

Lecture 7 - Distributed ML

What is the Problem Being Solved?

- Training models is time consuming
 - Convergence can be slow
 - Training is computationally intensive
- Not all models fit in single machine or GPU memory
- Less of a problem: big data
 - Problem for data preparation / management
 - Not a problem for training ... why?

How do we measure success of large scale ML?

- **Machine Learning:** Minimize passes through the training data
 - Easy to measure, but not informative ... why?
- **Systems:** minimize time to complete a pass through the training data
 - Easy to measure, but not informative ... why?

Ideal Metric of Success

How do we measure Learning?

$$\left(\frac{\text{"Learning"}}{\text{Second}} \right) = \left(\frac{\text{"Learning"}}{\text{Record}} \right) \times \left(\frac{\text{Record}}{\text{Second}} \right)$$

Convergence
Machine Learning
Property

Throughput
System
Property

Key Problems Addressed in DistBelief Paper

Main Problem

- **Speedup training for large models**

Sub Problems

- How to partition models and data
- Variance in worker performance → Stragglers
- Failures in workers → Fault-Tolerance

The Gradient Descent Algorithm

$\theta^{(0)} \leftarrow$ initial model parameters (random, warm start)

For τ from 1 to convergence:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta_t \left(\frac{1}{n} \sum_{i=1}^n \nabla_{\theta} \mathbf{L}(y_i, f(x_i; \theta)) \right) \Bigg|_{\theta=\theta^{(t)}}^{\text{Evaluated at}}$$

Learning Rate

How do we distribute this computation?

Average Gradient of
Over Training Dataset

$\theta^{(0)} \leftarrow$ initial model parameters (random, warm start)

For τ from 1 to convergence:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta_t \left(\frac{1}{n} \sum_{i=1}^n \nabla_{\theta} \mathbf{L}(y_i, f(x_i; \theta)) \right) \Bigg|_{\theta=\theta^{(t)}}^{\text{Evaluated at}}$$

Learning Rate

How do we distribute this computation?

Average Gradient of
Over Training Dataset

Data parallelism: divide data across machines, compute local gradient sums and then aggregate across machines. repeat.

Issues? Repeatedly scanning the data... what if we cache it?

The empirical gradient is an approximation of what I really want:

$$\eta_t \left(\frac{1}{n} \sum_{i=1}^n \nabla_{\theta} \mathbf{L}(y_i, f(x_i; \theta)) \right) \approx \mathbb{E}_{(x,y) \sim \mathcal{D}} [\nabla_{\theta} \mathbf{L}(y, f(x; \theta))] \Big|_{\theta=\theta^{(t)}}$$

Law of large numbers → more data provides a better approximation (variance in the estimator decreases linearly)

Do I really need to use all the data?

$$\frac{1}{n} \sum_{i=1}^n \nabla_{\theta} \mathbf{L}(y_i, f(x_i; \theta)) \approx \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\theta} \mathbf{L}(y_i, f(x_i; \theta))$$

Random subset of the data

Small B : fast but less accurate
Large B : slower but more accurate

Gradient Descent	$\theta^{(0)} \leftarrow$ initial vector (random, zeros ...) For t from 1 to convergence: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta_t \left(\frac{1}{n} \sum_{i=1}^n \nabla_{\theta} \mathbf{L}(y_i, f(x_i; \theta)) \Big _{\theta=\theta^{(t)}} \right)$	Assuming Decomposable Loss Functions
	$\theta^{(0)} \leftarrow$ initial vector (random, zeros ...) For t from 0 to convergence: $\mathcal{B} \sim$ Random subset of indices $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta_t \left(\frac{1}{ \mathcal{B} } \sum_{i \in \mathcal{B}} \nabla_{\theta} \mathbf{L}(y_i, f(x_i; \theta)) \Big _{\theta=\theta^{(t)}} \right)$	

Stochastic Gradient Descent

How do you distribute SGD?

Stochastic Gradient Descent

$\theta^{(0)} \leftarrow$ initial vector (random, zeros ...)

For t from 0 to convergence:

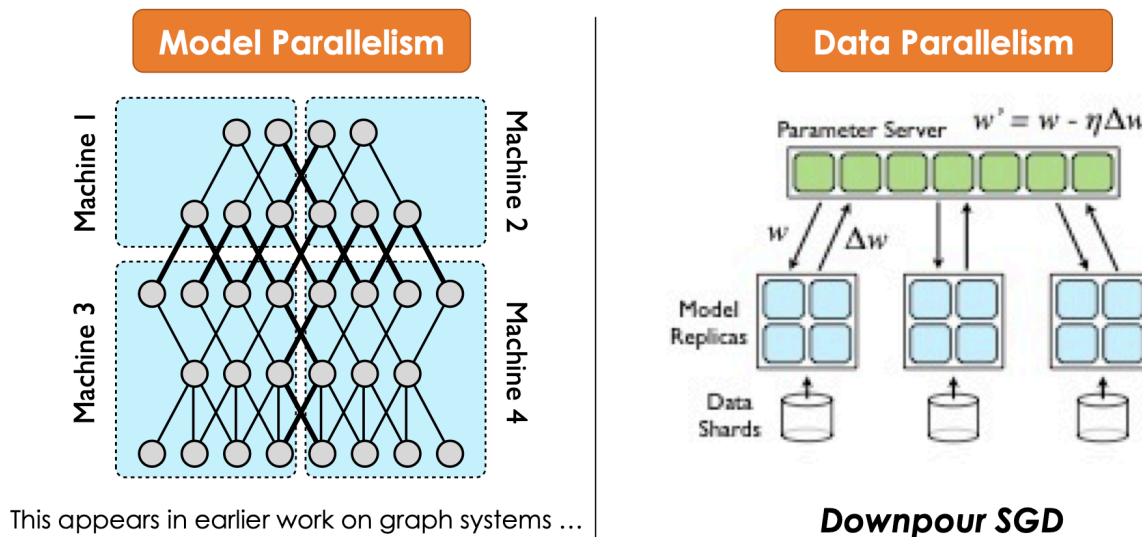
$\mathcal{B} \sim$ Random subset of indices

$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta_t \left(\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\theta} \mathbf{L}(y_i, f(x_i; \theta)) \right) \Big|_{\theta = \theta^{(t)}}$

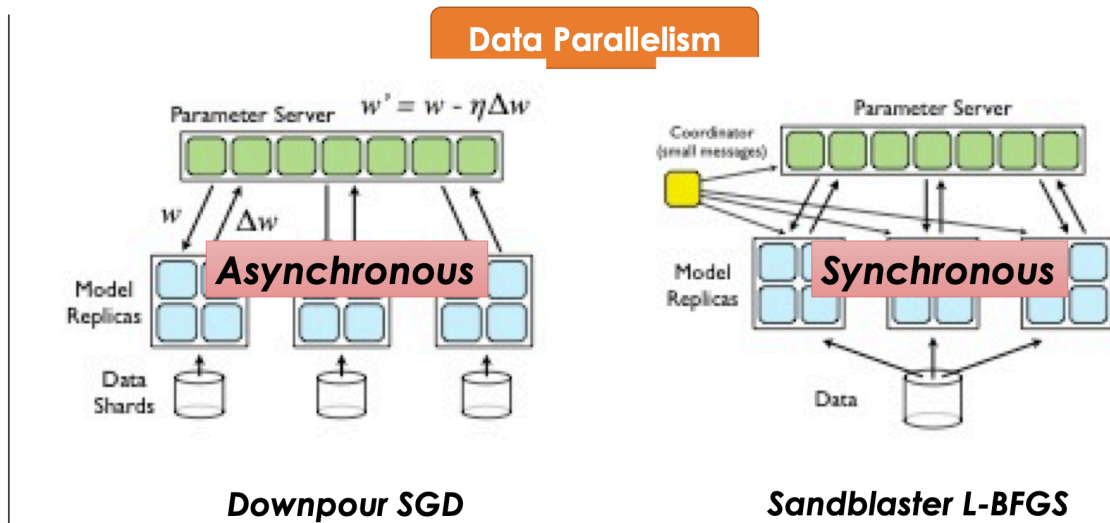
Model Parallelism
Speed up Gradient. Depends on Model

Data Parallelism
Speed up Sum. Depends on size of B

Combine Model and Data Parallelism



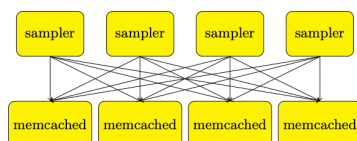
Combine Model and Data Parallelism



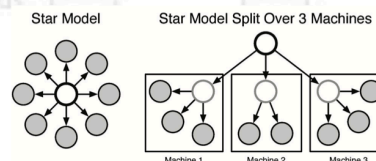
Parameter Servers

- Essentially a **sharded** key-value store
 - support for put, get, **add**
- Idea appears in earlier papers:

"An Architecture for Parallel Topic Models", Smola and Narayanamruthy. (VLDB'10)



"Scalable Inference in Latent Variable Models", Ahmed, Aly, **Gonzalez**, Narayanamruthy, and Smola. (WSDM'12)



DistBelief was probably the first paper to call a sharded key-value store a Parameter Server.

How do you distribute SGD?

Stochastic Gradient Descent

$\theta^{(0)} \leftarrow$ initial vector (random, zeros ...)

For t from 0 to convergence:

$\mathcal{B} \sim$ Random subset of indices

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta_t \left(\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\theta} \mathbf{L}(y_i, f(x_i; \theta)) \Big|_{\theta=\theta^{(t)}} \right)$$

**Data
Parallelism**

Slow? (~150ms)
Depending on size of B

Batch Size Scaling

- Increase the batch size by adding machines

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \hat{\eta} \left(\frac{1}{k} \sum_{j=1}^k \frac{1}{|\mathcal{B}_j|} \sum_{i \in \mathcal{B}_j} \nabla_{\theta} \mathbf{L}(y_i, f(x_i; \theta)) \Big|_{\theta=\theta^{(t)}} \right)$$

- Each server processes a fixed batch size (e.g., $n=32$)
- As more servers are added (k) the effective overall batch size increases linearly
- Why do these additional servers help?

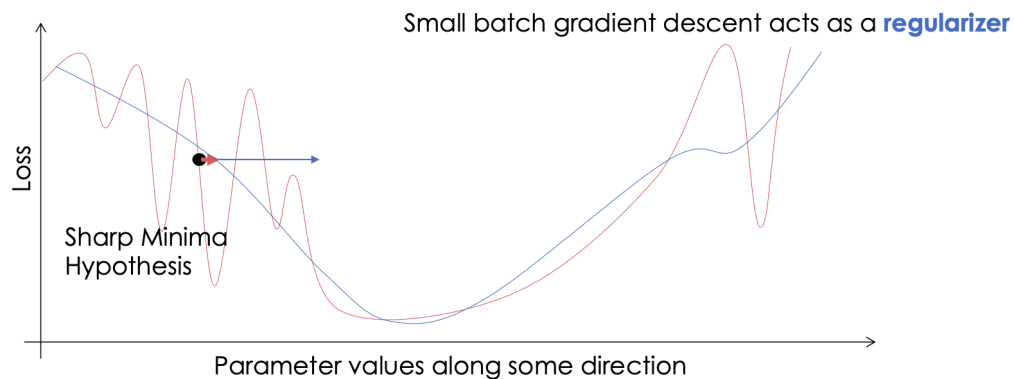
Bigger isn't Always Better

- Motivation for larger batch sizes
 - More opportunities for parallelism → but is it useful?
 - Recall (1/n variance reduction):

$$\frac{1}{n} \sum_{i=1}^n \nabla_{\theta} \mathbf{L}(y_i, f(x_i; \theta)) \approx \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\theta} \mathbf{L}(y_i, f(x_i; \theta))$$

- Is a variance reduction helpful?
 - Only if it let's you take bigger steps (move faster)
 - Doesn't affect the final answer...

Rough “Intuition”



Key problem: Addressing the generalization gap for large batch sizes.

Solution: Linear Scaling Rule

- Scale the learning rate linearly with the batch size

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \hat{\eta} \left(\frac{1}{k} \sum_{j=1}^k \frac{1}{|\mathcal{B}_j|} \sum_{i \in \mathcal{B}_j} \nabla_{\theta} \mathbf{L}(y_i, f(x_i; \theta)) \Big|_{\theta=\theta^{(t)}} \right)$$

- Addresses generalization performance by **taking larger steps** (also improves training convergence)
- **Sub-problem:** *Large learning rates can be destabilizing in the beginning. Why?*
 - **Gradual warmup solution:** increase learning rate scaling from constant to linear in first few epochs
 - Doesn't help for very large k...