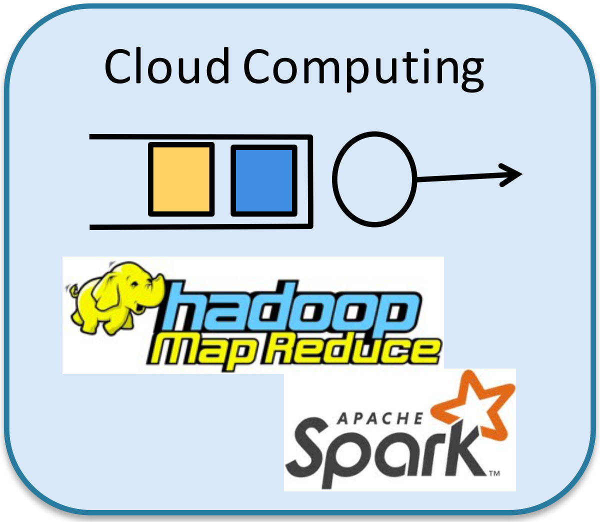
18-847F: Special Topics in Computer Systems

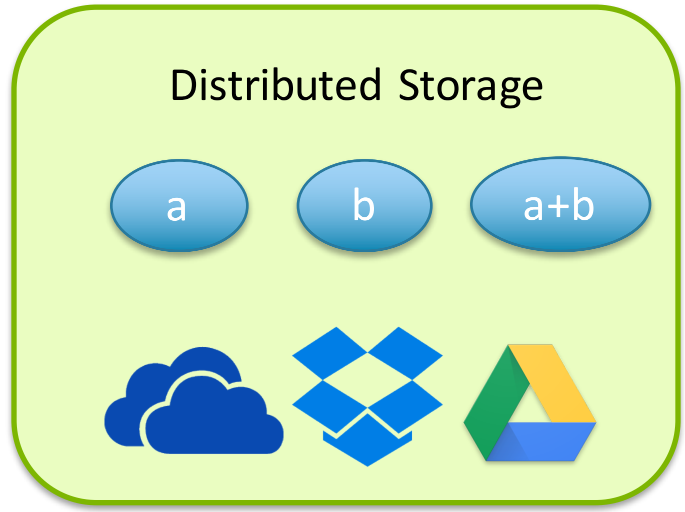
Foundations of Cloud and Machine Learning Infrastructure

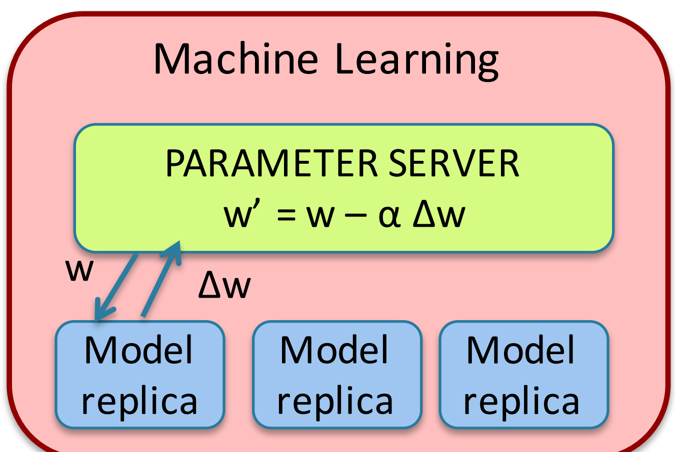
Lecture 3: Overview of ML Infrastructure

Foundations of Cloud and Machine Learning Infrastructure

Topics Covered

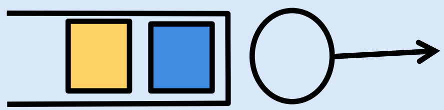




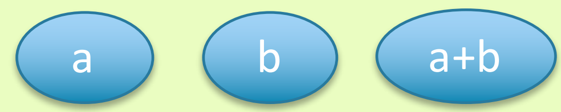


Let us recap what we learnt..

Cloud Computing

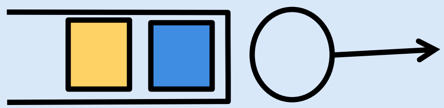


Distributed Storage



Let us recap what we learnt..

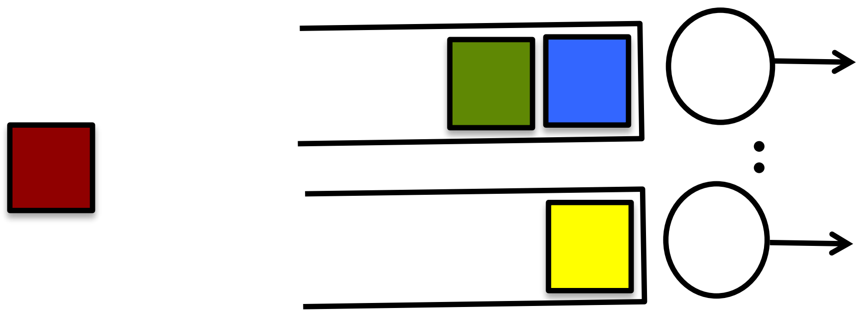
Cloud Computing



- MapReduce, Spark
- Scheduling in Parallel Computing
- Straggler Replication
- Task Replication in Queueing Systems

Scheduling in Parallel Computing: 1990's

- **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

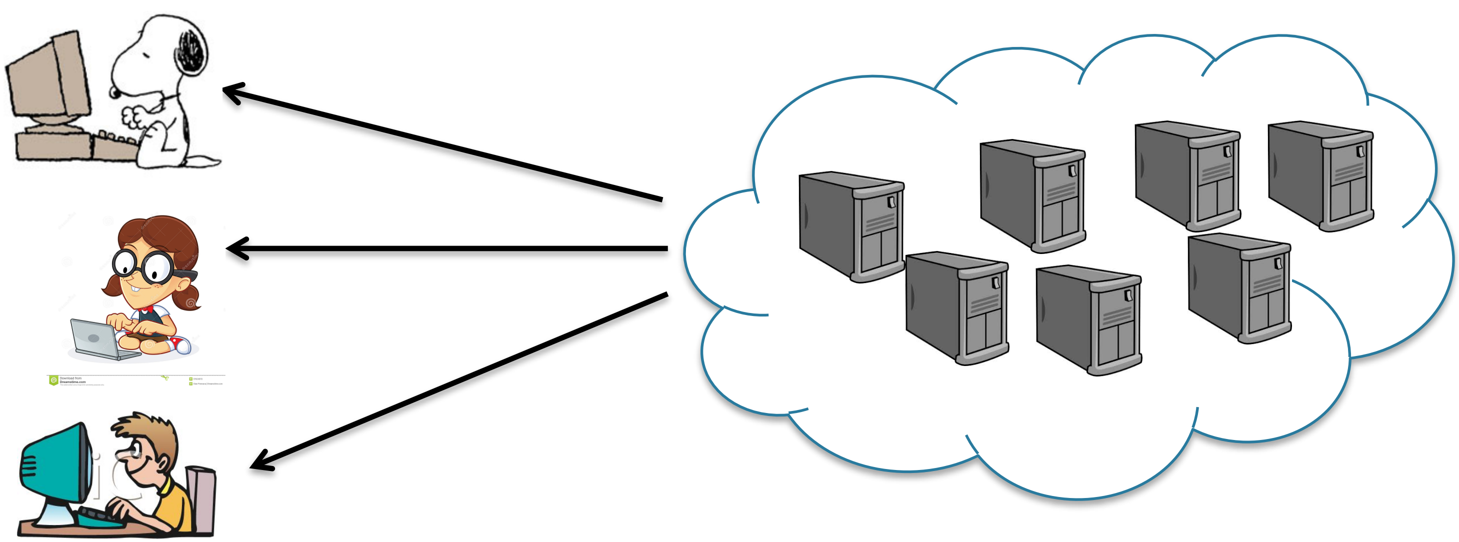


2000's: The Cloud Computing Era

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]



Cloud Frameworks

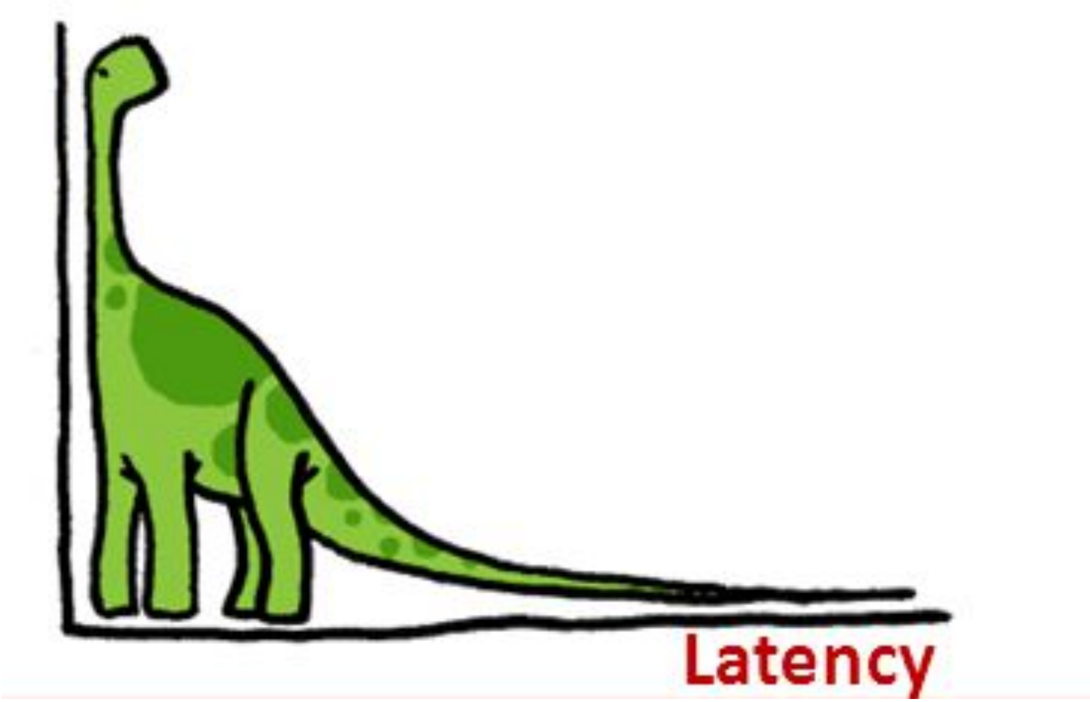
MapReduce

Spark: In-memory

Sparrow: Low-Latency Cluster Scheduling

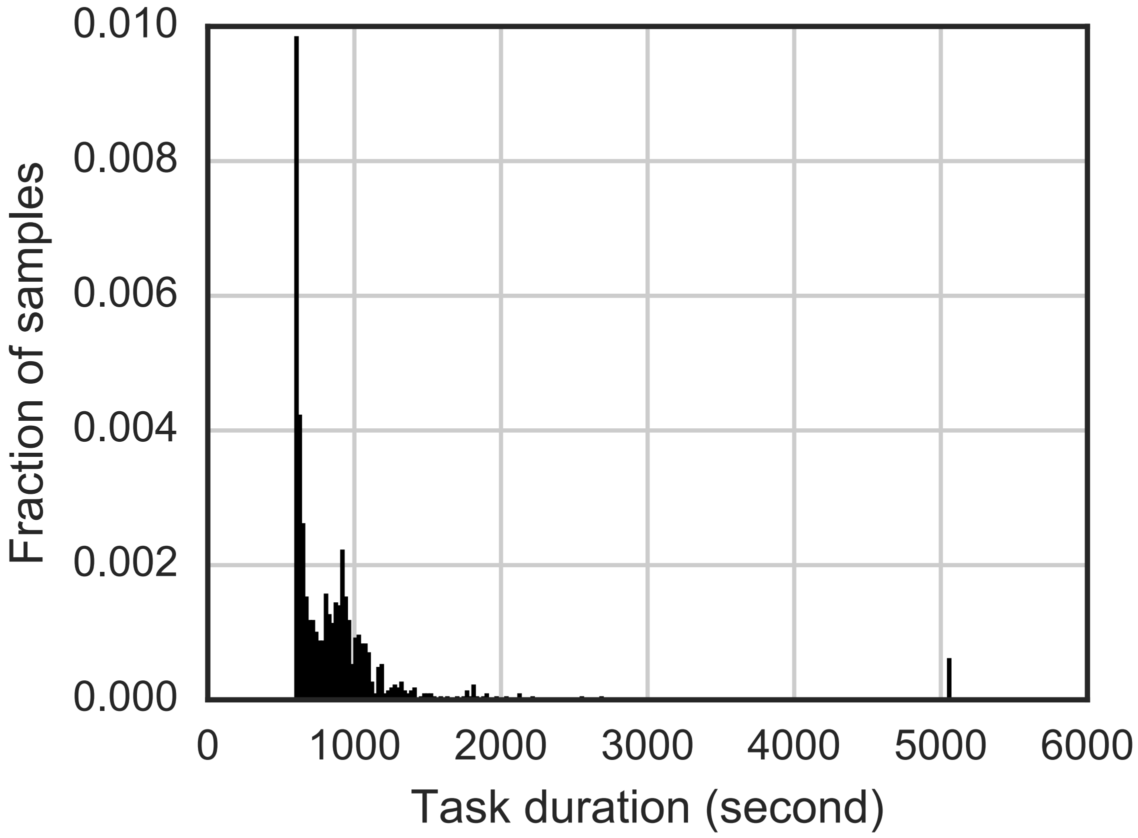
Dolly: Attack of the Clones

The Tale of Tails



Tail at Scale: 99%ile latency can be much higher than average

The Tale of Tails

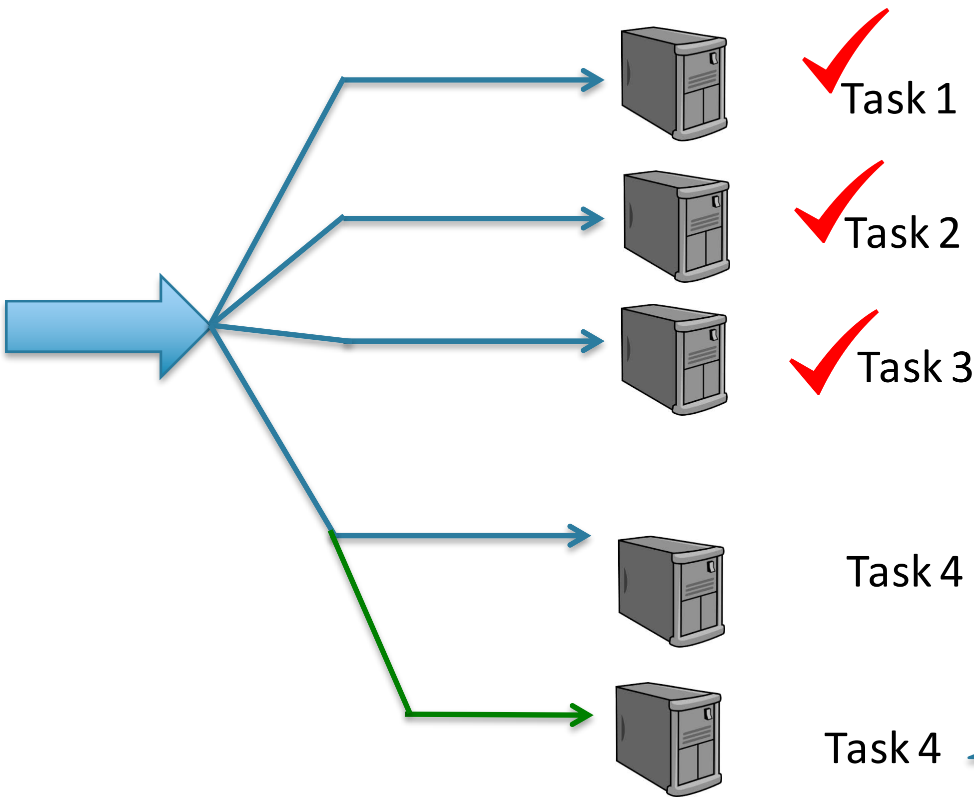


Tail at Scale: 99%ile latency much higher than average

Straggler Replication

PROBLEM: Slowest tasks become a bottleneck

SOLUTION: Replicate the stragglers and wait for one copy



PARAMETERS

p: Frac. of tasks replicated

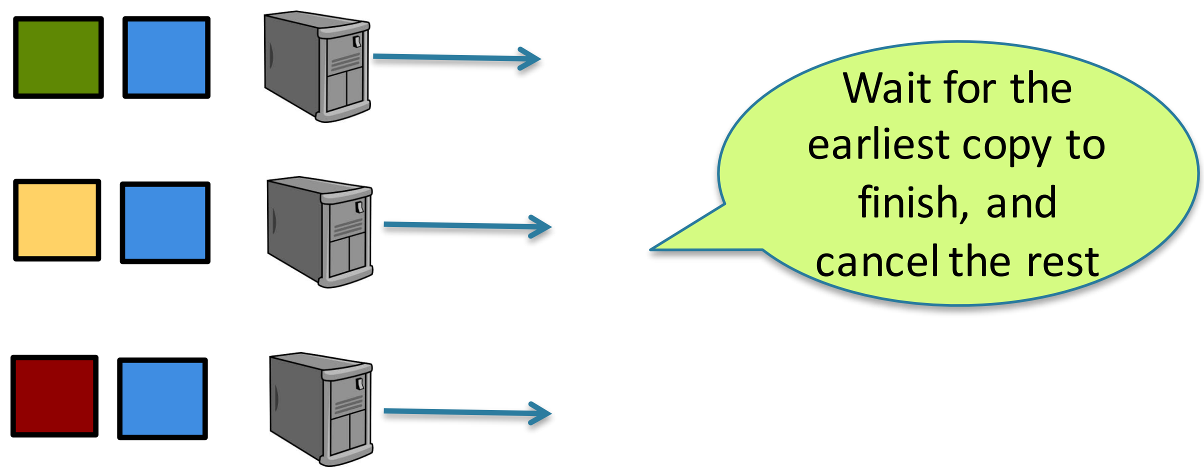
r: # additional replicas

c: kill/keep original task

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

Task Replication in Cloud Computing

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

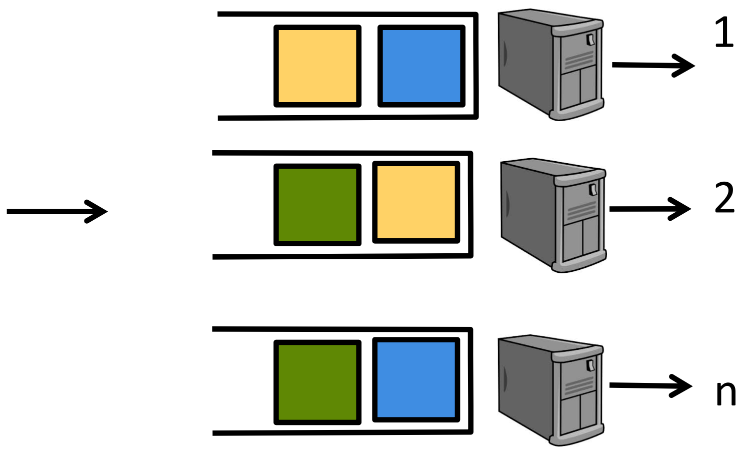


COST

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

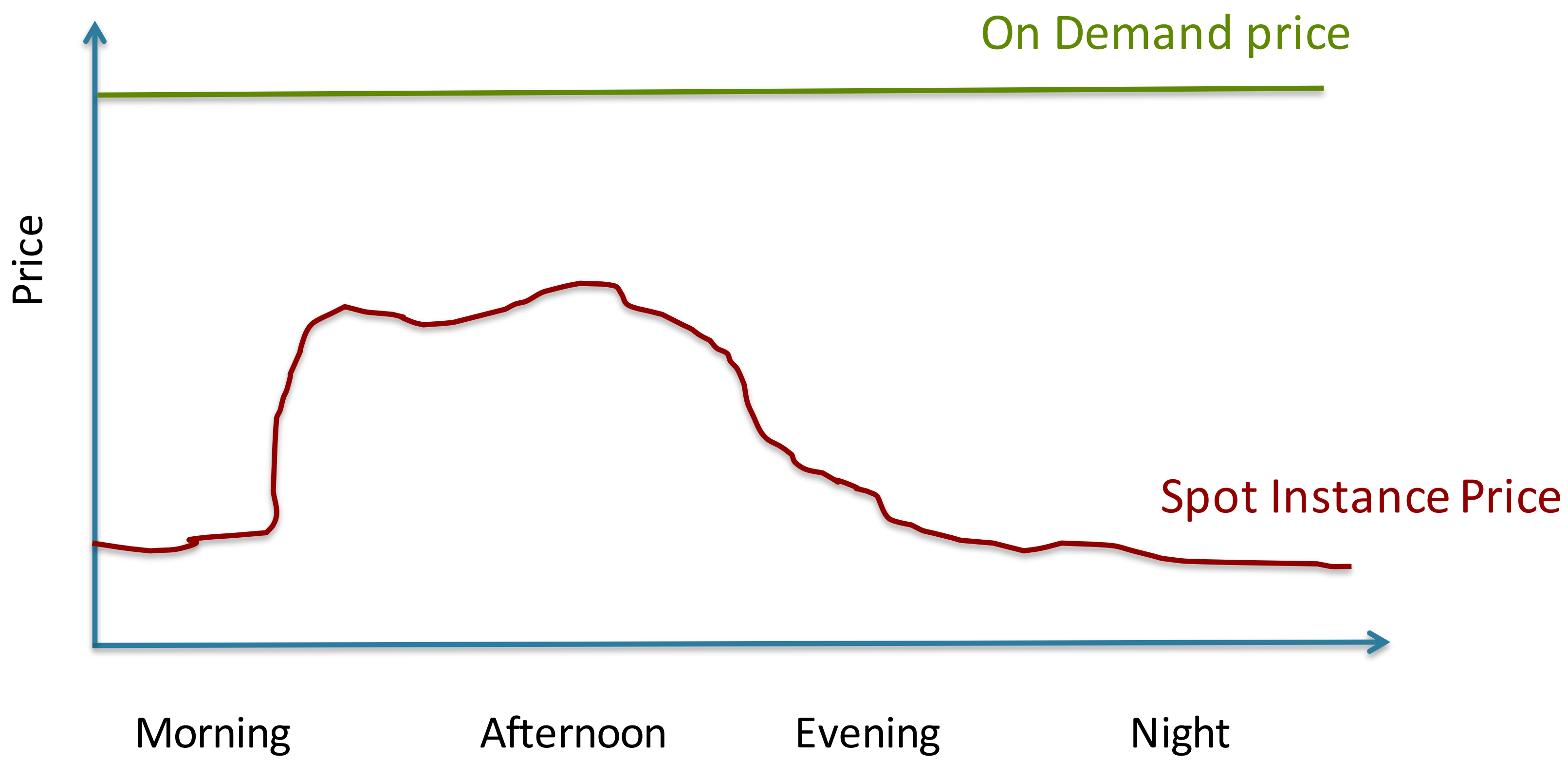
Design Questions

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



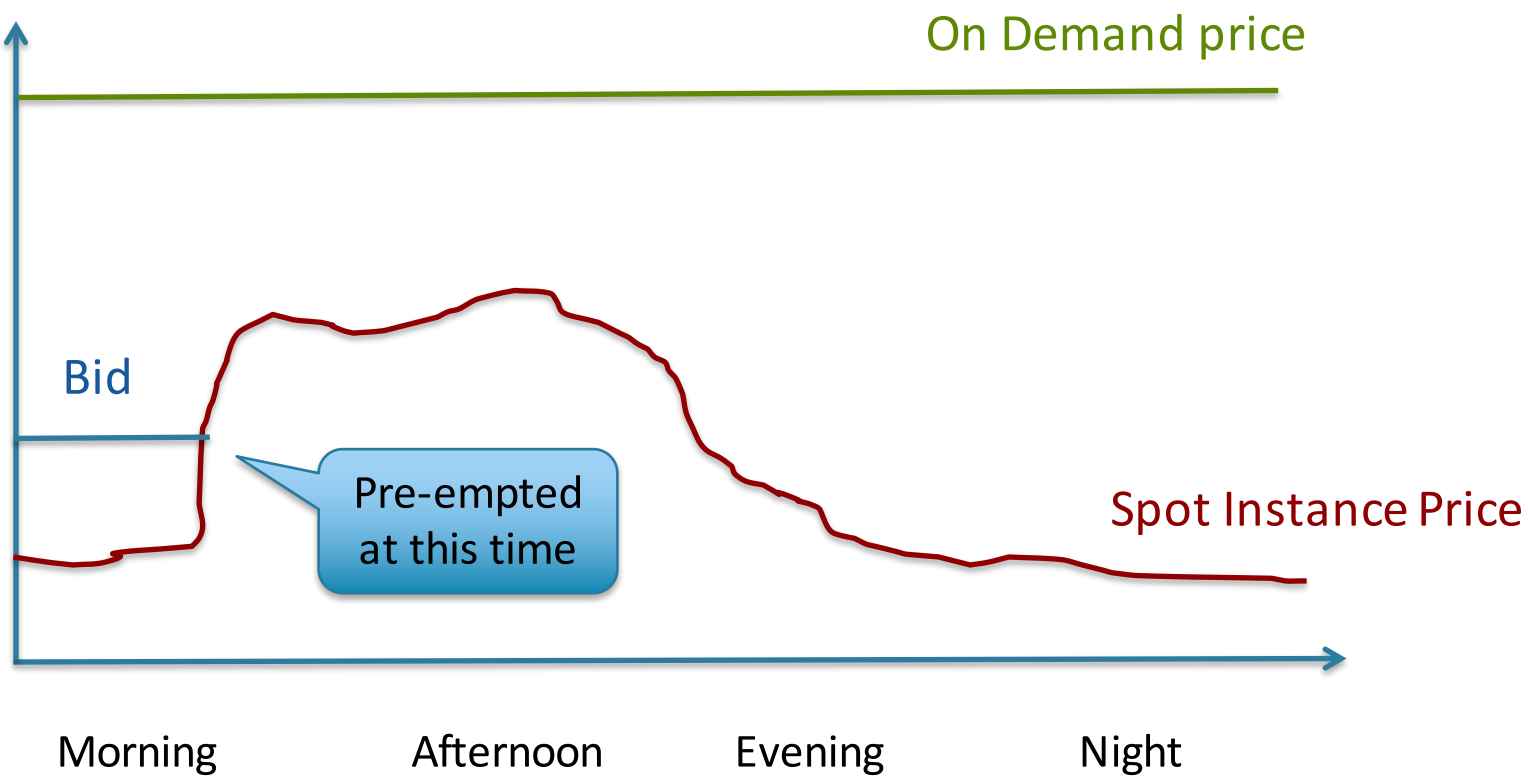
Cloud Spot Markets

- Sell it on the spot market for a lower price!



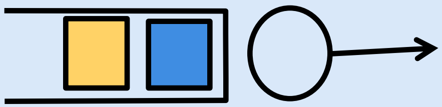
Guest Lecture: Prof. Carlee Joe-Wong

- Bidding and pricing strategies for spot markets



Let us recap what we learnt..

Cloud Computing



Distributed Storage

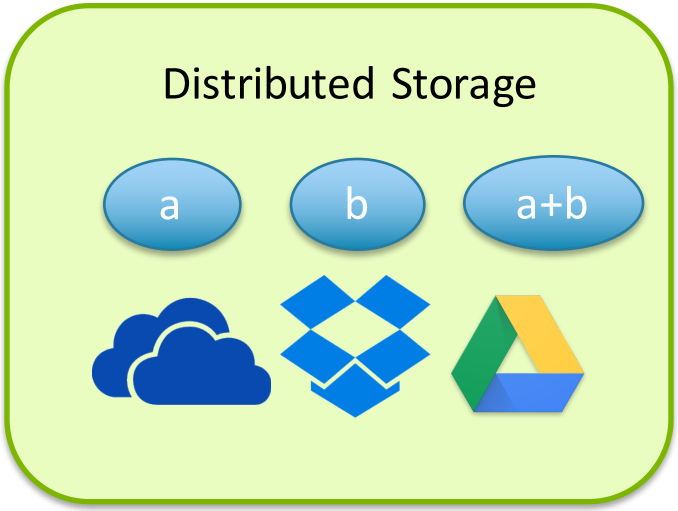
a

b

a+b

Let us recap what we learnt..

- RAID systems
- Coding for locality/repair
- Systems implementation of codes
- Reducing latency in content download



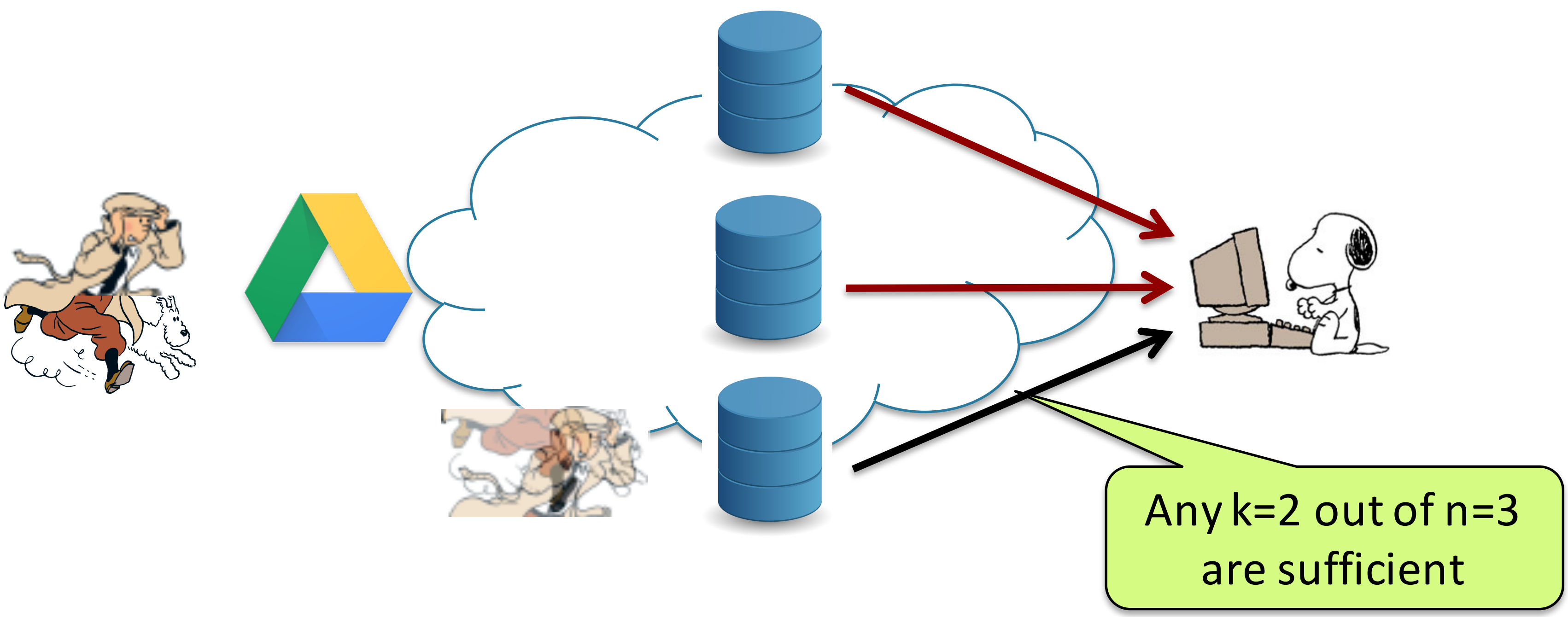
RAID: Redundant Array of Independent Disks (1987)

- Levels RAID 0, RAID 1, ... : design for different goals such as reliability, availability, capacity etc.

- One of the inventors, Garth Gibson is at CMU

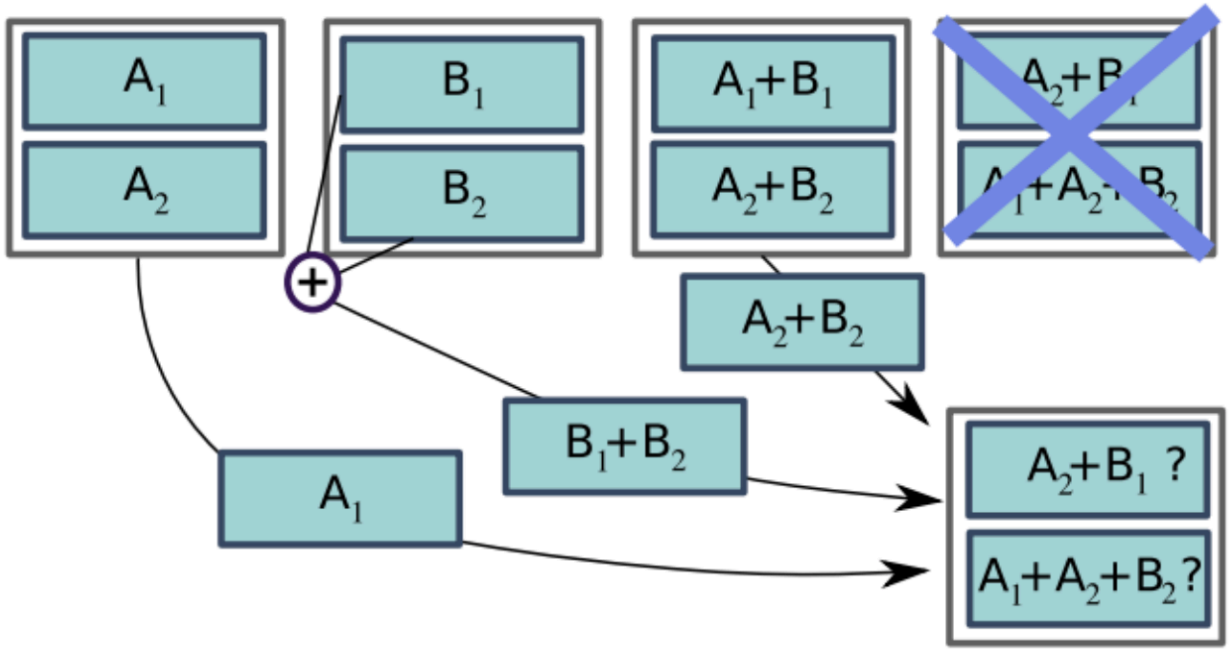
Erasure Coded Storage

- 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



Codes for Efficient Repair

- Exact repair
- Functional repair



Guest Lecture: Prof. Rashmi Vinayak

Hitchhiker Codes and EC-Cache

	An MDS Code		Intermediate Step		Piggybacked Code	
~~Node 1~~	~~a_1~~	~~b_1~~	~~a_1~~	~~b_1~~	~~a_1~~	~~b_1~~
Node 2	a_2	b_2	a_2	b_2	a_2	b_2
Node 3	a_3	b_3	a_3	b_3	a_3	b_3
Node 4	a_4	b_4	a_4	b_4	a_4	b_4
Node 5	$\sum_{i=1}^4 a_i$	$\sum_{i=1}^4 b_i$	$\sum_{i=1}^4 a_i$	$\sum_{i=1}^4 b_i$	$\sum_{i=1}^4 a_i$	$\sum_{i=1}^4 b_i$
Node 6	$\sum_{i=1}^4 ia_i$	$\sum_{i=1}^4 ib_i$	$\sum_{i=1}^4 ia_i$	$\sum_{i=1}^4 ib_i + \sum_{i=1}^2 ia_i$	$\sum_{i=3}^4 ia_i - \sum_{i=1}^4 ib_i$	$\sum_{i=1}^4 ib_i + \sum_{i=1}^2 ia_i$

(a) (b) (c)

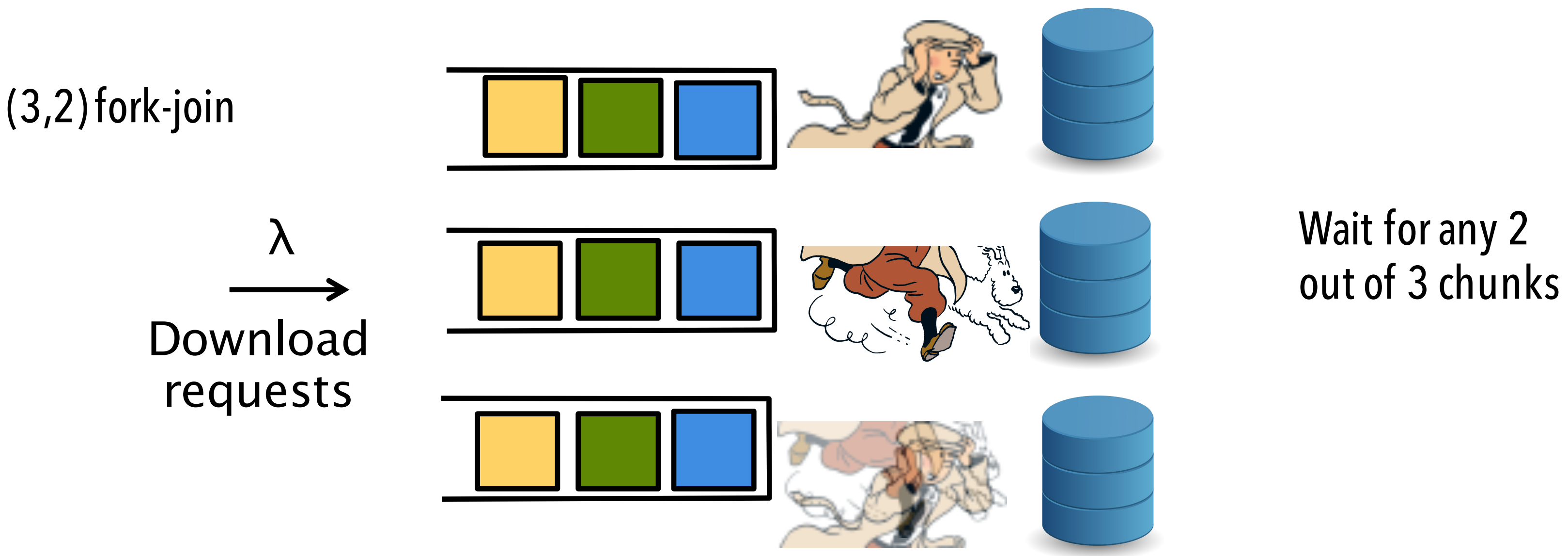
Needs 8 symbols to repair

Needs 6 symbols to repair

The (n,k) fork-join model

[GJ-Liu-Soljanin 2012,14]

- Request all n chunks, wait for any k to be downloaded
- Each chunk takes service time $X \sim F_X$

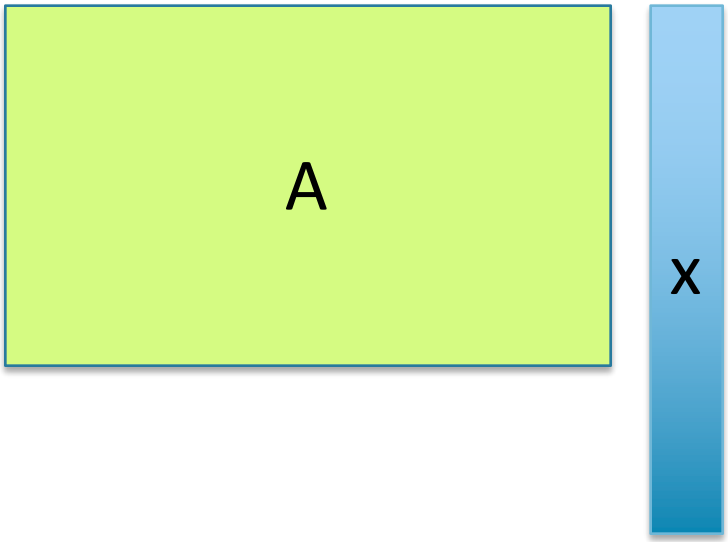


k = 1: Replicated Case

k = n: Fork-join system actively studied in 90's

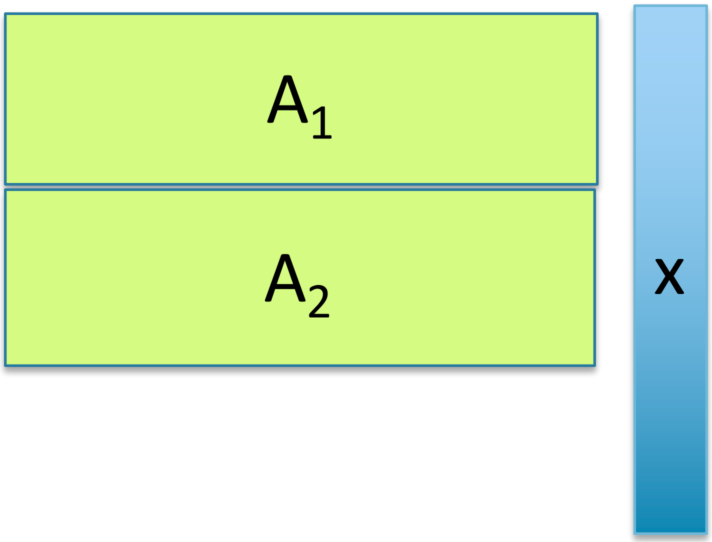
Coded Computing and ML

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



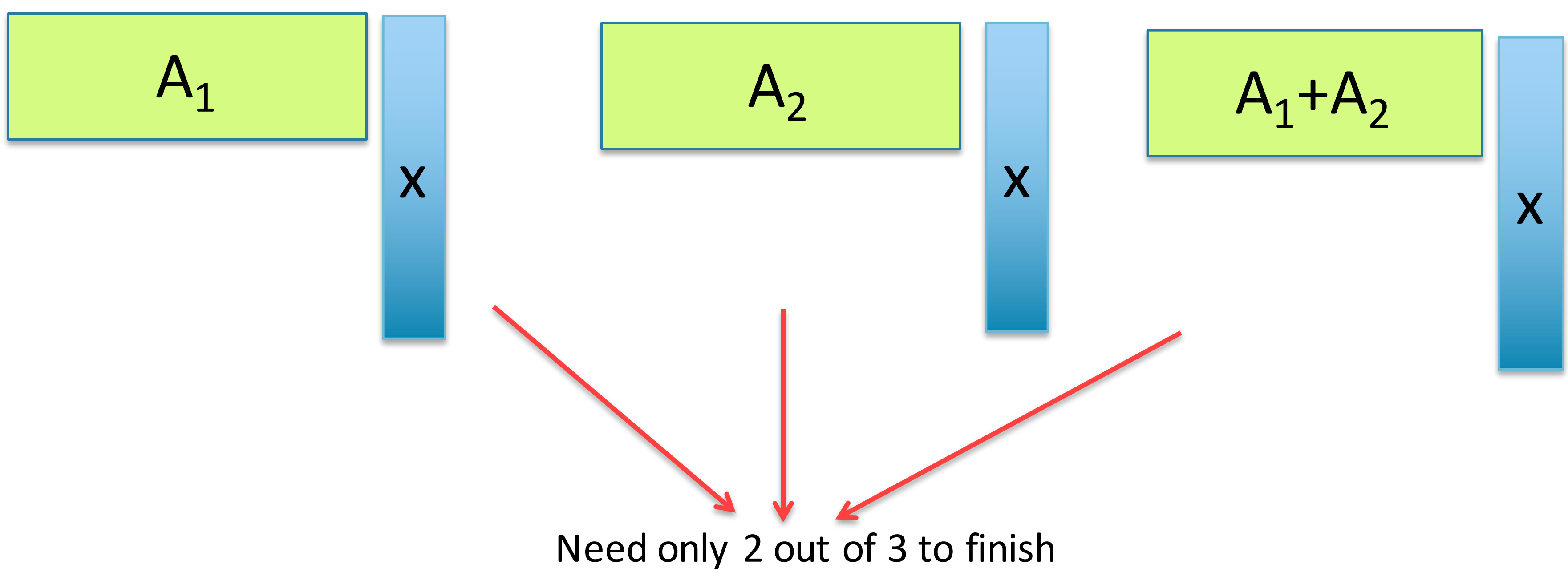
Coded Computing and ML

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

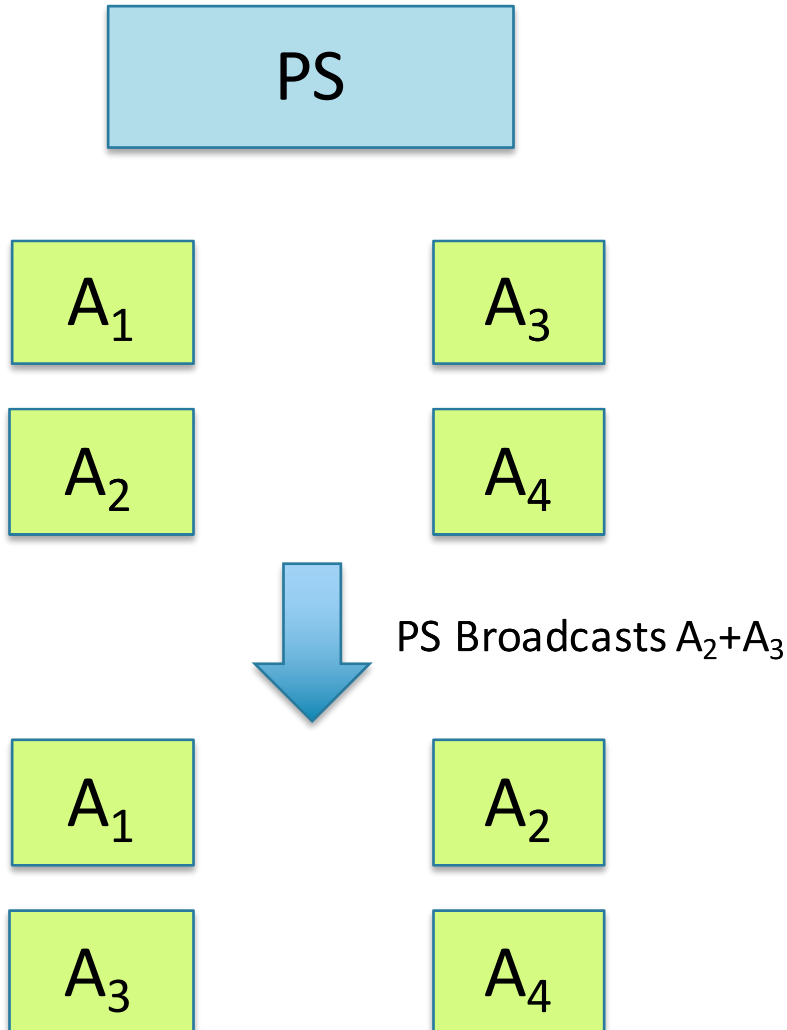


Coded Computing and ML

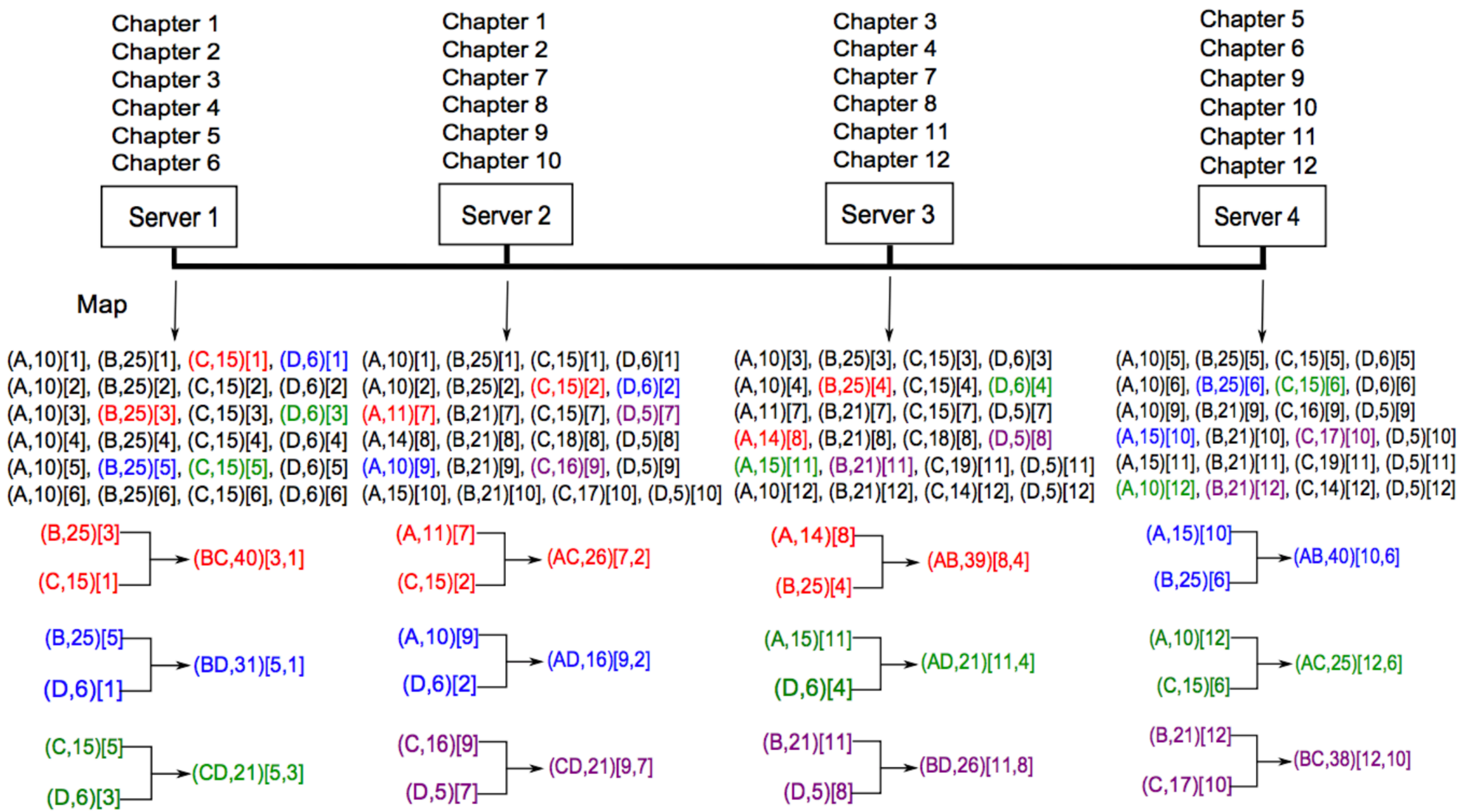
- So far: coding for storage
- Codes can also speed up computing and machine learning!
- Example: Matrix-Vector Multiplication



Coded Data Shuffling



Coded MapReduce



Guest Lecture: Sanghamitra Dutta

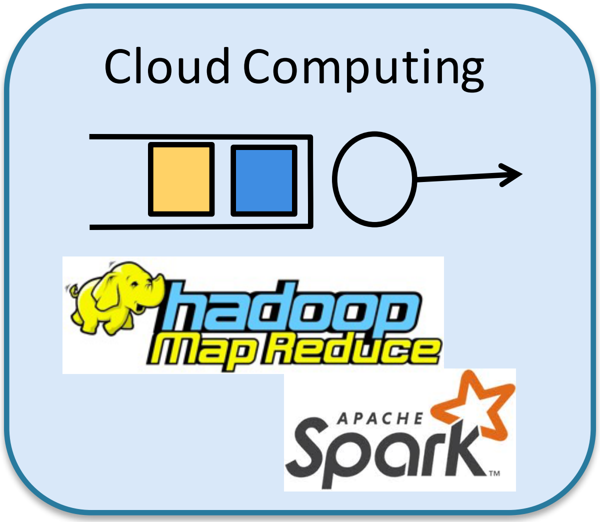
Short-dot codes

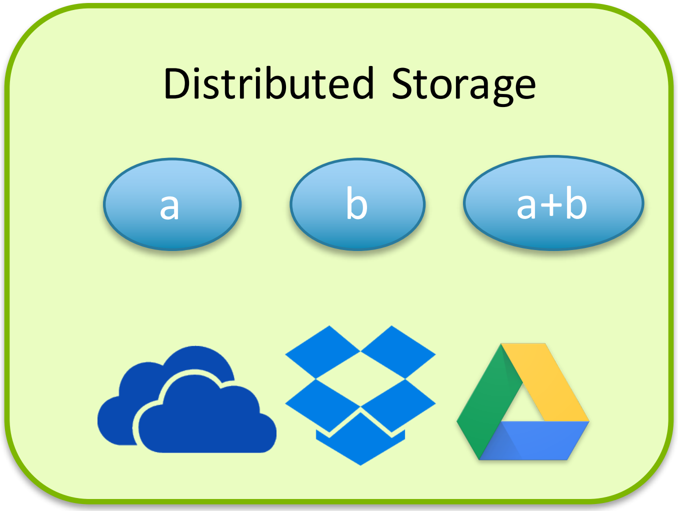
(*P*,*M*) MDS Code

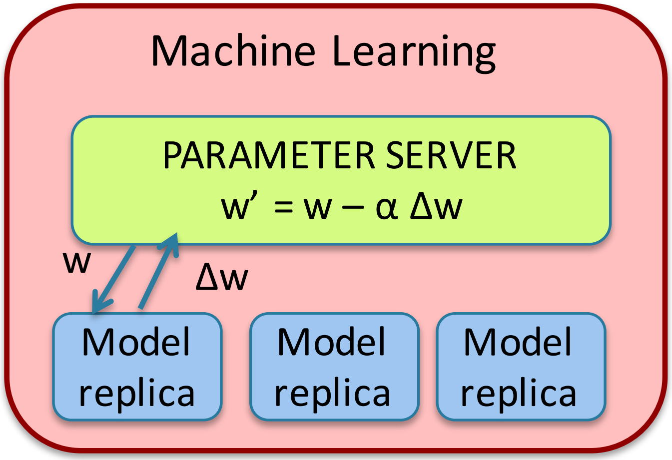
P **dense dot-products of length *N* in *P* parallel processors**

Wait for any *M* computations to finish

Last Module: Machine Learning

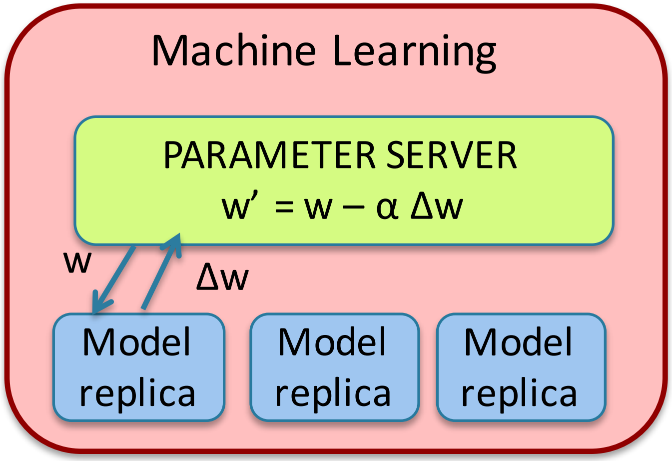




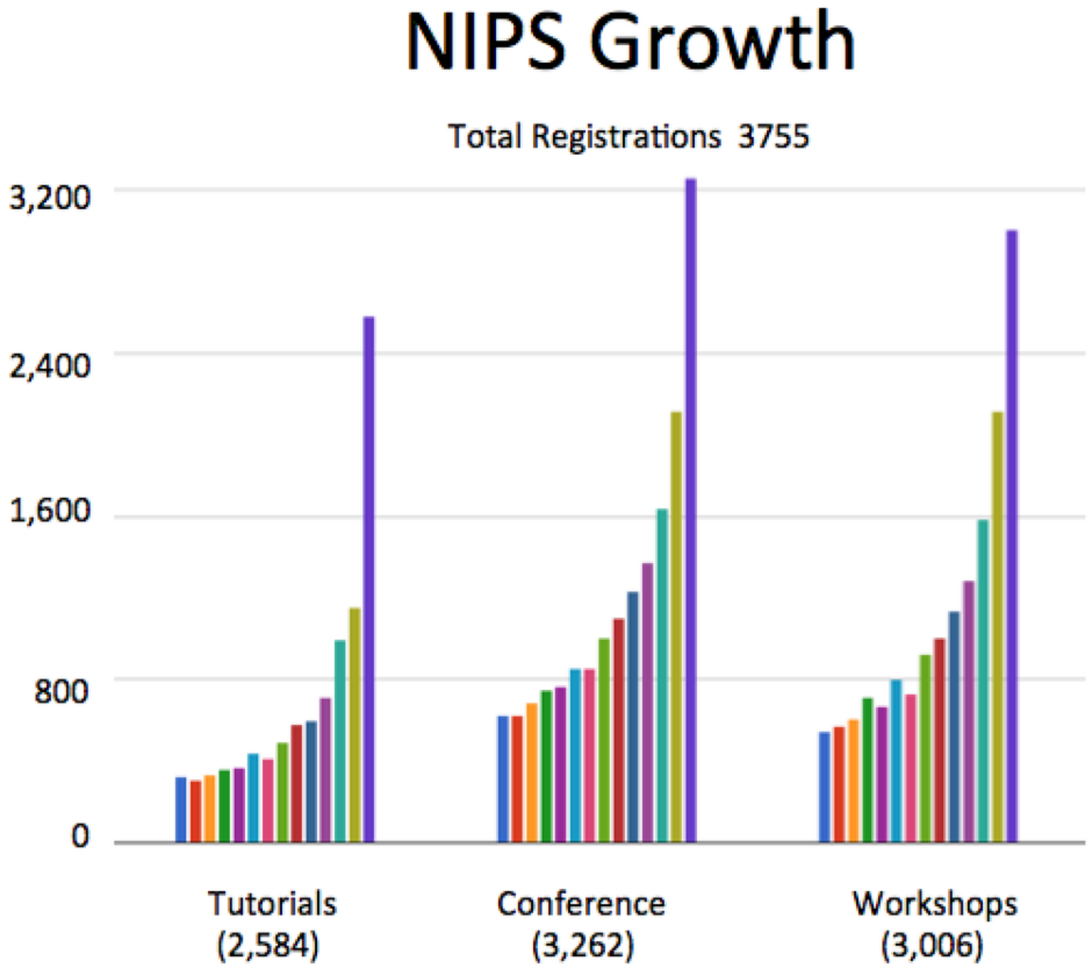


Last Module: Machine Learning

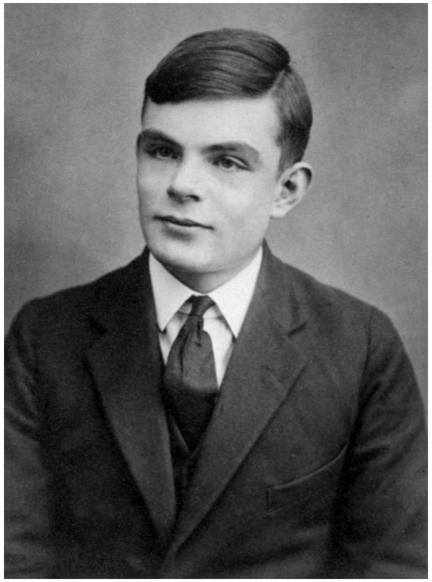
- SGD Methods, Convergence
- DistBelief, Alexnet
- Synchronous, Asynchronous SGD
- GANs, Reinforcement Learning



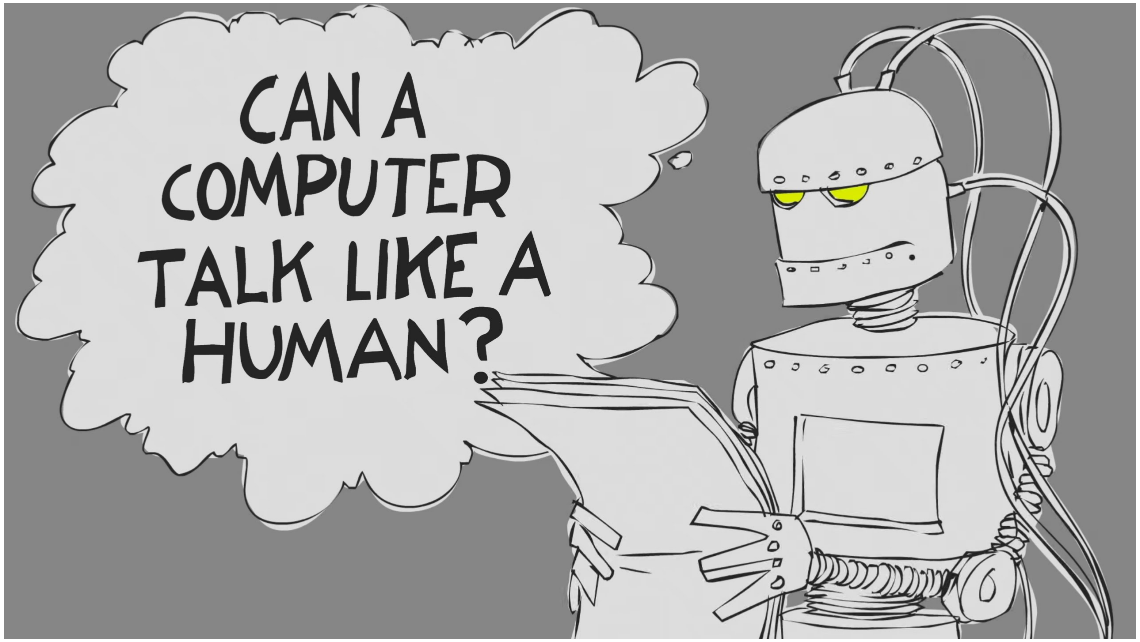
The unprecedented ML boom



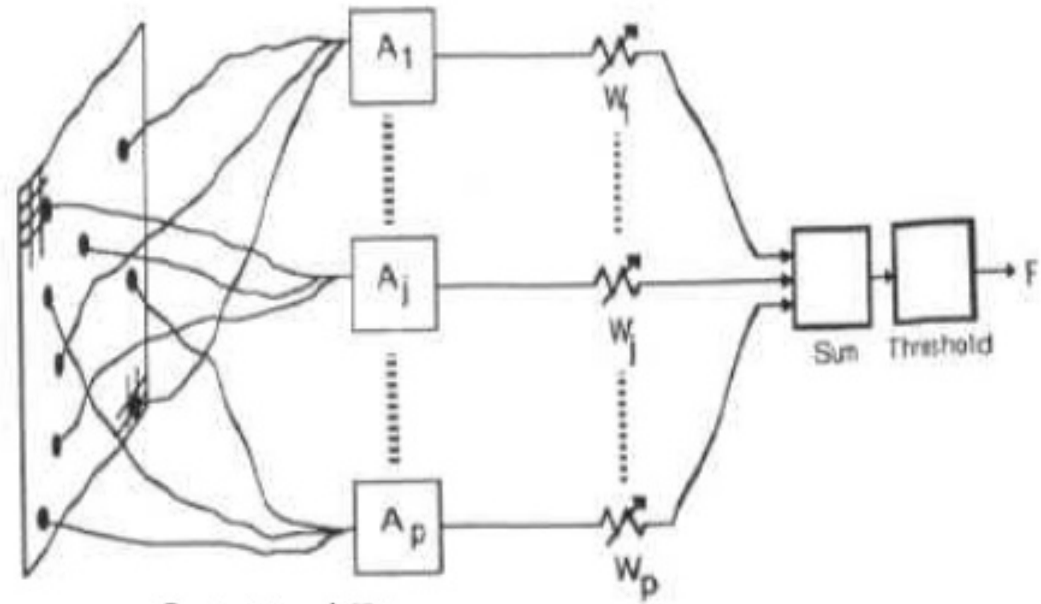
The Origins: 1950



Alan Turing



Neural Networks: Perceptron 1957

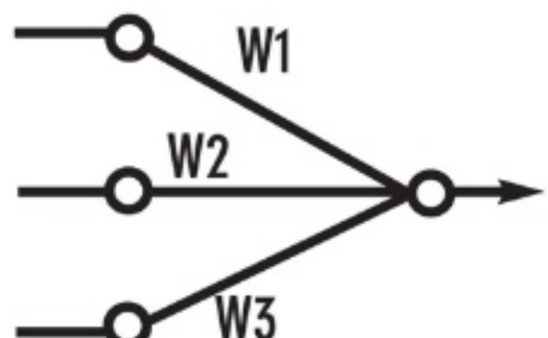


Original Perceptron

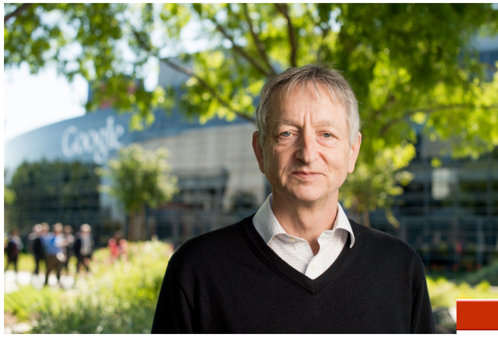
(From Perceptrons by M. L Minsky and S. Papert, 1969, Cambridge, MA: MIT Press. Copyright 1969 by MIT Press.

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[*]

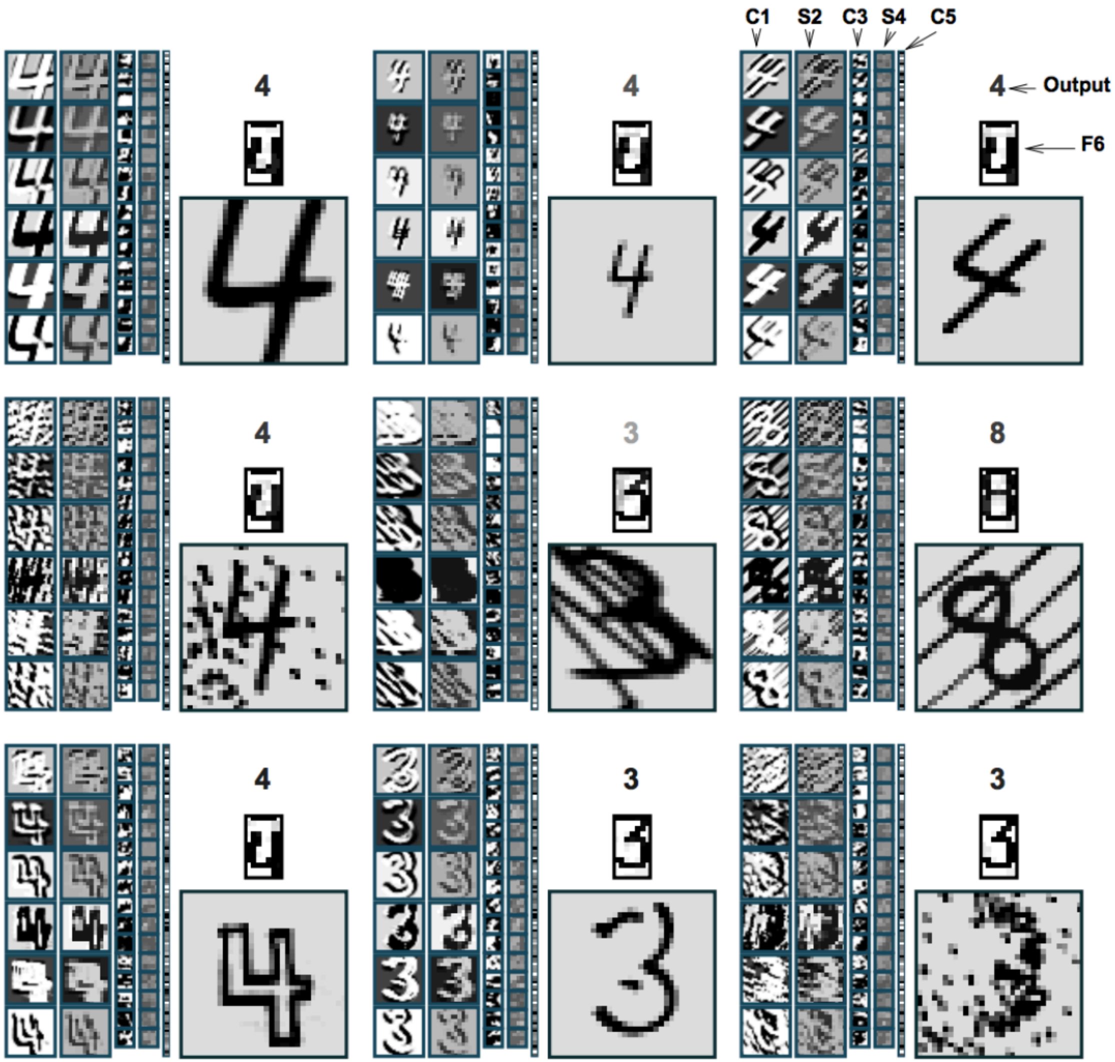
[*]Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
[†]Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA
[†]To whom correspondence should be addressed.

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].**

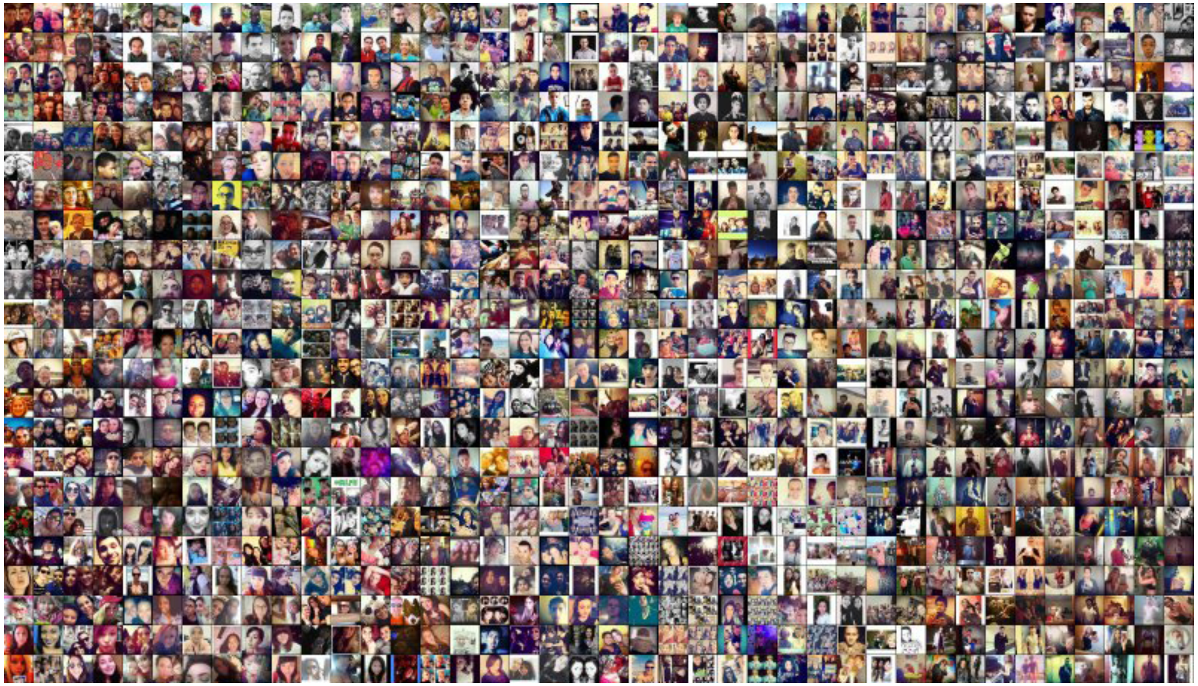
References

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

MNIST (LeCun et al 1998)

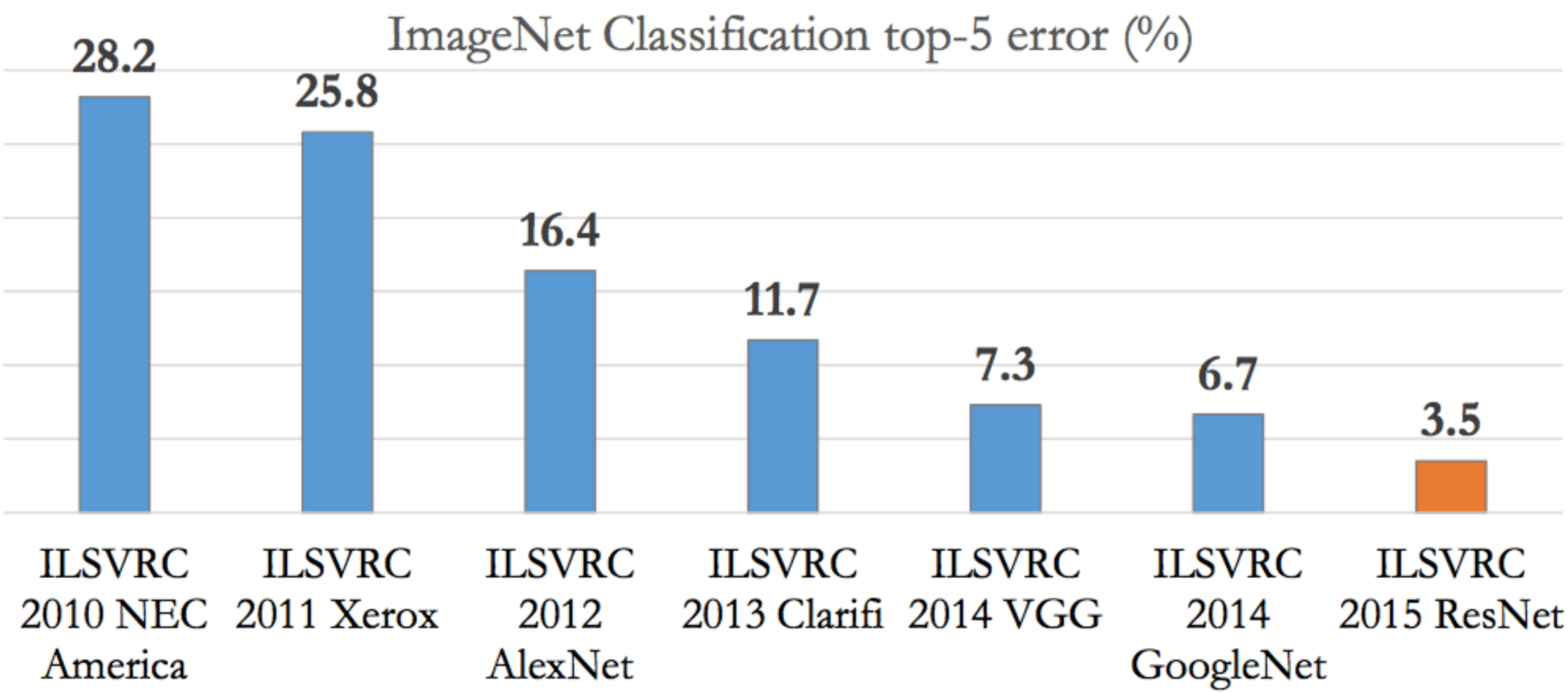


ImageNet and ILSVRC (2012)



Fei-Fei Li, Stanford

ImageNet and ILSVRC

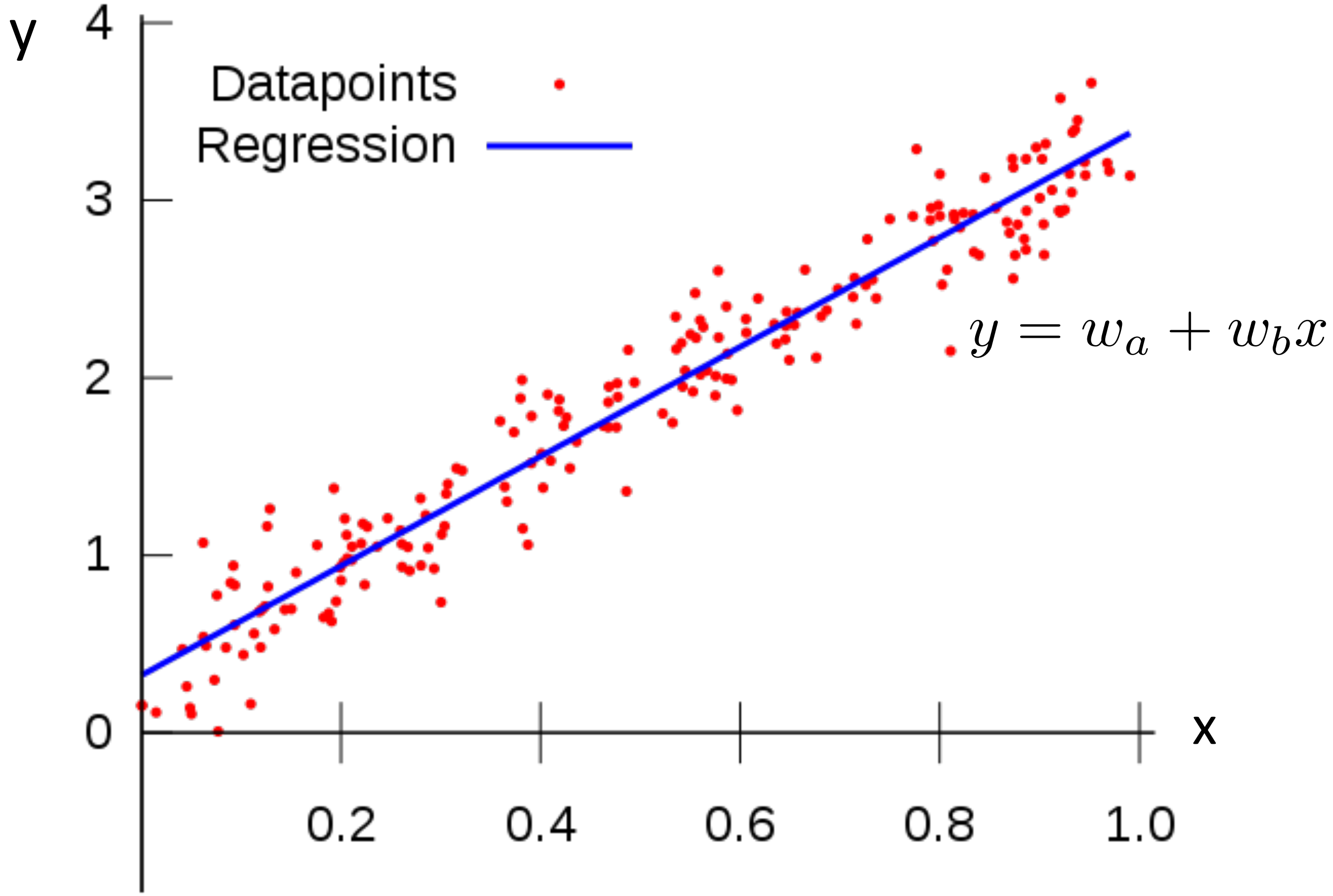


Why the sudden success?

- 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

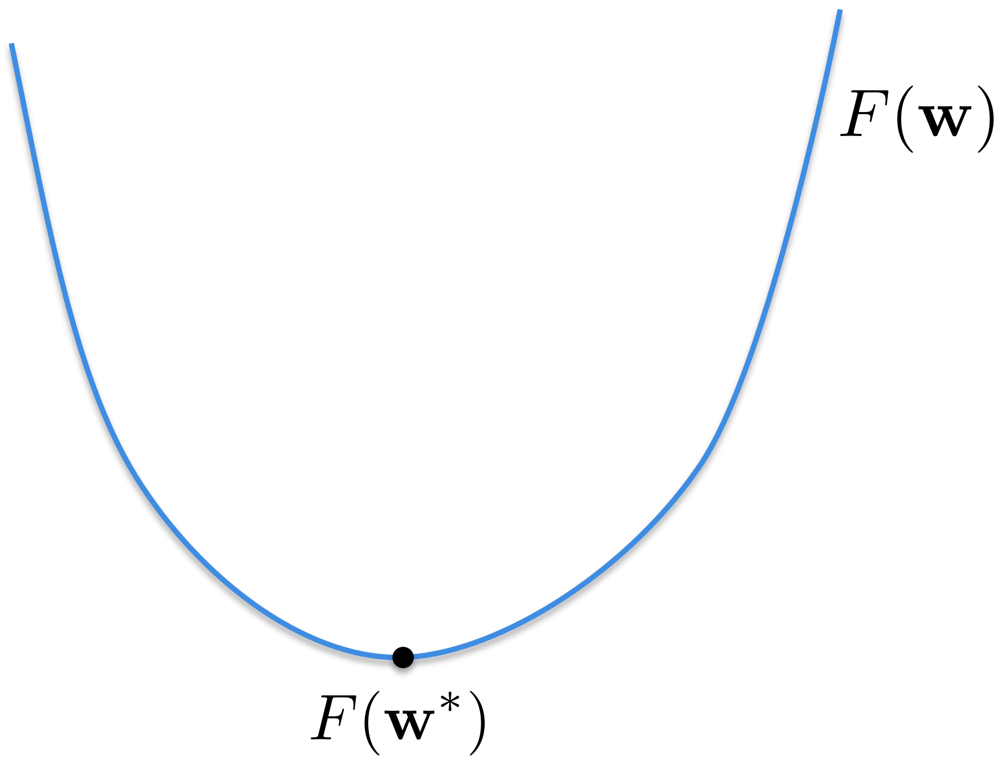


Given a big dataset of ($\mathbf{x_1}$, y_1), ($\mathbf{x_2}$, y_2), ($\mathbf{x_3}$, y_3), ($\mathbf{x_4}$, y_4), ….($\mathbf{x_N}$, y_N)

Find the optimal weights **w**

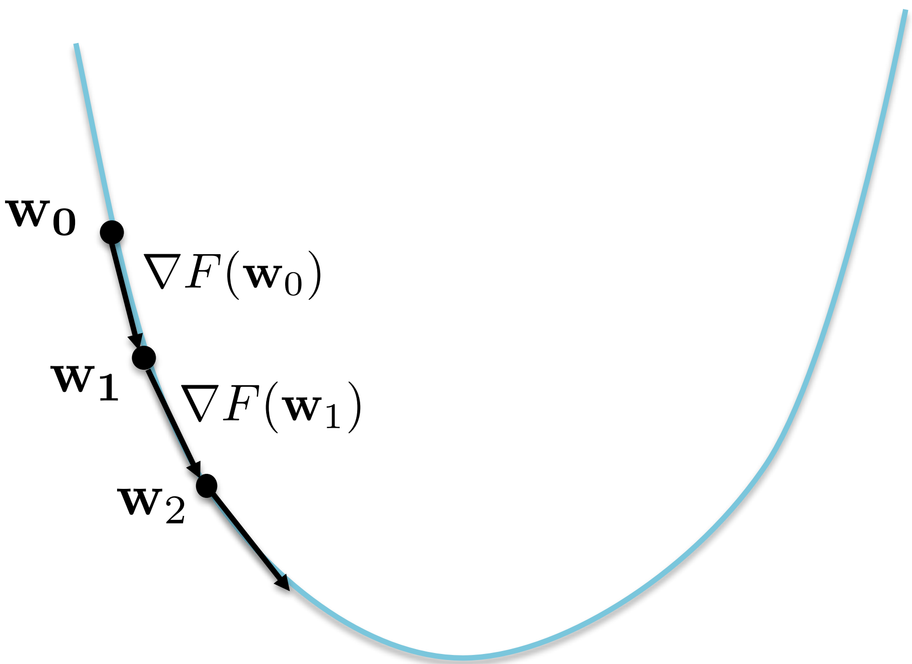
Core of ML: Gradient Descent (GD)

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

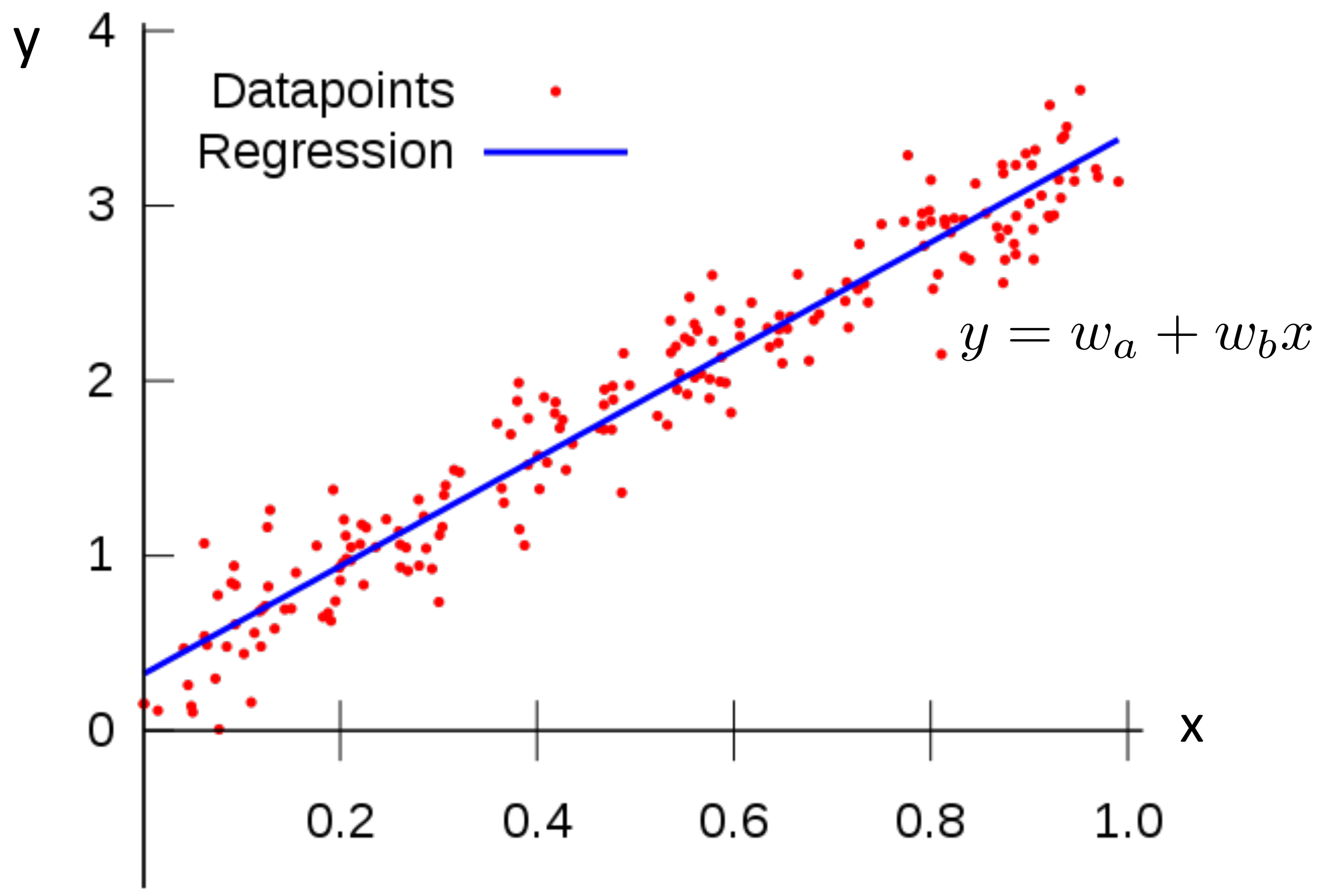


Core of ML: Gradient Descent (GD)

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



Exercise: Find the update rule for w_a and w_b



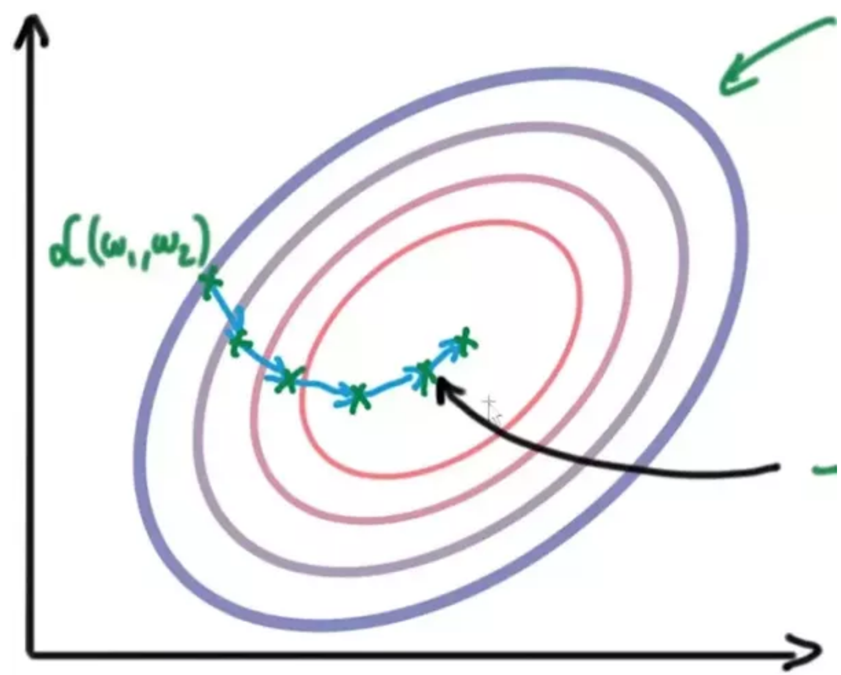
Given a big dataset of ($\mathbf{x_1}$, y_1), ($\mathbf{x_2}$, y_2), ($\mathbf{x_3}$, y_3), ($\mathbf{x_4}$, y_4), ...($\mathbf{x_N}$, y_N)

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

Gradient Descent (GD)

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

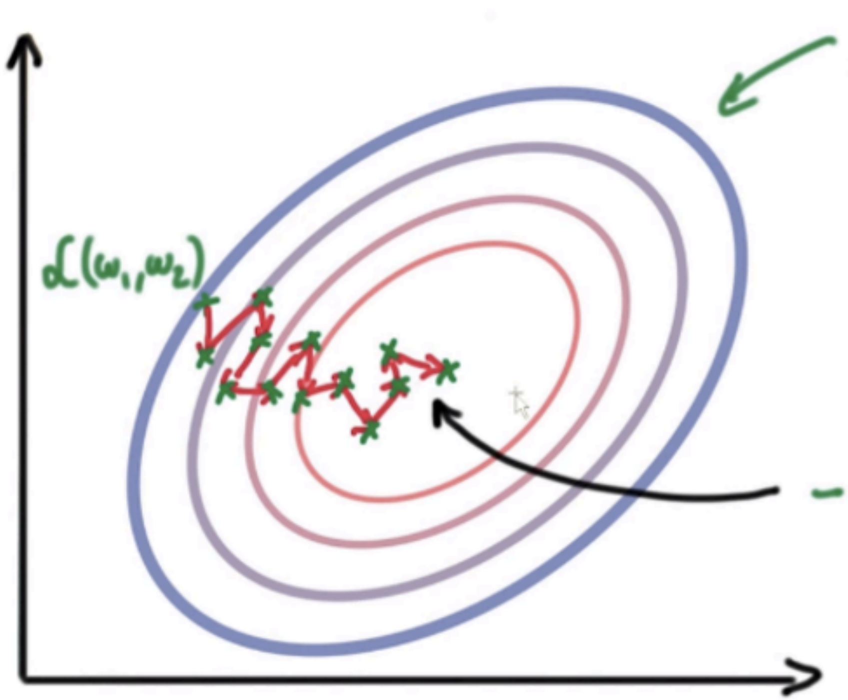
Too expensive
for large
datasets



Stochastic Gradient Descent (SGD)

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

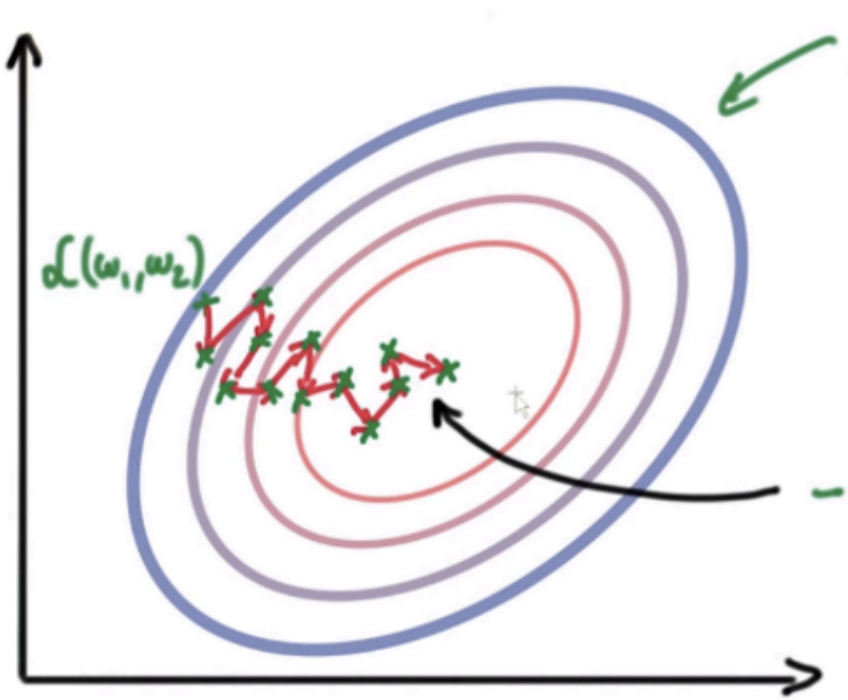
Easy, but possibly too noisy



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



Exercise: How does variance scale with m?

If $Var(\nabla F(\mathbf{w}, \xi_i)) = \sigma^2$

What is the variance of the gradient update in mini-batch SGD?

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \sum_{i=1}^{m} \frac{1}{m} \nabla F(\mathbf{w}_t, \xi_i)$$

Convergence of SGD

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

Error Floor

Decay Rate

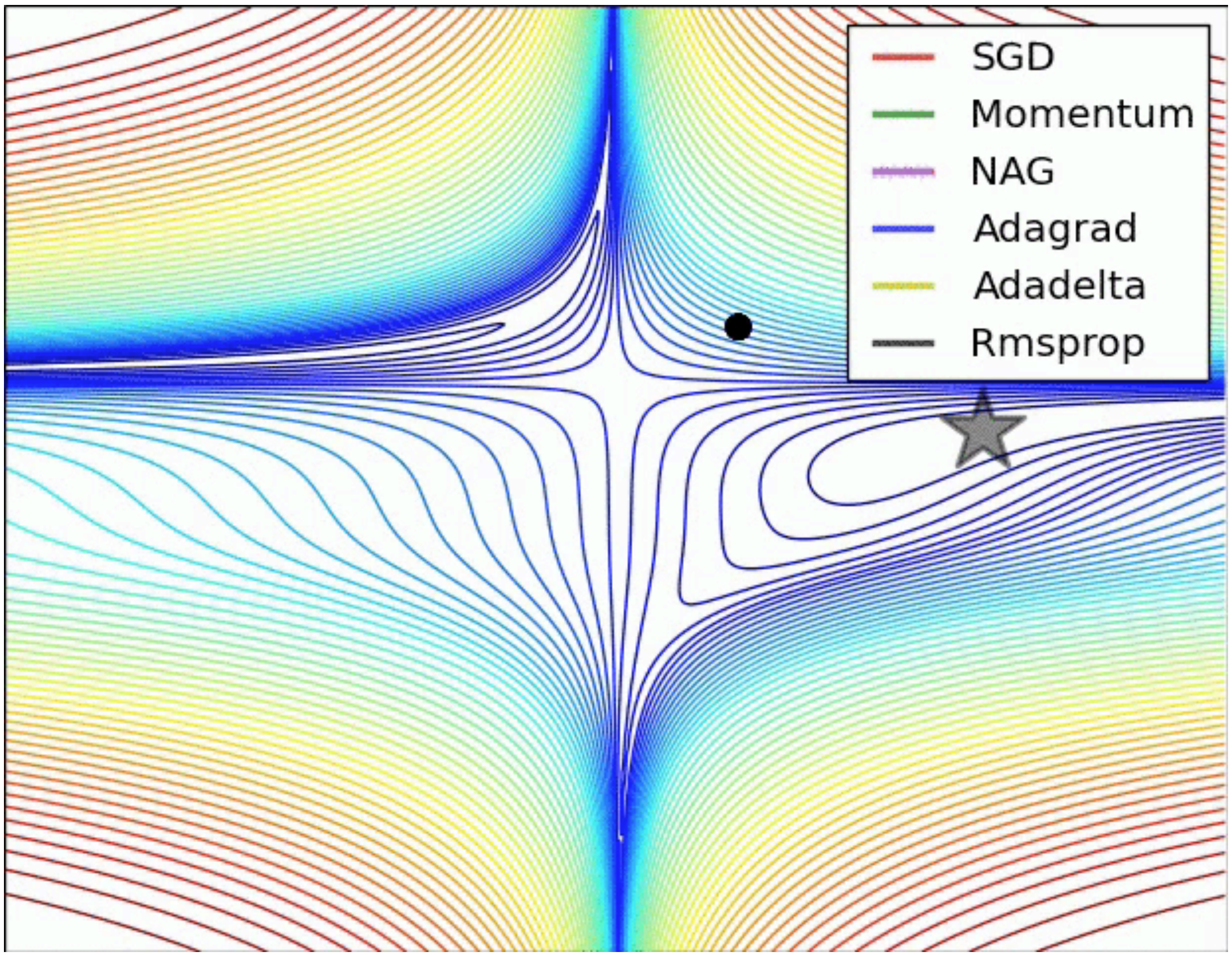
How does decay rate and error floor change with

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

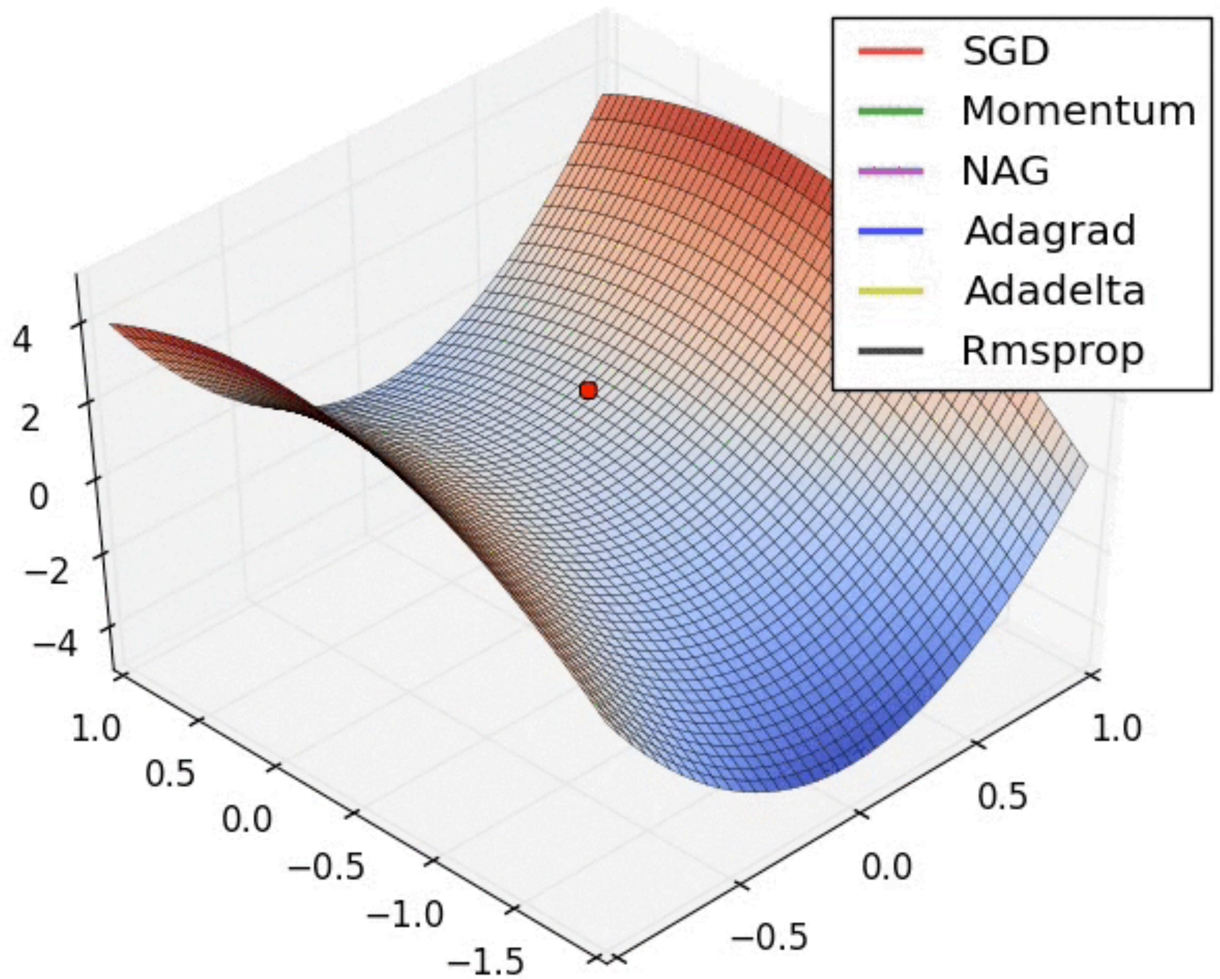
Many other variants of SGD

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

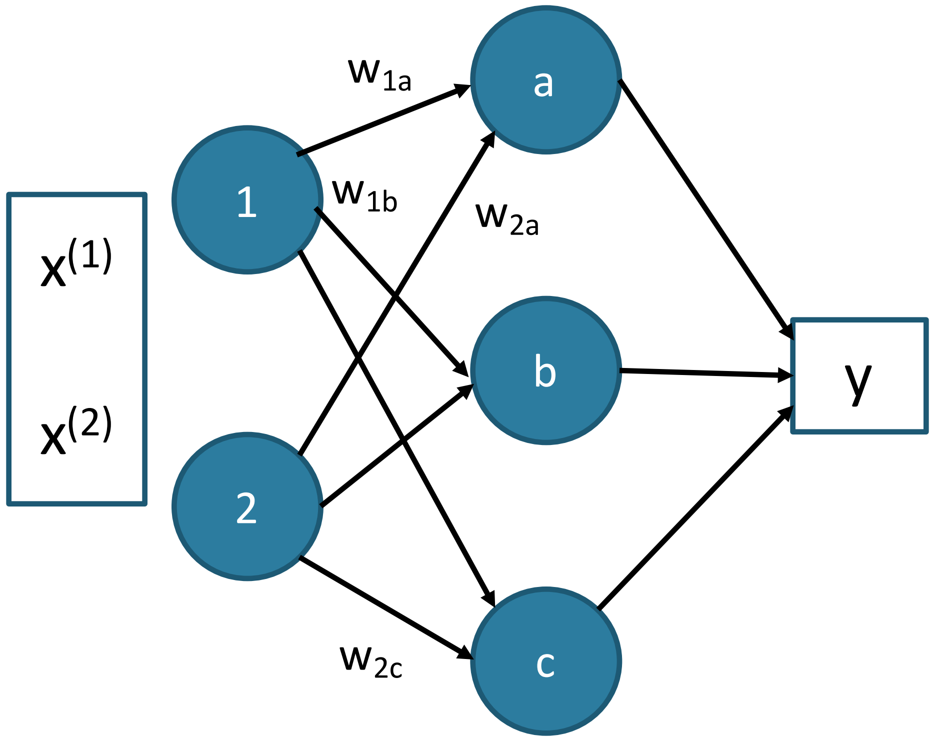
Many other variants of SGD



Many other variants of SGD

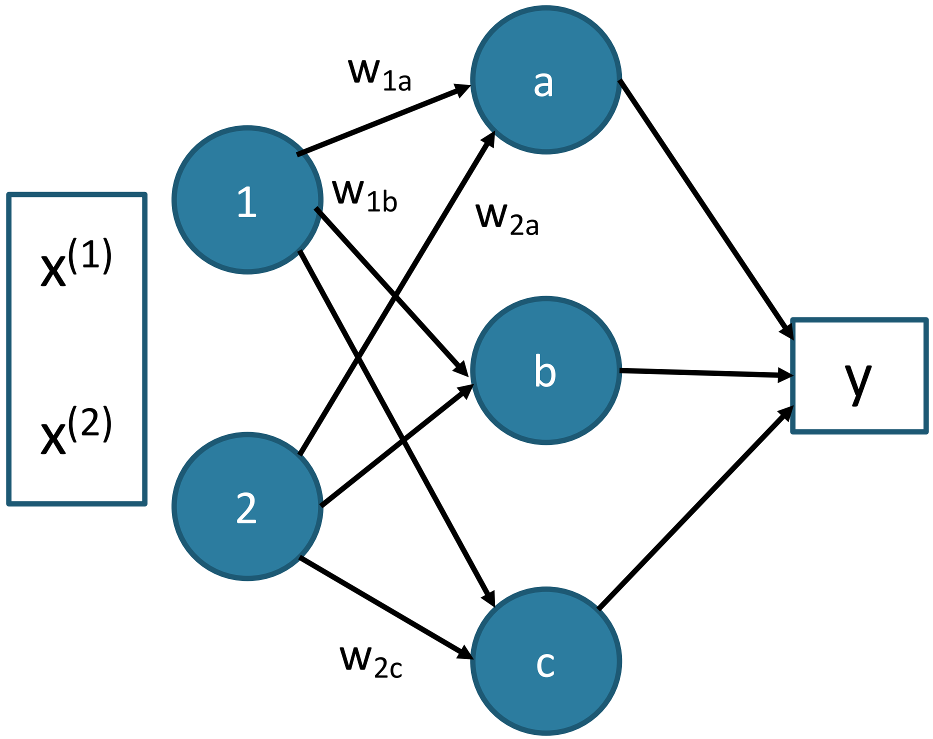


SGD and Backpropagation



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

SGD and Backpropagation

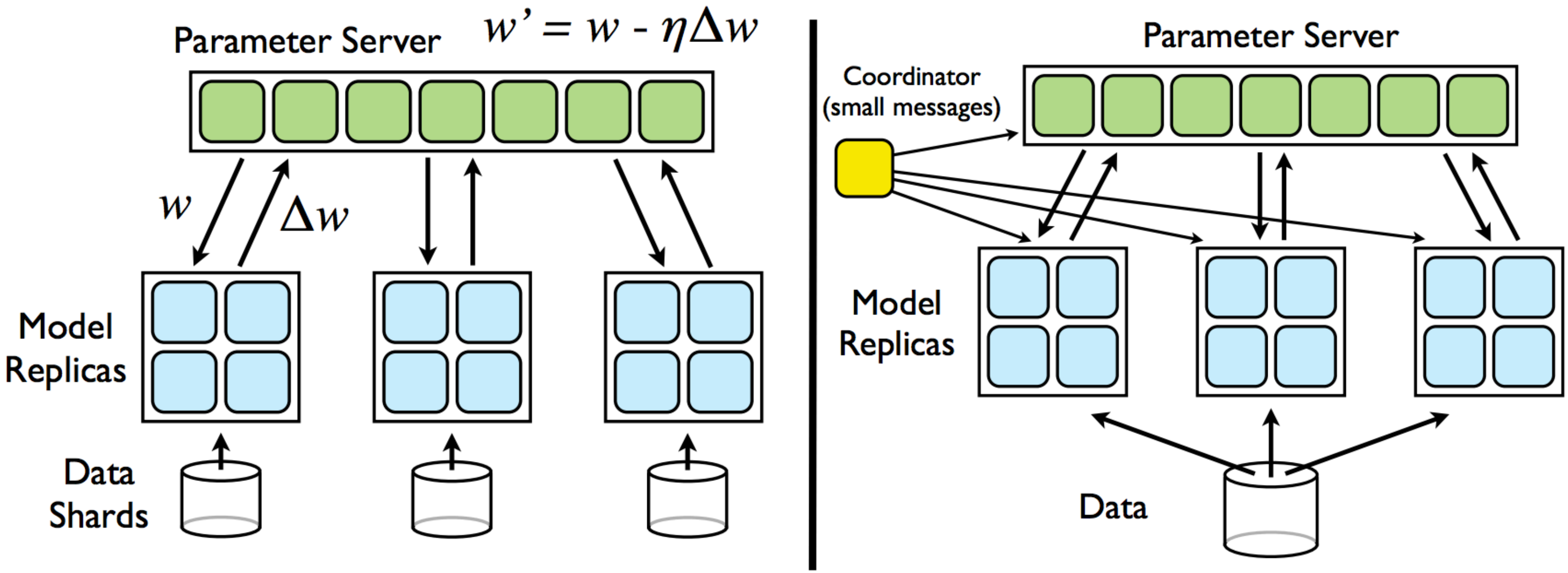


Input to a $=$ inp_a $= w_{1a}\, x_1 + w_{2a}\, x_2$

Output of a $=$ $out_a = g\,(inp_a)$

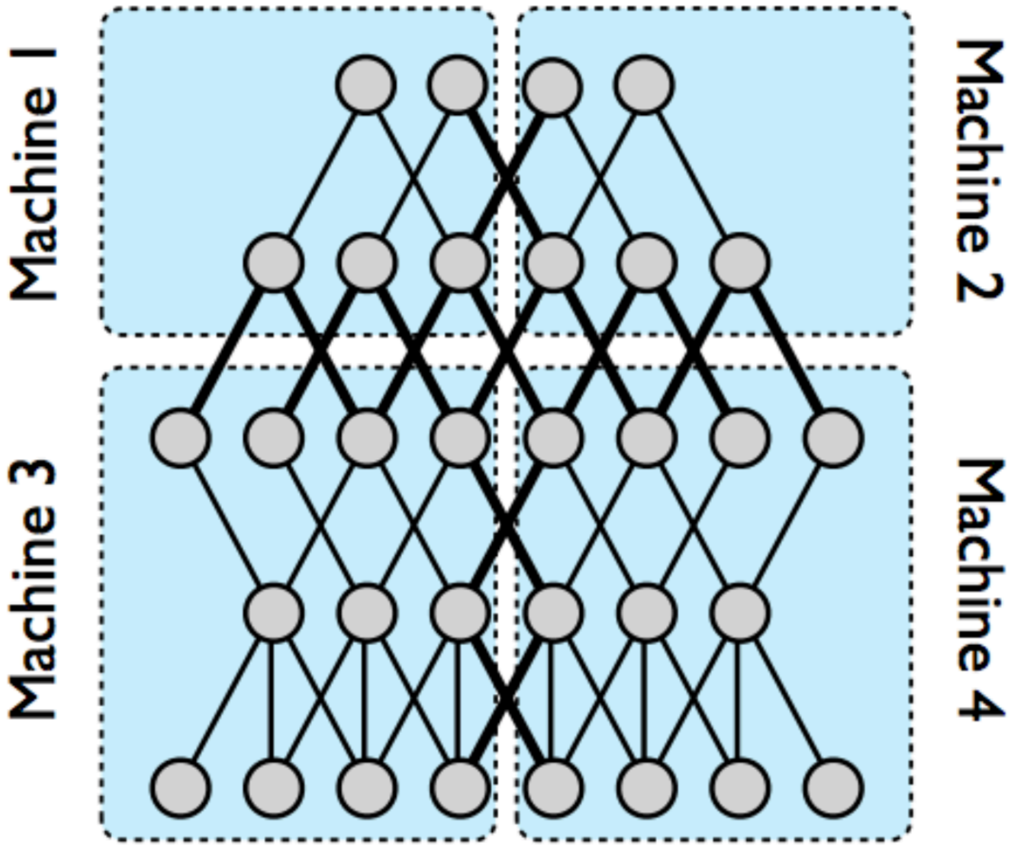
Distributed Deep Learning

Data Parallelism



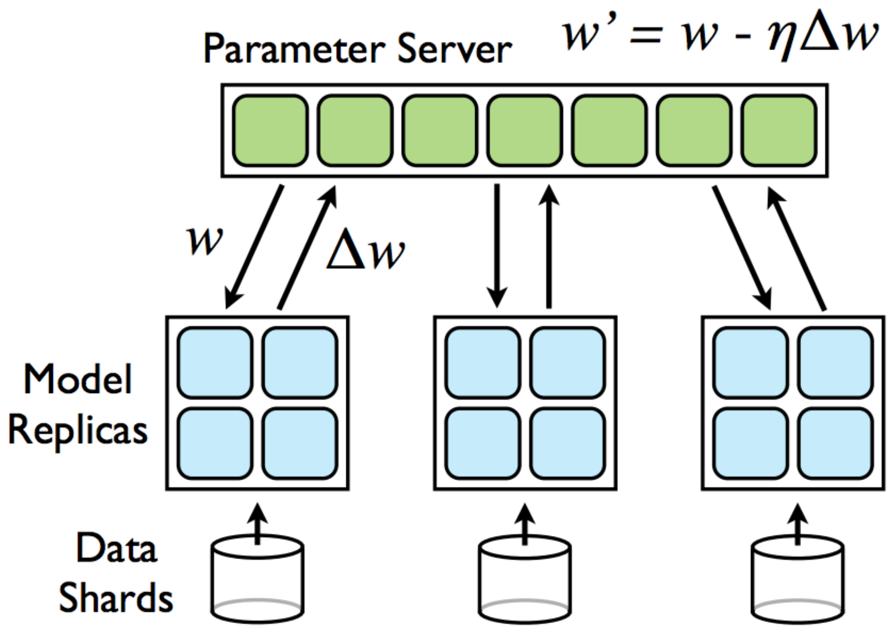
Distributed Deep Learning

Model Parallelism



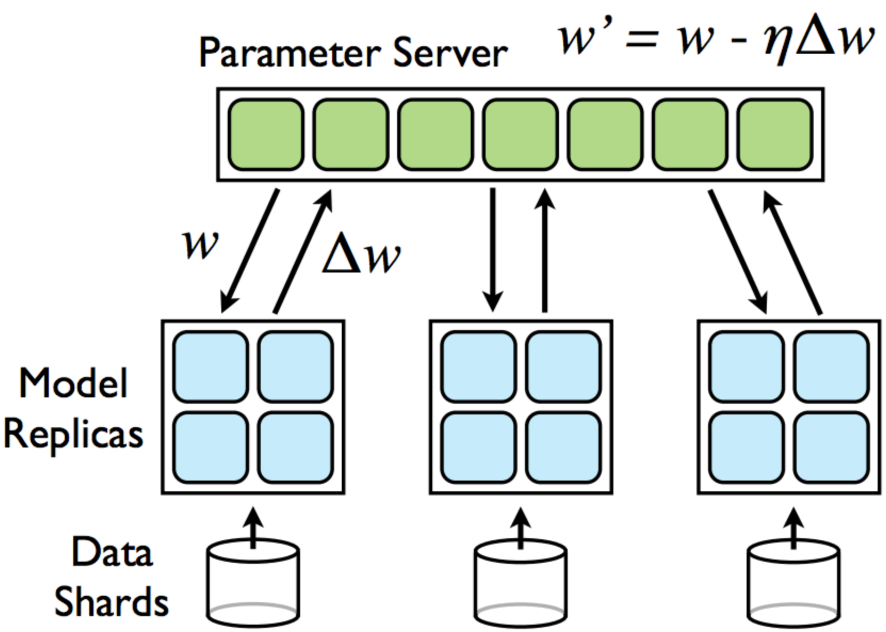
Synchronous SGD

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



Q: What is the convergence rate and error floor?

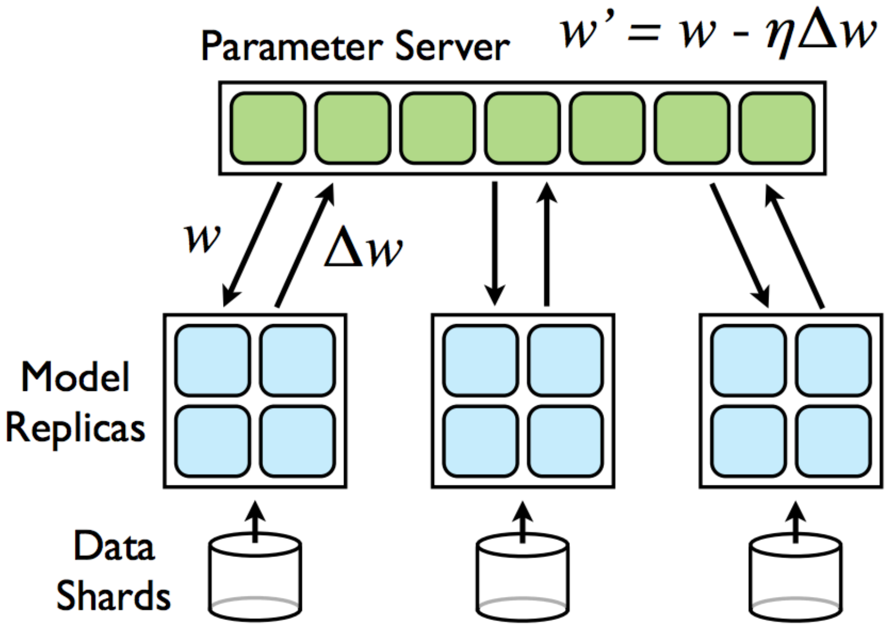
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \sum_{k=1}^{K} \frac{1}{K} \nabla F(\mathbf{w}_t, \xi_k)$$



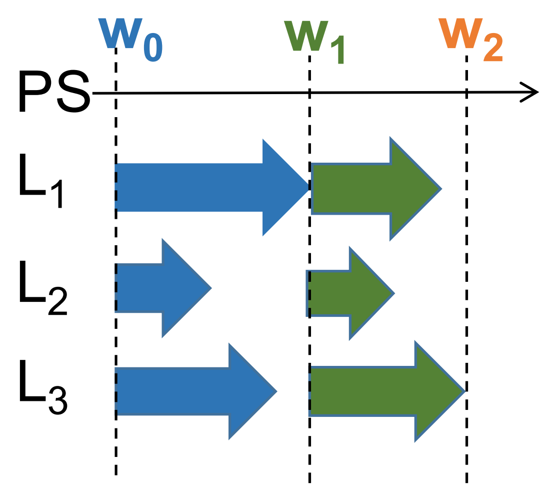
Q: What is the time to complete each iteration?

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

Slowest Learner is the bottleneck



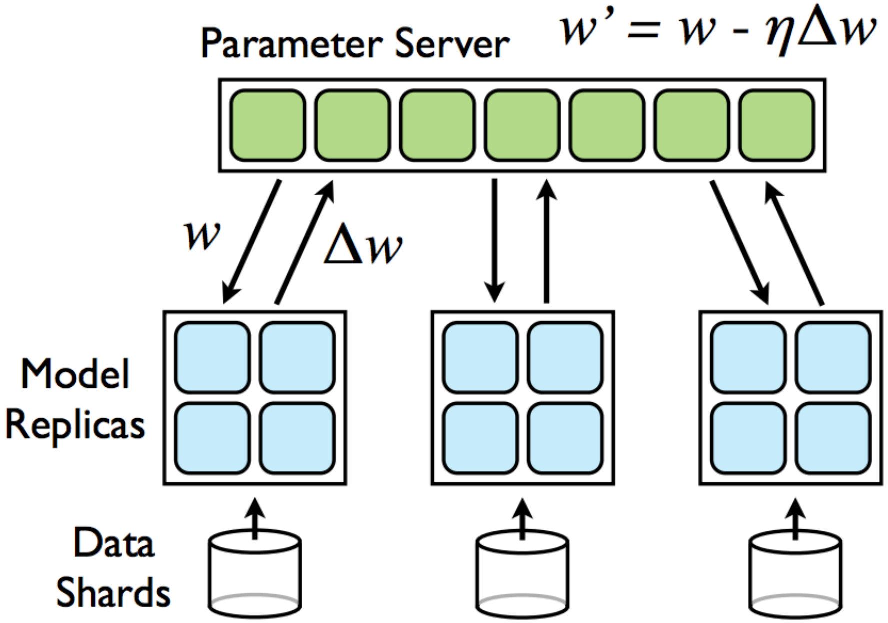
Fully Sync-SGD



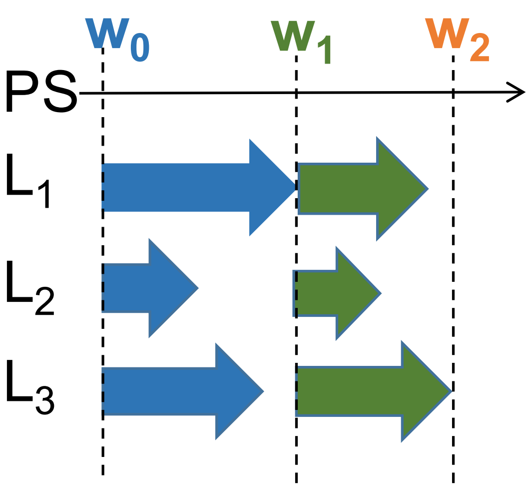
Q: How can we reduce it?

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

Slowest Learner is the bottleneck



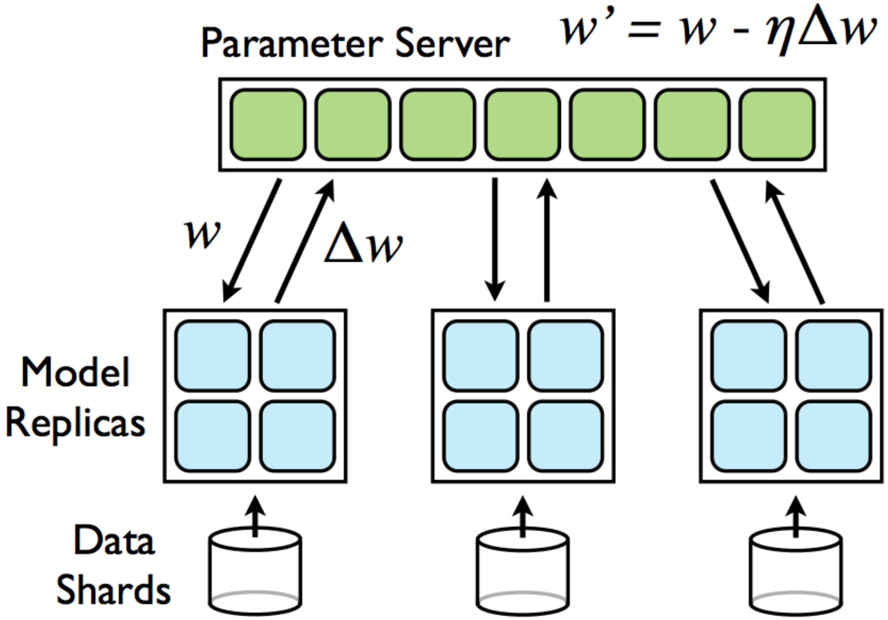
Fully Sync-SGD

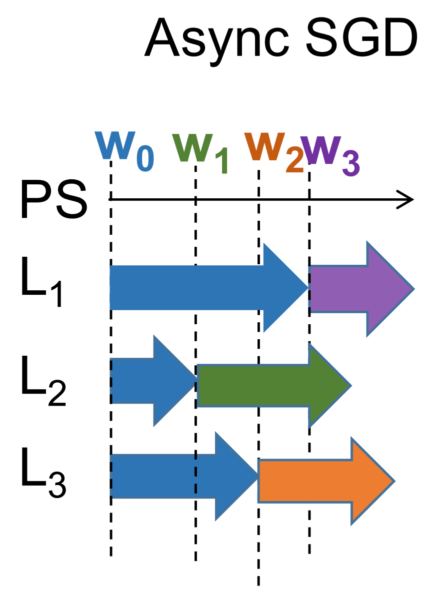


Asynchronous SGD: Don't wait for all

Asynchronous SGD cuts the latency tail.

But, what effect does it have on the error?





Variants of Distributed SGD

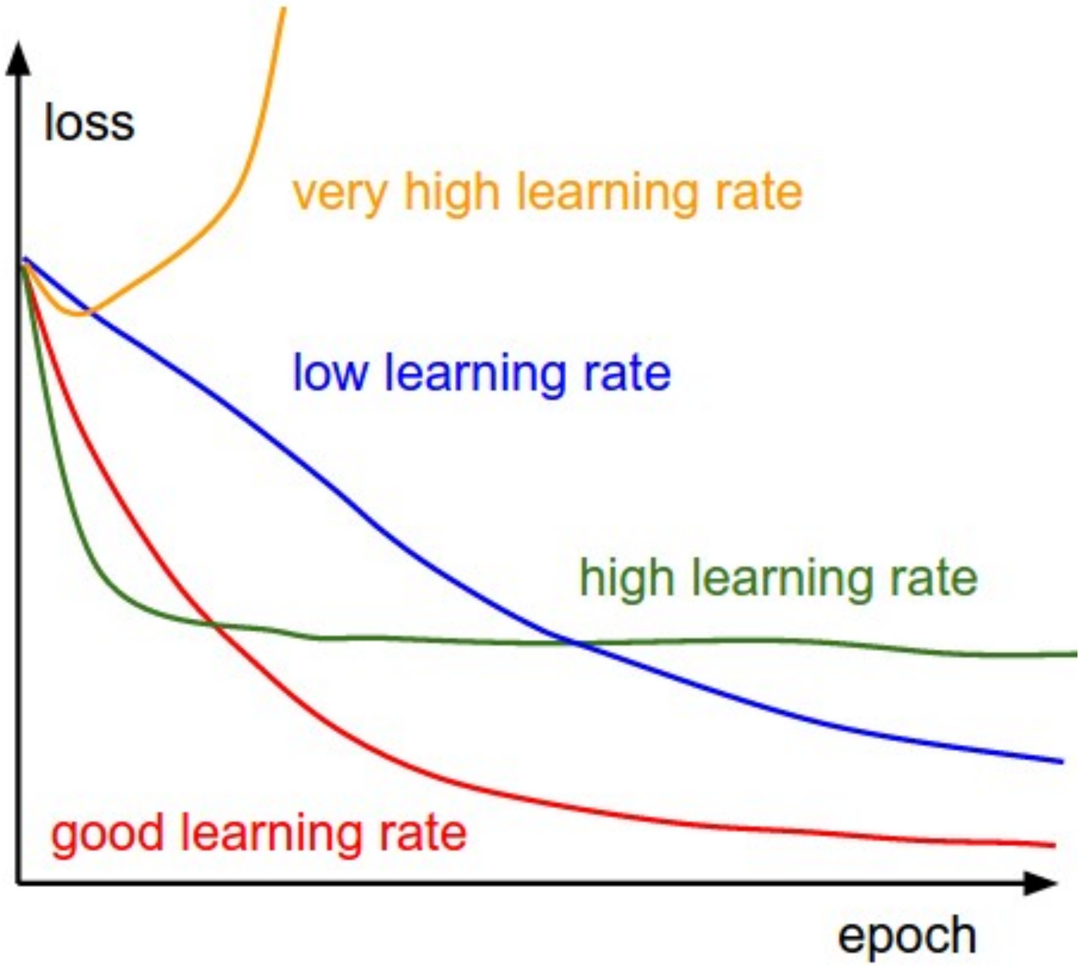
- Synchronous SGD
- Asynchronous SGD
- HogWild
- Elastic-Averaging SGD

Hyper-Parameter Tuning

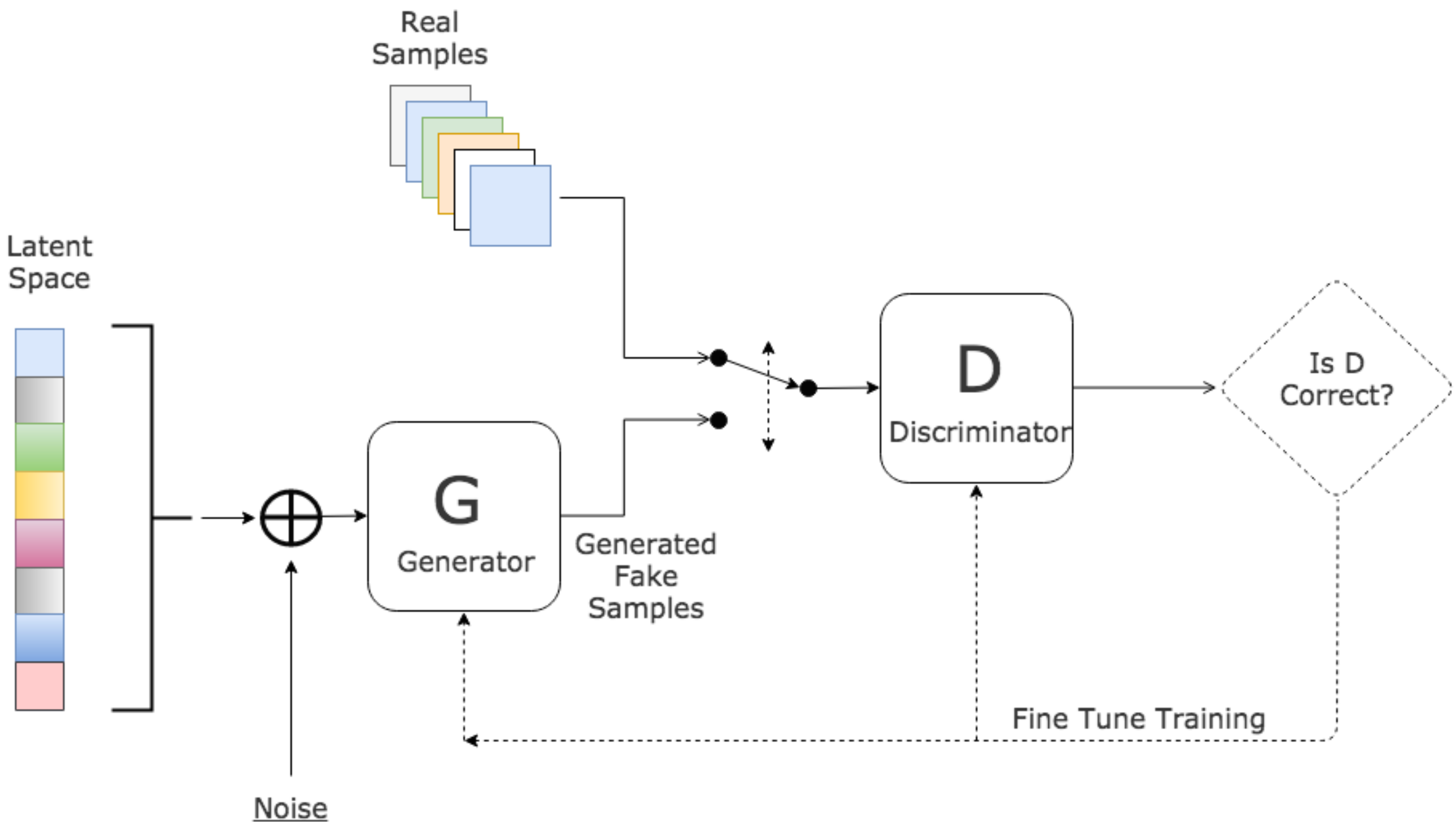
Need to choose the right

- Learning rate
- Mini-batch size
- Momentum
- Number of layers
- Number of neurons per layer

Hyper-Parameter Tuning



Generative Adversarial Networks



Reinforcement Learning

Reinforcement Learning