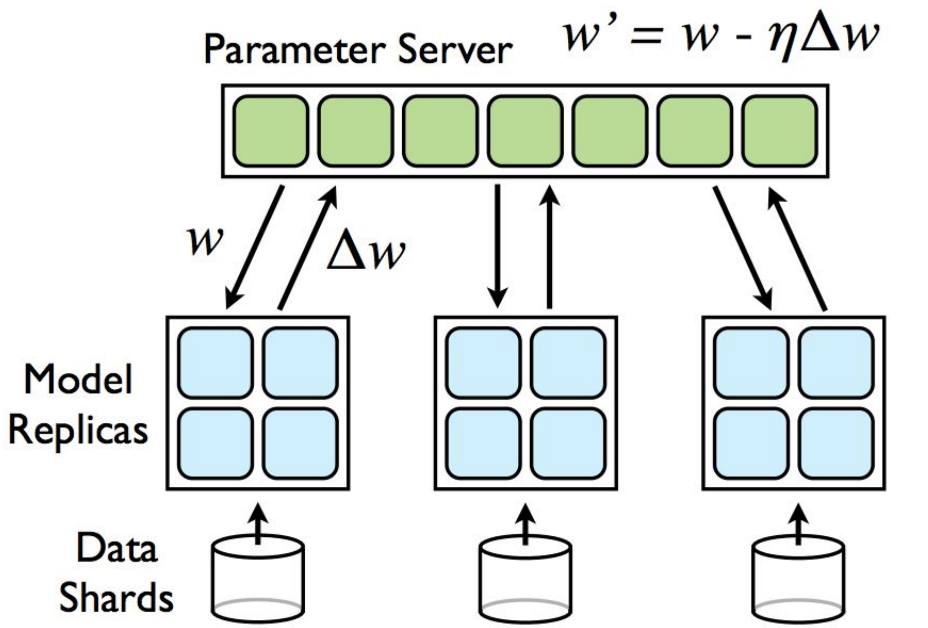
Synchronous SGD

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \sum_{k=1}^{K} \frac{1}{K} \nabla F(\mathbf{w}_t, \xi_k)$$



Q: What is the convergence rate and error floor?

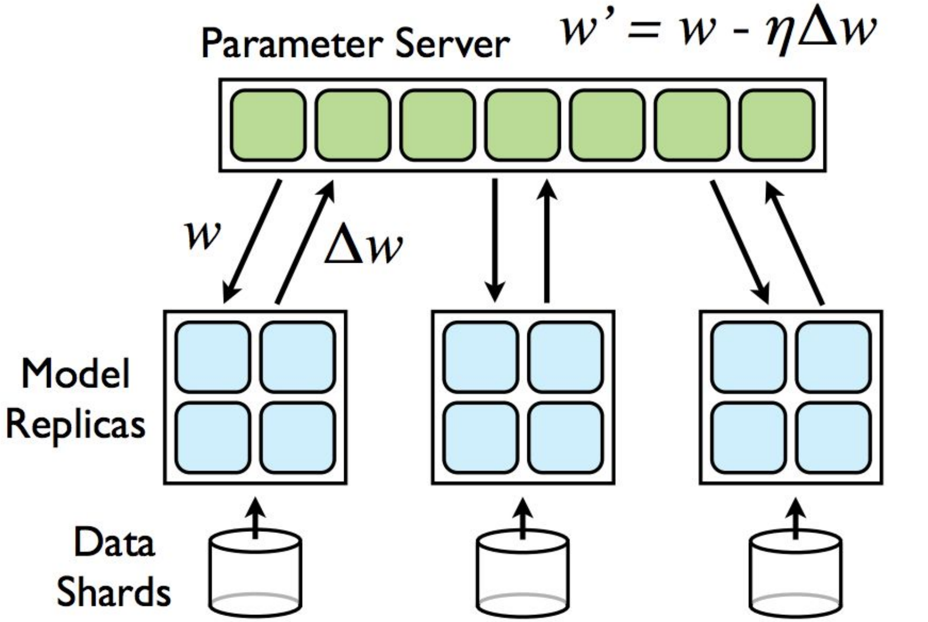
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \sum_{k=1}^{K} \frac{1}{K} \nabla F(\mathbf{w}_t, \xi_k)$$

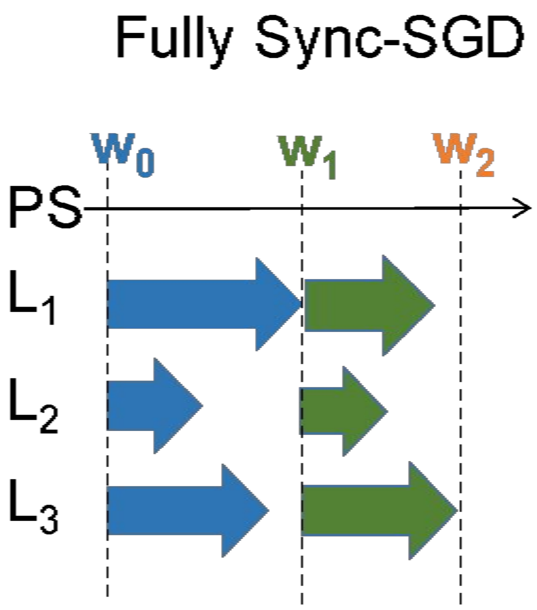


Q: What is the time to complete each iteration?

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

Slowest worker is the bottleneck

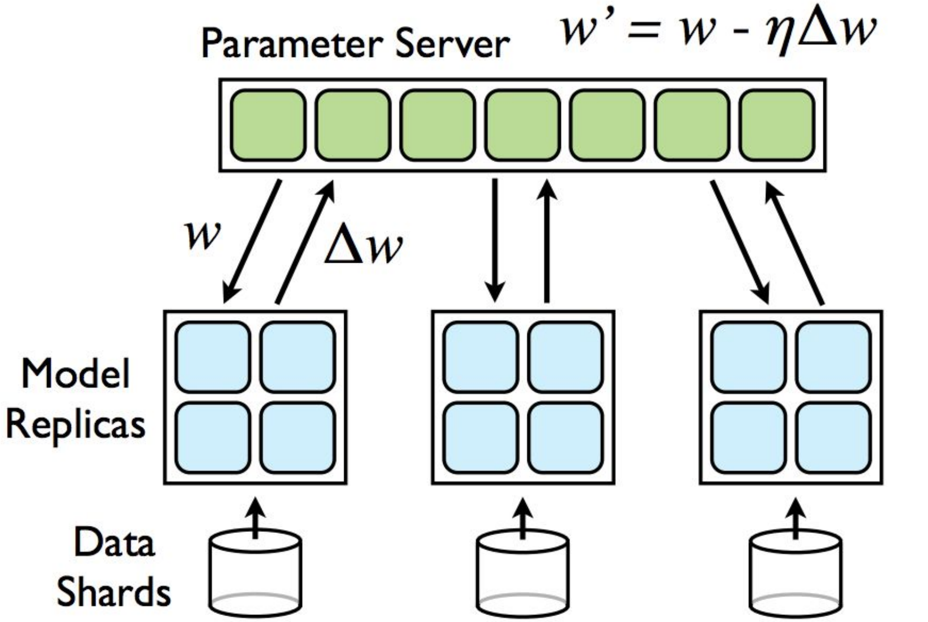


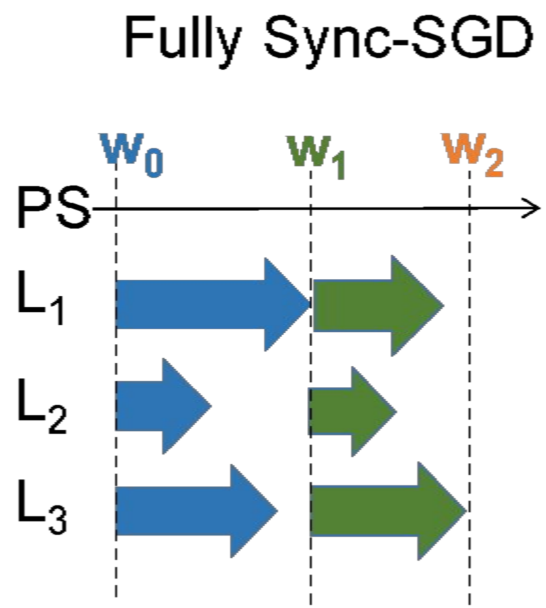


Q: How can we reduce it?

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

Slowest Worker is the bottleneck

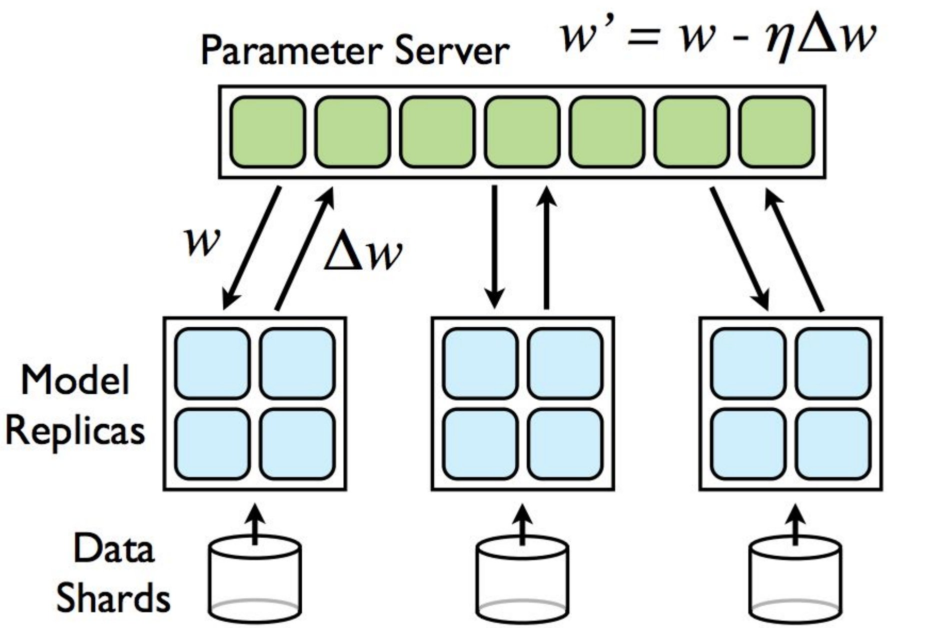


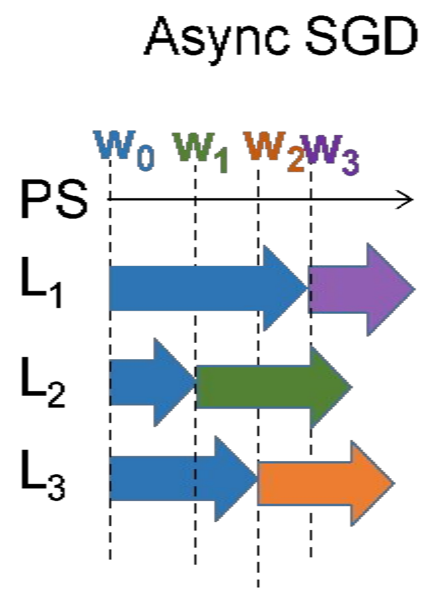


Asynchronous SGD: Don’t wait for all

Asynchronous SGD cuts the latency tail.

But, what effect does it have on the error?





Variants of Distributed SGD

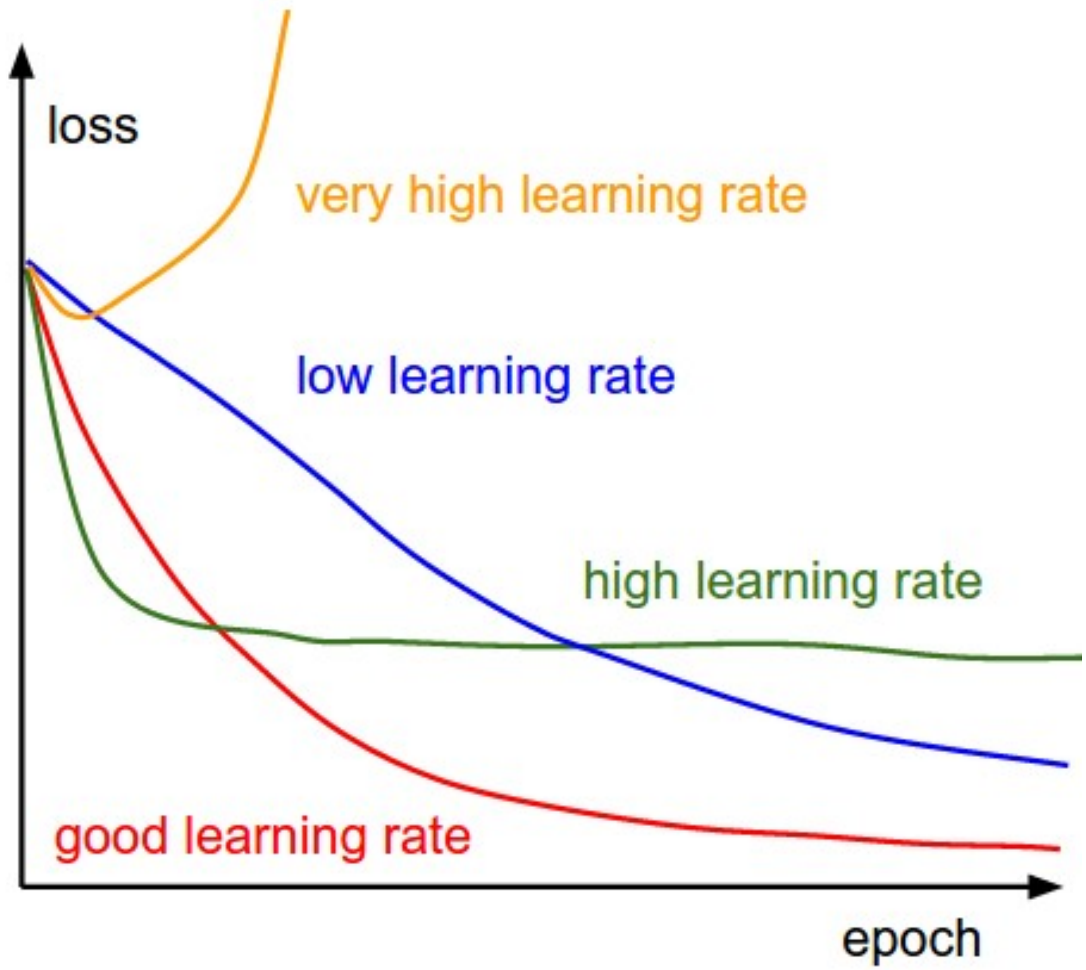
- Synchronous SGD
- Asynchronous SGD
- HogWild
- Elastic-Averaging SGD
- Federated Learning
- Gradient Compression/Quantization

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