Parallelization in the "Big Data" Regime 5: Data Parallelization?

Sham M. Kakade

Machine Learning for Big Data
CSE547/STAT548

University of Washington

Announcements...

- HW 3 posted
- Projects: the term is approaching the end.... (summer is coming!)

Today:

- Review: Adaptive gradient methods
- Parallelization: how do we parallelize in the “big data” regime?

Parallelization Overview

- Basic question: How can we do more operations "simultaneously" so that our overall task finishes more quickly?
- Issues:
  1. Break up computations on: single machine vs. cluster
  2. Breakup up: Models or Data
     Model parallelization or Data parallelization?
  3. Asynchrony?
- What are good models to study these issues?
  communication complexity, serial complexity, total computation, ??

1: One machine or a cluster?

- One machine:
  - Certain operations are much faster to do when specified without “for loops”: matrix multiplications, convolutions, Fourier transforms,
  - “parallelize” by structuring computations to take advantage of this: e.g. for larger matrix multiplies
  - GPUs!!!
  - Why one machine?
    Shared memory/communication is fast! Try to take advantage of fast “simultaneous” operations.
- Cluster:
  - Why? One machine can only do so much.
  - Try to (truly) breakup computations to be done.
  - Drawbacks: Communication is costly!
  - Simple method: run multiple jobs with different parameters.

2: Data Parallelization vs. Model parallelization

- Data parallelization:
  Breakup data into smaller chunks to process.
  - Mini-batching, batch gradient descent
  - Averaging
- Model parallelization:
  Breakup up your model.
  - Try to update parts of model in a distributed manner.
  - Coordinate ascent methods
  - Update layers of a neural net on different machines.
- Other issues:
  Asynchrony

Issues to consider...

- Work: the over all compute time.
- Depth: the serial runtime.
- Communication: between machines?
- Error: terminal error.
- Memory: how much do we need?

Mini-batching (Data Parallelization)

- stochastic gradient computation: at iteration t, compute:

$$w_{t+1} \leftarrow w_t - \eta \widehat{\nabla \ell(w_t)}$$

  where $\mathbb{E}\widehat{\nabla \ell(w)} = \nabla L(w)$.
- mini-batch SGD: using batch size b:

$$w_{t+1} \leftarrow w_t - \eta_b \left(\frac{1}{b} \sum_{j=1}^{b} \widehat{\nabla \ell_j(w_t)} \right)$$

  where η_b is our learning rate.
- How much does this help?
- It clearly reduces the variance.

How much does mini-batching help?

- Let's consider the regression/square loss case.
- $\overline{w}_t = \mathbb{E}[w_t]$ is the expected iterate at iteration t.
- Parameter decomposition

Dev. from the mean.

$$w_t - w_* = \underbrace{w_t - \overline{w}_t}_{\text{"Variance"}} + \underbrace{(\overline{w}_t - w_*)}_{\text{Bias}}$$

- mini-batch SGD: using batch size b:

$$w_{t+1} \leftarrow w_t - \eta_b \left(\frac{1}{b} \sum_{j=1}^{b} \widehat{\nabla \ell_j(w_t)} \right)$$

where η_b is our learning rate.

- How much does this help?
  Bias? Variance?

Variance reduction with b?

- Variance term (as a function of the mini-batch size):

$$\|w_t - \overline{w}_t\|^2 \approx O(1/b)$$

- This means with mini-batch size of b, the contribution to the error is $O(b)$ times less, as compared to using $b = 1$ (with the *same* number of iterations)
- However: as long as the Bias $\geq$ Variance, then no point in trying to make the Variance term smaller.

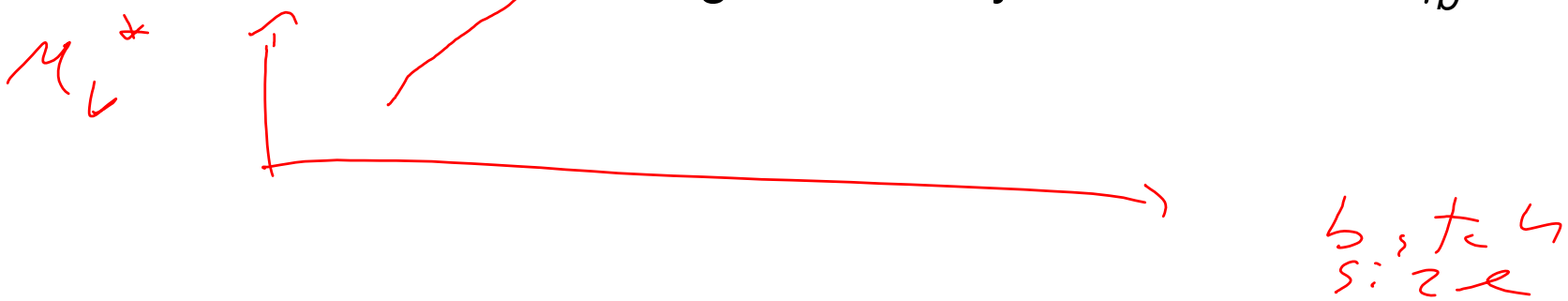
How much does mini-batching help the bias?

Bias reduction with b?

- As $b \to \infty$, do we continue to expect improvements to the Bias? For large enough b, we are just doing batch/exact gradient descent:

$$w_{t+1} \leftarrow w_t - \eta \nabla L(w_t)$$

- What happens in between $b = 1$ and $b \to \infty$?
- Define η_b^* as the maximal learning rate (again for the square loss case) that you can use before divergence.
- Theorem: The maximal learning rate strictly increases with η_b^*.



The "critical" batch size b

- Let $\tilde{b}$ be the critical b in which η_b^* is $\eta_\infty^*/2$.
- Theorem: For every square loss problem, in comparison to $b = 1$ (SGD), we have:

$$\|\overline{w}_{t+1} - w_*\| \leq \exp(-\gamma_b)\|\overline{w}_t - w_*\|$$

  - When $b \leq \tilde{b}$, the Bias contraction γ_b linearly increases with b.
  - When $b \geq \tilde{b}$, γ_b is within a factor of 2 times γ_∞.

$$\gamma_\infty = \gamma = \frac{\lambda_{min}}{\lambda_{max}}$$

How much can you mini-batch?

- Let $\tilde{b}$ be the critical b in which η_b^* is $\eta_\infty^*/2$.
- Theorem: Suppose $\|x\|^2 \leq L$ (almost surely) and $\lambda_{\max}$ is the maximal eigenvalue of $\mathbb{E}[xx^\top]$. Then:

$$\frac{\sum \lambda_i}{\lambda_{\max}} = \frac{\mathbb{E}[\|x\|^2]}{\lambda_{\max}} \leq \tilde{b} \leq \frac{L}{\lambda_{\max}}$$

(for not very kurtotic distributions, $L \approx \mathbb{E}[\|x\|^2]$).

- How big is $\tilde{b}$ in practice?
- (The above are really exact expressions for every problem). In one dimension:

$$\tilde{b} = \frac{\mathbb{E}[x^4]}{(\mathbb{E}[x^2])^2}$$

Putting it all together

- Comparing to $b = 1$, with mini-batching, you can get the same error with:
  - Depth: reduce the serial run-time by a factor of $\tilde{b}$.
  - Work: keep the over all compute time the same.
  - Communication: need average $\tilde{b}$ parameters, per update.
- Initially, above $\tilde{b}$ mini-batching is useless.
- Asymptotically (for large enough iterations t, when the Bias becomes smaller than the variance), mini-batching more is always helpful.
- Practice/Punchline?? $\tilde{b}$ is often not all that large for natural problems.This favors the "GPU" model on one machine.

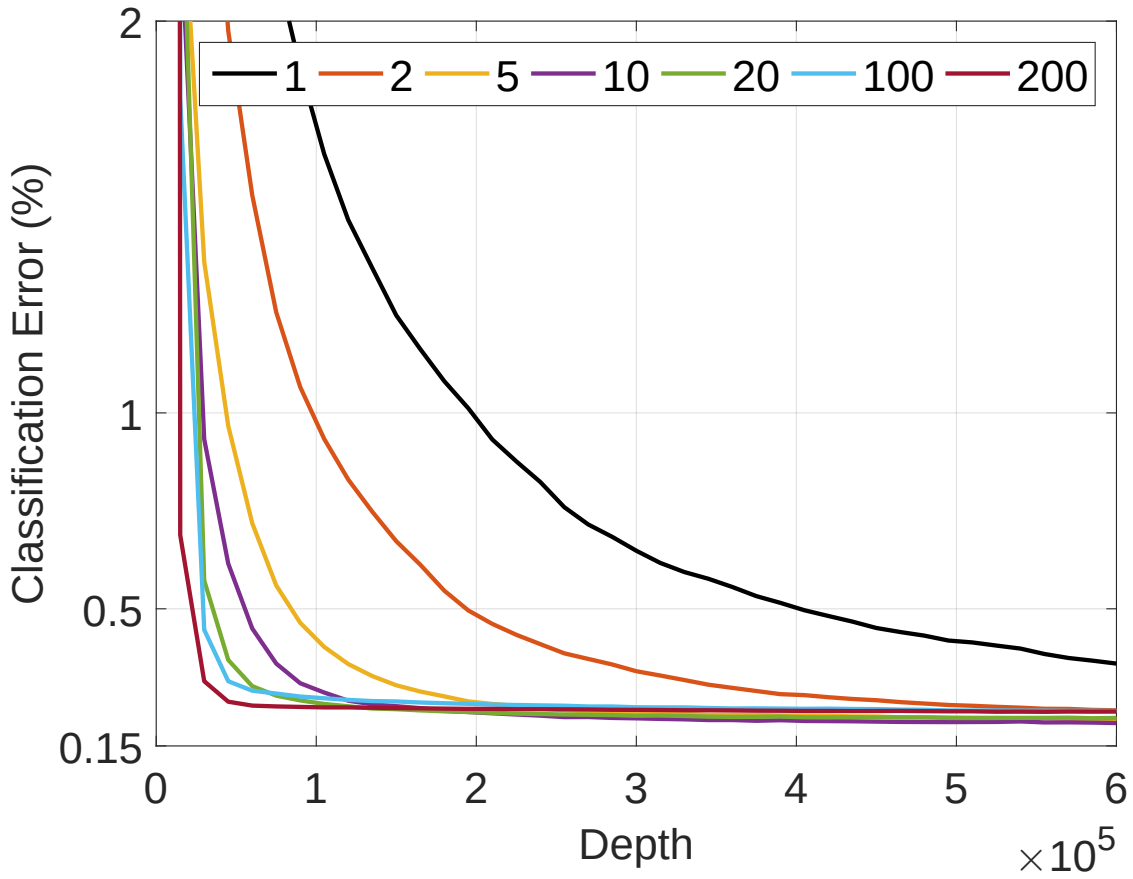
Extra Slide

Extra Slide

Is it general?

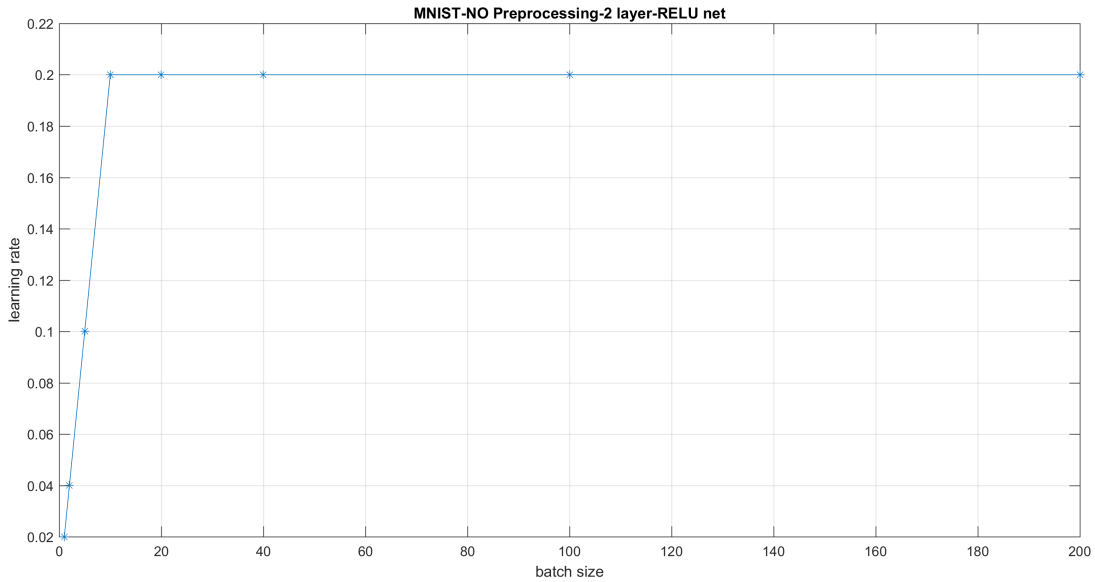
- Claims were only made for SGD in the square loss case.
- How general are these ideas? the non-convex case?
- Empirically, we often go with a “GPU model” where we max out our mini-batch size.

Mini-batching: Neural Net training on Mnist



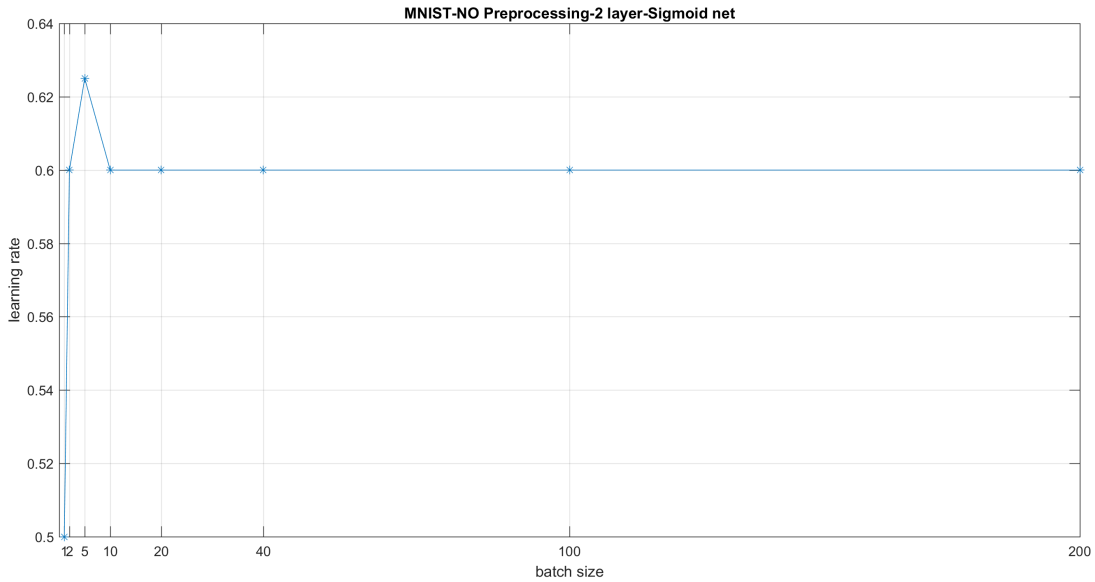
- Depth: The number of serial iterations.
- Work: Depth $\otimes$ the mini-batch size.

Neural Net Learning rates vs. batch size



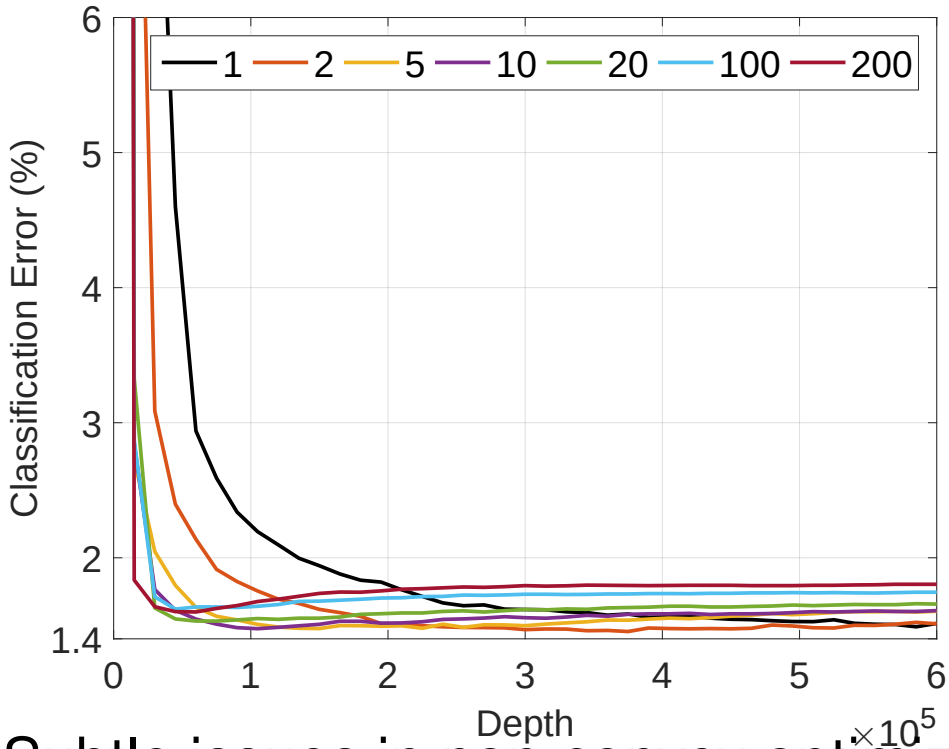
- The 'maximal' learning rate as a function of the batch size.
- Found with a crude grid search.

Neural Net Learning rates vs. batch size



- The 'maximal' learning rate as a function of the batch size.
- Found with a crude grid search.

Neural Net: Test error vs. batch size



- Subtle issues in non-convex optimization.
- More overfitting seen here with larger bath sizes.
  unclear how general this is.
- for this case, we were too aggressive with the learning rates.

Averaging

- With “mini-batching”, our presentation suggests you might as well just use a single machine with mini-batching.
- What can we do with multiple machines?
- In the convex case:
  - Breakup your data on multiple machines (Must still have “enough” data on each machine.)
  - Run (mini-batch) SGD separately on each machine separately.
  - Communicate each machines anser to a central “parameter server”, and (by convexity) average the final answer from each machine.
  - Then can repeat.

Averaging: How good is it?

- Question: What if there isn't enough data on each machine?
- How much data do we need on each machine?
  Roughly, we need κ data points per machine.
- Theorem: With "enough" data on each machine, then one can just run one pass of SGD separately on each machine and then average their answers. This will be optimal statistically (e.g. in terms of generalization).

Hogwild