Speculative Approximations for Terascale Distributed Gradient Descent Optimization

Chengjie Qin and **Florin Rusu**
University of California, Merced

May 31, 2015

Terascale Analytics

- Massive amounts of example data, e.g., 10 billion
- Highly-dimensional models, e.g., 50 to 600 million

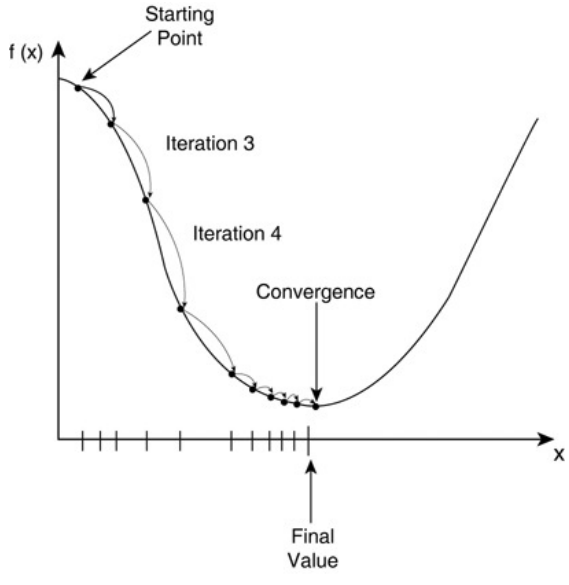
Analytics task	**Objective function**
Logistic Regression (LR)	$\sum_{(\vec{x}_i,y_i)\in \text{data}} \log\left(1+e^{-y_i\vec{w}^T\vec{x}_i}\right)+\mu\|\vec{w}\|_1$
Classification (Support Vector Machines - SVM)	$\sum_{(\vec{x}_i,y_i)\in \text{data}} \left(1-y_i\vec{w}^T\vec{x}_i\right)+\mu\|\vec{w}\|_1$
Recommendation (Low-Rank Matrix Factorization - LMF)	$\sum_{(i,j)\in\Omega}\left(\vec{L}_i^T\vec{R}_j-M_{ij}\right)^2+\mu\|\vec{\vec{L}},\vec{\vec{R}}\|_F^2$
Labeling (Conditional Random Fields - CRF)	$\sum_{(\vec{x}_i,y_i)\in \text{data}}\left[\sum_j \vec{w}_j F_j(\vec{x}_i,y_i)-\log Z(\vec{x}_i)\right]$

Table 1: [Feng, Kumar, Recht, and Re: *Towards a Unified Architecture for in-RDBMS Analytics*, SIGMOD 2012]

Agenda

- Gradient Descent Optimization
- Speculative Iterations
- Intra-Iteration Approximation
- Experiments
- Conclusions

Gradient Descent Optimization



http://www.yaldex.com/game-development/1592730043_ch18lev1sec4.html

$$min_{\vec{w}\in\mathbb{R}^d}\left\{\Lambda(\vec{w}) = \sum_{(\vec{x}_i,y_i)\in \text{data}} f\left(\vec{w},\vec{x}_i;y_i\right)\right\}$$

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \alpha^{(k)}\nabla\Lambda\left(\vec{w}^{(k)}\right)$$

$\Lambda(\vec{w})$ is the loss

$\nabla\Lambda(\vec{w}) = \left[\frac{\partial\Lambda(\vec{w})}{\partial w_1},\ldots,\frac{\partial\Lambda(\vec{w})}{\partial w_d}\right]$ is the gradient

$\alpha^{(k)}$ is step size or learning rate

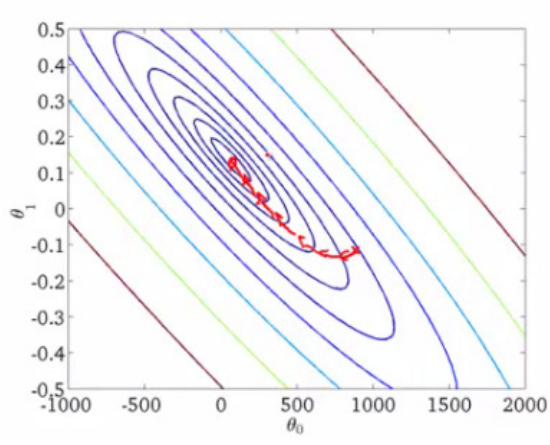
$\vec{w}^{(0)}$ is the starting point (random)

- Convergence to minimum guaranteed for convex objective function

Batch Gradient Descent

Input: $\{(\vec{x}_j, y_j)\}_{1 \le j \le N}$, f, Λ, $\nabla\Lambda$, $\vec{w}^{(0)}$, $\alpha^{(0)}$
Output: $\vec{w}^{(k-1)}$

1: Let $k = 1$
2: **while** (**true**) **do**
3: **if** convergence$(\{\Lambda(\vec{w}^{(l)})\}_{0 \le l < k})$ **then break**
4: Compute gradient: $\nabla\Lambda(\vec{w}^{(k-1)})\{(\vec{x}_j, y_j)\}_{1 \le j \le N}$
5: Determine step size $\alpha^{(k)}$
6: Update model: $\vec{w}^{(k)} = \vec{w}^{(k-1)} - \alpha^{(k)}\nabla\Lambda(\vec{w}^{(k-1)})$
7: Let $k = k + 1$
8: **end while**
9: **return** $\vec{w}^{(k-1)}$



http://www.holehouse.org/mlclass/17_Large_Scale_Machine_Learning.html

Compute gradient

```
SELECT SUM(∂f/∂w_1 (w, x; y)),
       ...,
       SUM(∂f/∂w_d (w, x; y))
FROM data
```

Determine step size & convergence

```
SELECT SUM(f(w, x; y))
FROM data
```

Batch Gradient Descent

Input: $\{(\vec{x}_j, y_j)\}_{1 \le j \le N}$, f, Λ, $\nabla\Lambda$, $\vec{w}^{(0)}$, $\alpha^{(0)}$
Output: $\vec{w}^{(k-1)}$

1: Let $k = 1$
2: **while** (**true**) **do**
3: **if** convergence($\{\Lambda(\vec{w}^{(l)})\}_{0 \le l < k}$) **then break**
4: Compute gradient: $\nabla\Lambda(\vec{w}^{(k-1)})\{(\vec{x}_j, y_j)\}_{1 \le j \le N}$
5: Determine step size $\alpha^{(k)}$
6: Update model: $\vec{w}^{(k)} = \vec{w}^{(k-1)} - \alpha^{(k)}\nabla\Lambda(\vec{w}^{(k-1)})$
7: Let $k = k + 1$
8: **end while**
9: **return** $\vec{w}^{(k-1)}$

Line search: finding optimal or good-enough (Wolfe conditions) step size $\alpha^{(k)}$ requires multiple passes over `data` at each iteration

Compute gradient

```
SELECT SUM(∂f/∂w_1(w⃗, x⃗, y)),
       ...,
       SUM(∂f/∂w_d(w⃗, x⃗, y))
FROM data
```

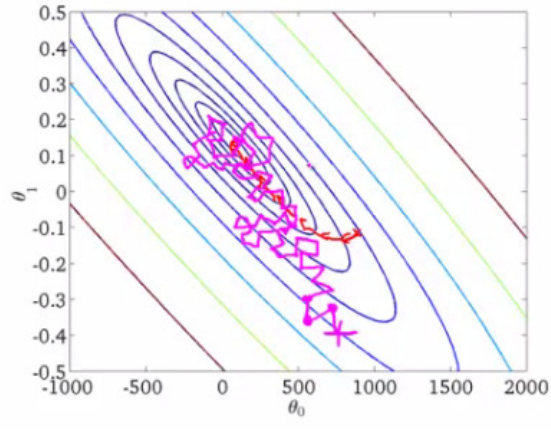
Determine step size & convergence

```
SELECT SUM(f(w⃗, x⃗, y))
FROM data
```

Stochastic Gradient Descent

Input: $\{(\vec{x}_j, y_j)\}_{1 \le j \le N}$, f, ∇f, $\vec{w}^{(0)}$, $\alpha^{(0)}$
Output: $\vec{w}^{(k-1)}$

1: Let $k = 1$
2: **while** (**true**) **do**
3: **if** convergence$(\{\Lambda(\vec{w}^{(l)})\}_{0 \le l < k})$ **then break**
4: **for each example** $(\vec{x}_{\eta^{(i)}}, y_{\eta^{(i)}})$ **do**
5: Approximate gradient: $\nabla f\left(\vec{w}^{(k)}_{(i-1)}, \vec{x}_{\eta^{(i)}}; y_{\eta^{(i)}}\right)$
6: $\vec{w}^{(k)}_{(i)} = \vec{w}^{(k)}_{(i-1)} - \alpha^{(k)} \nabla f\left(\vec{w}^{(k)}_{(i-1)}, \vec{x}_{\eta^{(i)}}; y_{\eta^{(i)}}\right)$
7: **end for**
8: Update step size $\alpha^{(k)}$
9: Let $k = k + 1$
10: **end while**
11: **return** $\vec{w}^{(k-1)}$



http://www.holehouse.org/mlclass/17_Large_Scale_Machine_Learning.html

Approximate gradient

SELECT $\frac{\partial f}{\partial w_1}\left(\vec{w},\ \vec{x}_{\eta^{(i)}};\ y_{\eta^{(i)}}\right),\ \ldots$
FROM data

Update model

UPDATE W
SET $w_1 = w_1 - \alpha^{(k)} \cdot \frac{\partial f}{\partial w_1}\left(\vec{w},\ \vec{x}_{\eta^{(i)}};\ y_{\eta^{(i)}}\right),\ \ldots$

Stochastic Gradient Descent

Input: $\{(\vec{x}_j, y_j)\}_{1 \le j \le N}$, f, ∇f, $\vec{w}^{(0)}$, $\alpha^{(0)}$
Output: $\vec{w}^{(k-1)}$

1: Let $k = 1$
2: **while** (**true**) **do**
3: **if** convergence($\{\Lambda(\vec{w}^{(l)})\}_{0 \le l < k}$) **then break**
4: **for each example** $(\vec{x}_{\eta^{(i)}}, y_{\eta^{(i)}})$ **do**
5: Approximate gradient: $\nabla f\left(\vec{w}^{(k)}_{(i-1)}, \vec{x}_{\eta^{(i)}}; y_{\eta^{(i)}}\right)$
6: $\vec{w}^{(k)}_{(i)} = \vec{w}^{(k)}_{(i-1)} - \alpha^{(k)} \nabla f\left(\vec{w}^{(k)}_{(i-1)}, \vec{x}_{\eta^{(i)}}; y_{\eta^{(i)}}\right)$
7: **end for**
8: Update step size $\alpha^{(k)}$
9: Let $k = k + 1$
10: **end while**
11: **return** $\vec{w}^{(k-1)}$

- Executed for every example (or mini-batch)
- Requires random order (at every iteration), i.e., cursor

Approximate gradient

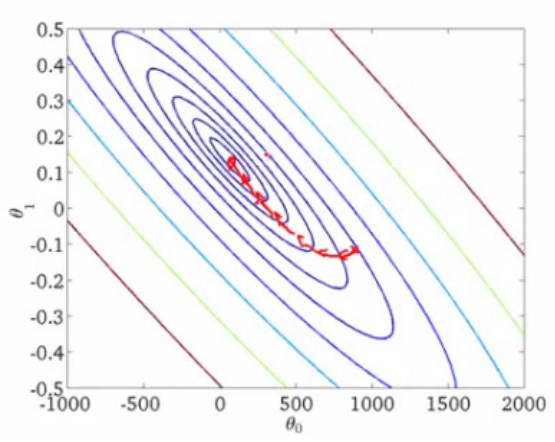
```
SELECT ∂f/∂w1 (w, x_η(i); y_η(i)), ...
FROM data
```

Update model

```
UPDATE W
SET w1 = w1 − α(k) · ∂f/∂w1 (w, x_η(i); y_η(i)), ...
```

Batch and Stochastic Gradient Descent

Batch Gradient Descent (BGD)

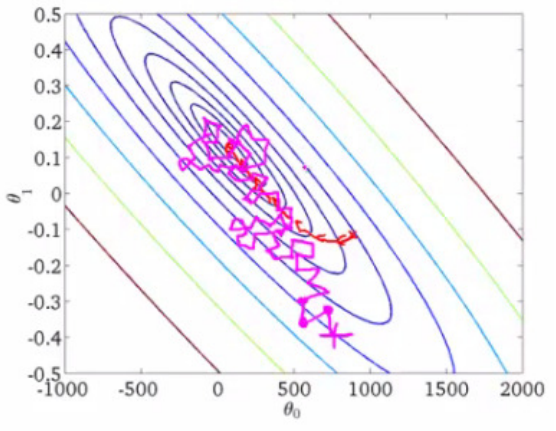


http://www.holehouse.org/mlclass/17_Large_Scale_Machine_Learning.html

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \alpha^{(k)} \nabla \Lambda \left(\vec{w}^{(k)} \right)$$

- Exact gradient computation
- Single step for one iteration
- Faster convergence close to minimum

Stochastic Gradient Descent (SGD)



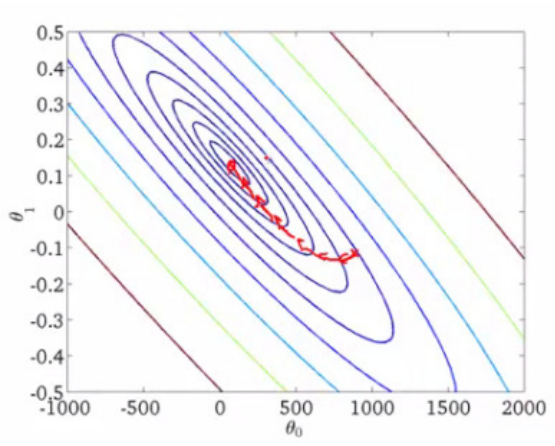
http://www.holehouse.org/mlclass/17_Large_Scale_Machine_Learning.html

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \beta^{(k)} \nabla f \left(\vec{w}^{(k)}, \vec{x}_{\eta^{(k)}}; y_{\eta^{(k)}} \right)$$

- Approximate gradient at data point
- One step for each random data point
- Faster convergence far from minimum

Distributed Gradient Descent

Batch Gradient Descent (BGD)

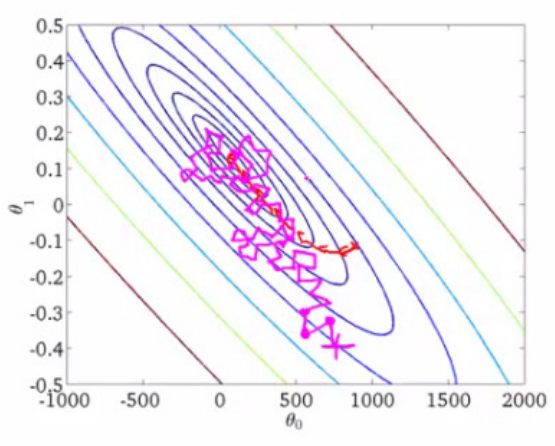


http://www.holehouse.org/mlclass/17_Large_Scale_Machine_Learning.html

$$\nabla \sum f(\vec{w}, \vec{x}_i; y_i) = \sum \nabla f(\vec{w}, \vec{x}_i; y_i)$$

- Linearly separable loss
- Trivially parallel

Stochastic Gradient Descent (SGD)



http://www.holehouse.org/mlclass/17_Large_Scale_Machine_Learning.html

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \beta^{(k)} \nabla f\left(\vec{w}^{(k)}, \vec{x}_{\eta^{(k)}}; y_{\eta^{(k)}}\right)$$

- Updates not commutative: **IGNORE!!!**
- Model averaging

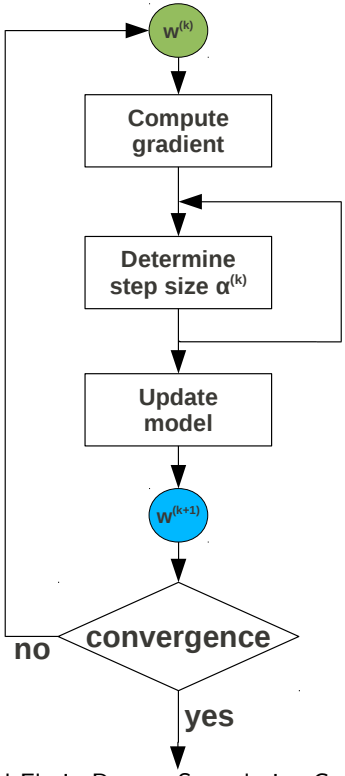
Agenda

- Gradient Descent Optimization
- Speculative Iterations
- Intra-Iteration Approximation
- Experiments
- Conclusions

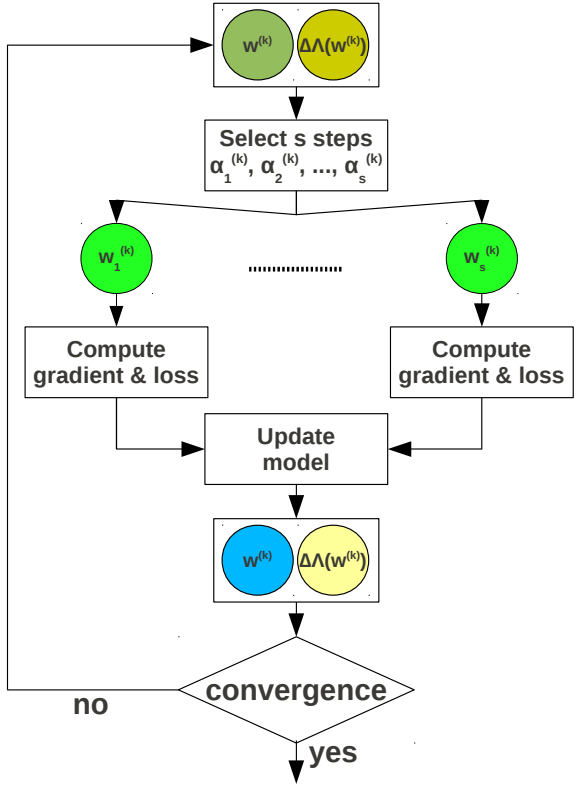
Speculative Iterations

Main idea: overlap gradient and loss computation for multiple step sizes

Sequential Gradient Descent



Speculative Gradient Descent



Speculative BGD

Input: $\{(\vec{x}_j, y_j)\}_{1 \le j \le N}$, Λ, $\nabla\Lambda$, $\vec{w}^{(0)}$, $\nabla\Lambda(\vec{w}^{(0)})$, s, Φ
Output: $\vec{w}^{(k)}$

1: Let $k = 1$
2: **while** (**true**) **do**
3: Draw s step sizes $\{\alpha_1, \ldots, \alpha_s\}$ from distribution Φ
4: Let $\vec{w}_i = \vec{w}^{(k-1)} - \alpha_i \nabla\Lambda(\vec{w}^{(k-1)})$, $1 \le i \le s$
5: **for each example** $(\vec{x}_j, y_j)$ **do**
6: **for each model** $\vec{w}_i$ **do in parallel**
7: Compute gradient: $\nabla\Lambda(\vec{w}_i)\{(\vec{x}_j, y_j)\}$
8: Compute loss: $\Lambda(\vec{w}_i, \vec{x}_j; y_j)$
9: **end for**
10: **end for**
11: Let $\vec{w}^{(k)} = min_{\Lambda(\vec{w}_i)}\{\vec{w}_i\}$, $1 \le i \le s$
12: **if** convergence$(\{\Lambda(\vec{w}^{(l)})\}_{0 \le l < k})$ **then break**
13: Update step distribution Φ and number of steps s
14: Let $k = k + 1$
15: **end while**
16: **return** $\vec{w}^{(k)}$

Speculative BGD

Input: $\{(\vec{x}_j, y_j)\}_{1 \le j \le N}$, Λ, $\nabla\Lambda$, $\vec{w}^{(0)}$, $\nabla\Lambda(\vec{w}^{(0)})$, s, Φ
Output: $\vec{w}^{(k)}$

1: Let $k = 1$
2: **while (true) do**
3: Draw s step sizes $\{\alpha_1, \ldots, \alpha_s\}$ from distribution Φ
4: Let $\vec{w}_i = \vec{w}^{(k-1)} - \alpha_i \nabla\Lambda(\vec{w}^{(k-1)})$, $1 \le i \le s$
5: **for each example** $(\vec{x}_j, y_j)$ **do**
6: **for each model** $\vec{w}_i$ **do in parallel**
7: Compute gradient: $\nabla\Lambda(\vec{w}_i)\{(\vec{x}_j, y_j)\}$
8: Compute loss: $\Lambda(\vec{w}_i, \vec{x}_j; y_j)$
9: **end for**
10: **end for**
11: Let $\vec{w}^{(k)} = min_{\Lambda(\vec{w}_i)}\{\vec{w}_i\}$, $1 \le i \le s$
12: **if** convergence$(\{\Lambda(\vec{w}^{(l)})\}_{0 \le l < k})$ **then break**
13: Update step distribution Φ and number of steps s
14: Let $k = k + 1$
15: **end while**
16: **return** $\vec{w}^{(k)}$

```
SELECT
    SUM(∂f/∂w_1 (w_1, x; y)),
    ...,
    SUM(∂f/∂w_d (w_1, x; y)),
    SUM(f (w_1, x; y)),
    ...
    SUM(∂f/∂w_1 (w_s, x; y)),
    ...,
    SUM(∂f/∂w_d (w_s, x; y)),
    SUM(f (w_s, x; y))
FROM data
```

Speculative BGD

Input: $\{(\vec{x}_j, y_j)\}_{1 \le j \le N}$, Λ, $\nabla\Lambda$, $\vec{w}^{(0)}$, $\nabla\Lambda(\vec{w}^{(0)})$, s, Φ
Output: $\vec{w}^{(k)}$

1: Let $k = 1$
2: **while** (**true**) **do**
3: Draw s step sizes $\{\alpha_1, \ldots, \alpha_s\}$ from distribution Φ
4: Let $\vec{w}_i = \vec{w}^{(k-1)} - \alpha_i \nabla\Lambda(\vec{w}^{(k-1)})$, $1 \le i \le s$
5: **for each example** $(\vec{x}_j, y_j)$ **do**
6: **for each model** $\vec{w}_i$ **do in parallel**
7: Compute gradient: $\nabla\Lambda(\vec{w}_i)\{(\vec{x}_j, y_j)\}$
8: Compute loss: $\Lambda(\vec{w}_i, \vec{x}_j; y_j)$
9: **end for**
10: **end for**
11: Let $\vec{w}^{(k)} = min_{\Lambda(\vec{w}_i)}\{\vec{w}_i\}$, $1 \le i \le s$
12: **if** convergence$(\{\Lambda(\vec{w}^{(l)})\}_{0 \le l < k})$ **then break**
13: Update step distribution Φ and number of steps s
14: Let $k = k + 1$
15: **end while**
16: **return** $\vec{w}^{(k)}$

How to choose s?

- adaptively and dynamically at runtime
- exponential increase to saturate available resources

How to choose $\{\alpha_1, \ldots, \alpha_s\}$?

- sample from distribution Φ
- Bayesian process to update distribution Φ at runtime

Speculative SGD

1: **while** (**true**) **do**
2: Draw s step sizes $\{\alpha_1, \ldots, \alpha_s\}$ from distribution Φ
3: Let $\vec{w}_{il} = \vec{w}_i^{(k-1)}$, $1 \leq i, l \leq s$
4: **for each example** $(\vec{x}_{\eta^{(j)}}, y_{\eta^{(j)}})$ **do**
5: **for each model** $\vec{w}_{il}$ **do in parallel**
6: Approximate gradient: $\nabla f\left(\vec{w}_{il}, \vec{x}_{\eta^{(j)}}; y_{\eta^{(j)}}\right)$
7: Update: $\vec{w}_{il} = \vec{w}_{il} - \alpha_l \nabla f\left(\vec{w}_{il}, \vec{x}_{\eta^{(j)}}; y_{\eta^{(j)}}\right)$
8: **end for**
9: **for each original model** $\vec{w}_i^{(k-1)}$ **do in parallel**
10: Compute loss: $f\left(\vec{w}_i^{(k-1)}, \vec{x}_{\eta^{(j)}}; y_{\eta^{(j)}}\right)$
11: **end for**
12: Let $\vec{w}^{(k)} = min_{f(\vec{w}_i^{(k-1)})}\{\vec{w}_i^{(k-1)}\}$, $1 \leq i \leq s$
13: Let m be the index of the minimum loss $f(\vec{w}_i^{(k-1)})$
14: **if** model_convergence$(\{f(\vec{w}^{(l)})\}_{0 \leq l < k})$ **then break**
15: Update step distribution Φ and number of steps s
16: Let $\vec{w}_i^{(k)} = \vec{w}_{mi}$, $1 \leq i \leq s$
17: **end while**

Speculative SGD

1: **while** (**true**) **do**
2: Draw s step sizes $\{\alpha_1, \ldots, \alpha_s\}$ from distribution Φ
3: Let $\vec{w}_{il} = \vec{w}_i^{(k-1)}$, $1 \leq i, l \leq s$
4: **for each example** $(\vec{x}_{\eta^{(j)}}, y_{\eta^{(j)}})$ **do**
5: **for each model** $\vec{w}_{il}$ **do in parallel**
6: Approximate gradient: $\nabla f\left(\vec{w}_{il}, \vec{x}_{\eta^{(j)}}; y_{\eta^{(j)}}\right)$
7: Update: $\vec{w}_{il} = \vec{w}_{il} - \alpha_l \nabla f\left(\vec{w}_{il}, \vec{x}_{\eta^{(j)}}; y_{\eta^{(j)}}\right)$
8: **end for**
9: **for each original model** $\vec{w}_i^{(k-1)}$ **do in parallel**
10: Compute loss: $f\left(\vec{w}_i^{(k-1)}, \vec{x}_{\eta^{(j)}}; y_{\eta^{(j)}}\right)$
11: **end for**
12: Let $\vec{w}^{(k)} = min_{f(\vec{w}_i^{(k-1)})}\{\vec{w}_i^{(k-1)}\}$, $1 \leq i \leq s$
13: Let m be the index of the minimum loss $f(\vec{w}_i^{(k-1)})$
14: **if** model_convergence$(\{f(\vec{w}^{(l)})\}_{0 \leq l < k})$ **then break**
15: Update step distribution Φ and number of steps s
16: Let $\vec{w}_i^{(k)} = \vec{w}_{mi}$, $1 \leq i \leq s$
17: **end while**

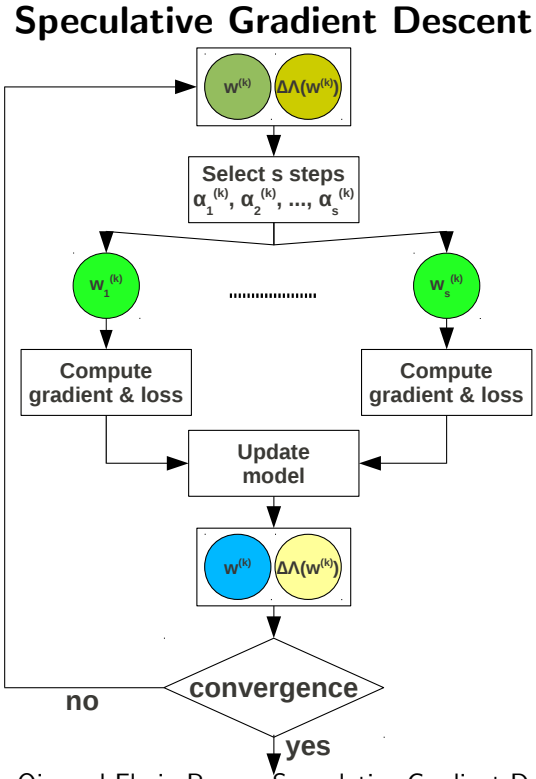
- input has s models
- s^2 models are updated
- keep s models corresponding to original with minimum loss

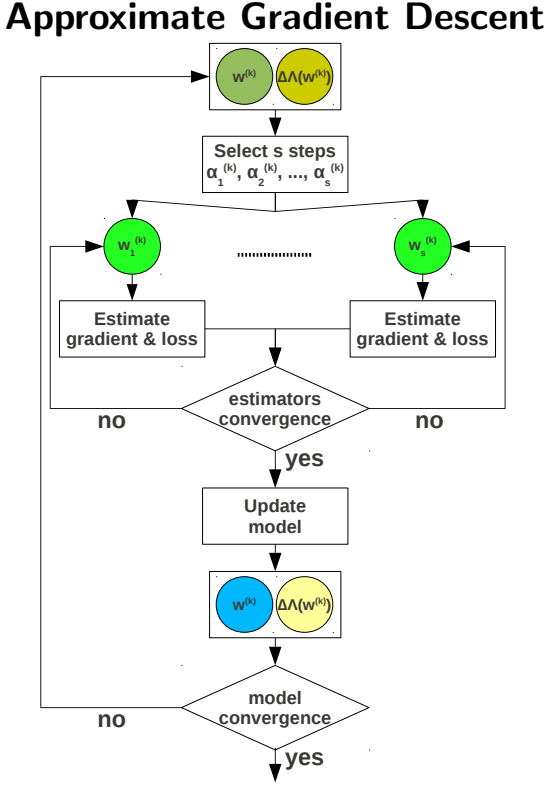
Agenda

- Gradient Descent Optimization
- Speculative Iterations
- Intra-Iteration Approximation
- Experiments
- Conclusions

Intra-Iteration Approximation

Main idea: apply online aggregation (OLA) sampling to speed-up the execution of a speculative iteration





Approximate BGD

1: **for each example** $(\vec{x}_{\eta(j)}, y_{\eta(j)})$ **do**
2: **for each active model** $\vec{w}_i$ **do in parallel**
3: Estimate gradient $G_i[est, std]$: $\nabla\Lambda(\vec{w}_i)\{(\vec{x}_{\eta(j)}, y_{\eta(j)})\}$
4: Estimate loss $L_i[est, std]$: $\Lambda(\vec{w}_i, \vec{x}_{\eta(j)}; y_{\eta(j)})$
5: **end for**
6: **if** estimators_convergence_check(j) **then**
7: Prune out models: *Stop Loss*$(\{L_i[est, std]\}_{i \leq s}, \varepsilon_1)$
8: Let t be the number of remaining models, i.e., *active*
9: **if** *Stop Gradient*$(\{G_{tl}[est, std]\}_{l \leq d}, \varepsilon_2)$ **then break**
10: Let $s = t$
11: **end if**
12: **end for**

Approximate BGD

1: **for each example** $(\vec{x}_{\eta^{(j)}}, y_{\eta^{(j)}})$ **do**
2: **for each active model** $\vec{w}_i$ **do in parallel**
3: Estimate gradient $G_i[est, std]$: $\nabla\Lambda(\vec{w}_i)\{(\vec{x}_{\eta^{(j)}}, y_{\eta^{(j)}})\}$
4: Estimate loss $L_i[est, std]$: $\Lambda(\vec{w}_i, \vec{x}_{\eta^{(j)}}; y_{\eta^{(j)}})$
5: **end for**
6: **if** estimators_convergence_check(j) **then**
7: Prune out models: *Stop Loss*$(\{L_i[est, std]\}_{i \le s}, \varepsilon_1)$
8: Let t be the number of remaining models, i.e., *active*
9: **if** *Stop Gradient*$(\{G_{tl}[est, std]\}_{l \le d}, \varepsilon_2)$ **then break**
10: Let $s = t$
11: **end if**
12: **end for**

```
SELECT
  ..., SUM(f_q), ...
FROM data AS D
SELECT
  ..., Z_q=SUM(f_q), ...
  ..., Z_q^2=SUM(f_q^2), ...
FROM sample AS S
```

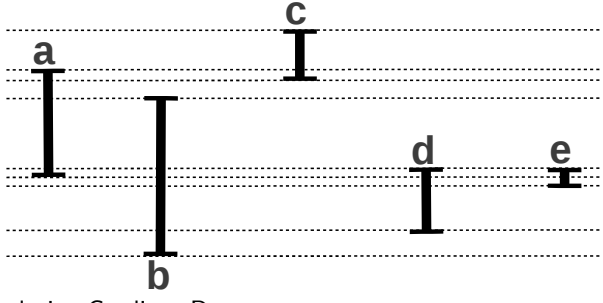
- estimator: $Z_q \cdot \frac{|D|}{|S|}$
- variance: $\frac{|D|(|D|-|S|)}{|S|^2(|S|-1)}\left(|S|Z_q^2 - \left(Z_q\right)^2\right)$

OLA sampling

- no pre-determined sample size
- avoid correlation between estimators by random data shuffling (at runtime)
- continuous sampling at runtime until estimators converge

Approximate BGD

1: **for each example** $(\vec{x}_{\eta(j)}, y_{\eta(j)})$ **do**
2: **for each active model** $\vec{w}_i$ **do in parallel**
3: Estimate gradient $G_i[est, std]$: $\nabla\Lambda(\vec{w}_i)\{(\vec{x}_{\eta(j)}, y_{\eta(j)})\}$
4: Estimate loss $L_i[est, std]$: $\Lambda(\vec{w}_i, \vec{x}_{\eta(j)}; y_{\eta(j)})$
5: **end for**
6: **if** estimators_convergence_check(j) **then**
7: Prune out models: *Stop Loss*$(\{L_i[est, std]\}_{i \leq s}, \varepsilon_1)$
8: Let t be the number of remaining models, i.e., *active*
9: **if** *Stop Gradient*$(\{G_{tl}[est, std]\}_{l \leq d}, \varepsilon_2)$ **then break**
10: Let $s = t$
11: **end if**
12: **end for**



- Identify step size with minimum loss based on estimate & confidence bounds
- Interacting estimators
- Exact pruning
- Approximate pruning

Approximate BGD

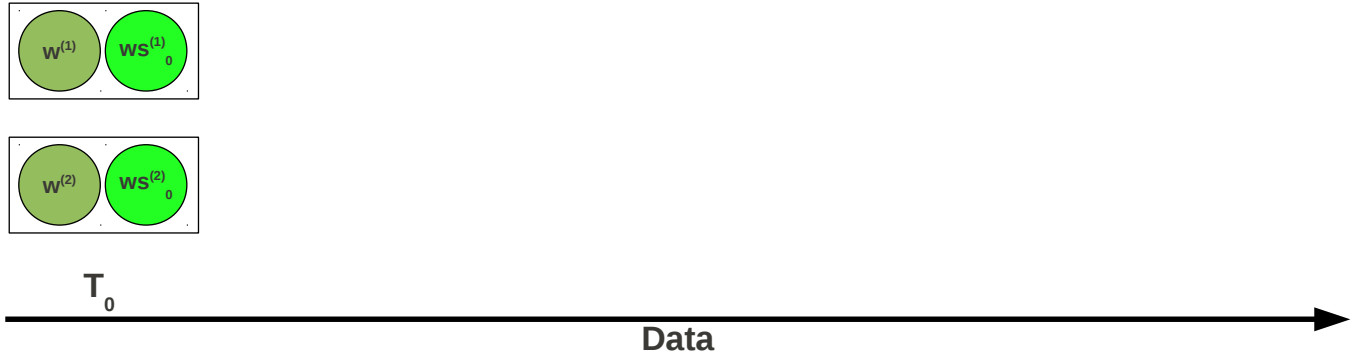
1: **for each example** $(\vec{x}_{\eta(j)}, y_{\eta(j)})$ **do**
2: **for each active model** $\vec{w}_i$ **do in parallel**
3: Estimate gradient $G_i[est, std]$: $\nabla\Lambda(\vec{w}_i)\{(\vec{x}_{\eta(j)}, y_{\eta(j)})\}$
4: Estimate loss $L_i[est, std]$: $\Lambda(\vec{w}_i, \vec{x}_{\eta(j)}; y_{\eta(j)})$
5: **end for**
6: **if** estimators_convergence_check(j) **then**
7: Prune out models: *Stop Loss*$(\{L_i[est, std]\}_{i \leq s}, \varepsilon_1)$
8: Let t be the number of remaining models, i.e., *active*
9: **if** *Stop Gradient*$(\{G_{tl}[est, std]\}_{l \leq d}, \varepsilon_2)$ **then break**
10: Let $s = t$
11: **end if**
12: **end for**

- One estimator for each dimension in model $\vec{w}$
- Grouped estimators

When to stop?

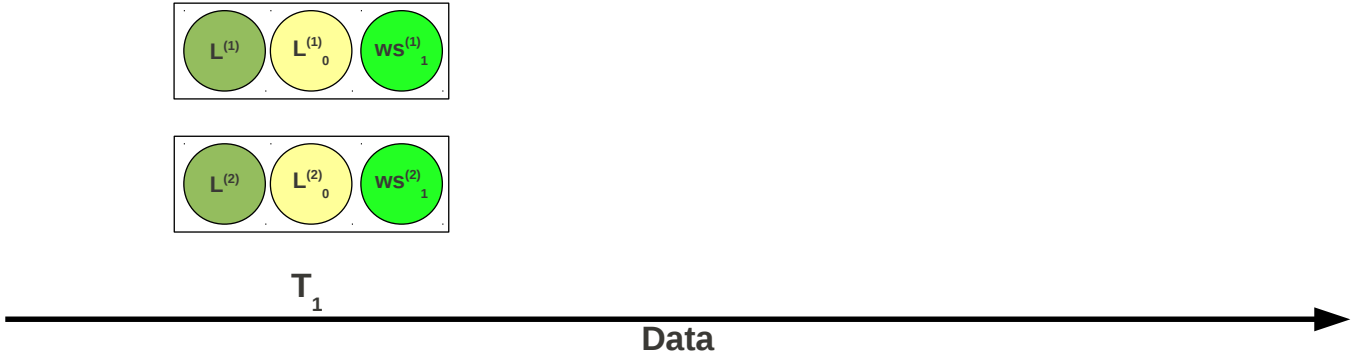
- all estimators converge (million dimensions)
- percentage of estimators converge, e.g., 90%
- unified convergence threshold, e.g., $\varepsilon' = d \cdot \varepsilon$

Approximate SGD



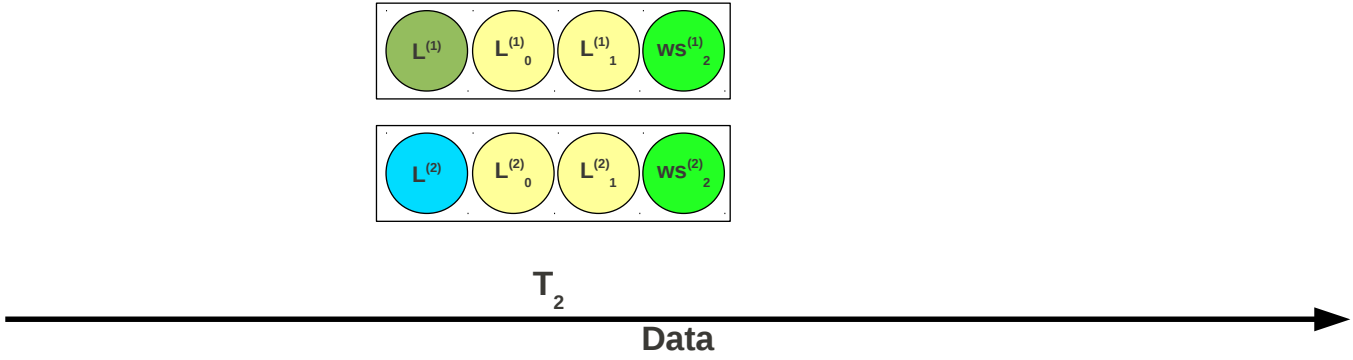
- 2 models, 4 configurations
- estimate loss for 2 originals
- update & estimate loss for 2 active

Approximate SGD



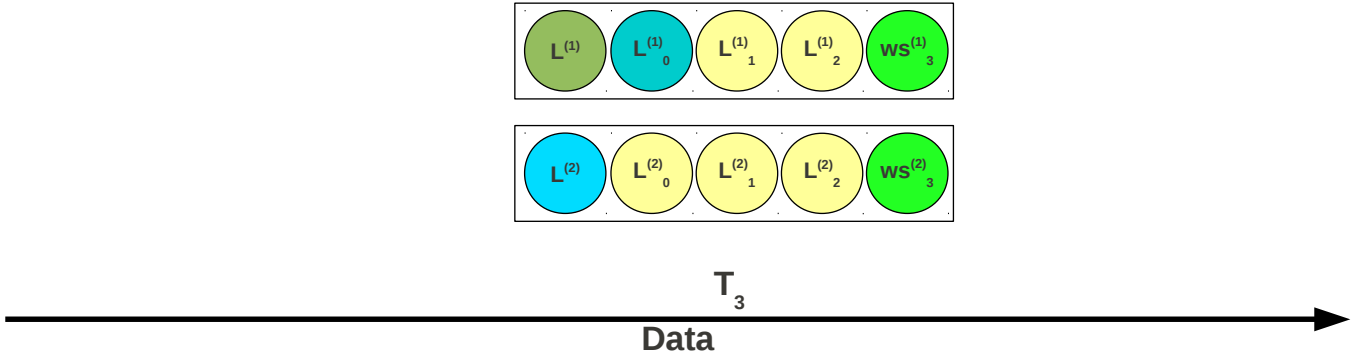
- 2 original loss estimates
- 2 previous snapshot estimates
- 2 active snapshots

Approximate SGD



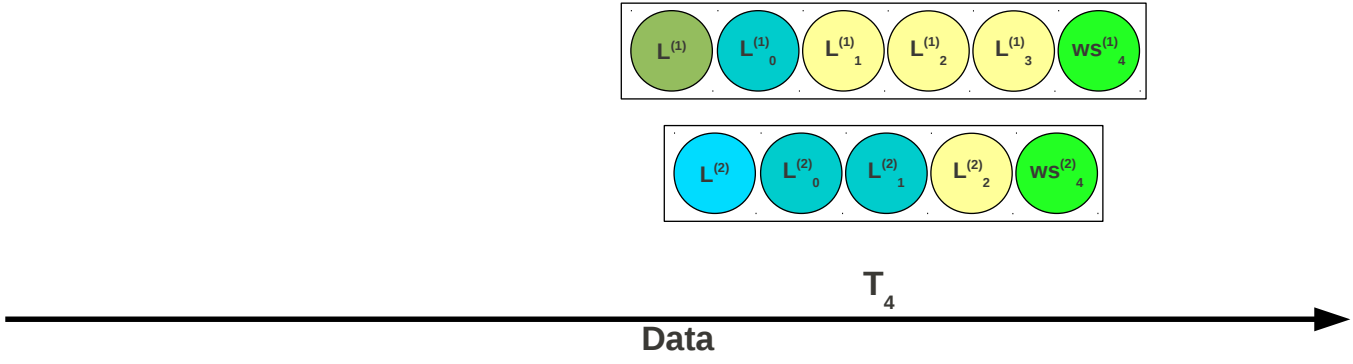
- loss estimate for original $w^{(2)}$ converge
- add 2 new snapshots

Approximate SGD



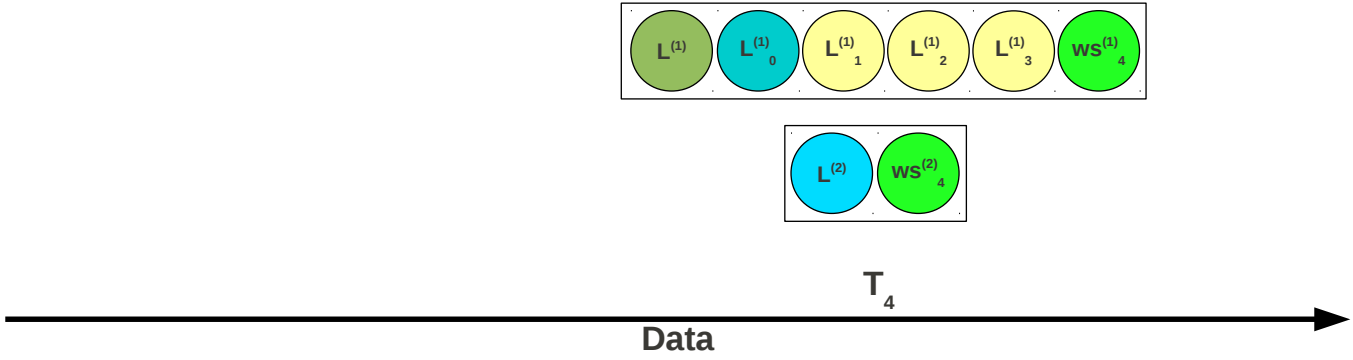
- loss estimate for snapshot $ws_0^{(1)}$ converge
- add 2 new snapshots

Approximate SGD



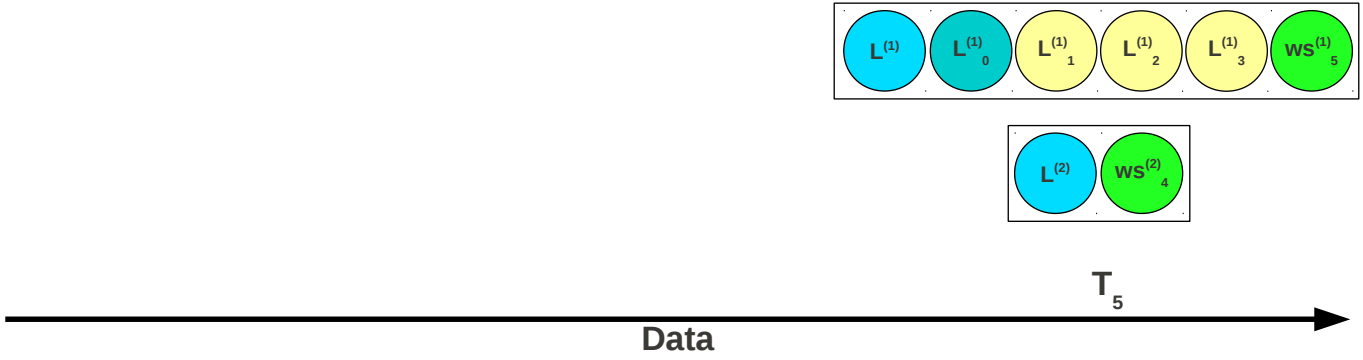
- loss estimate for snapshots $ws_0^{(2)}$ and $ws_1^{(2)}$ converge; thus, active model $ws^{(2)}$ converge
- add 1 new snapshot $ws_4^{(1)}$

Approximate SGD



- original model $w^{(2)}$ and corresponding snapshot $ws_4^{(2)}$ are finalized
- only original model $w^{(1)}$ is active

Approximate SGD

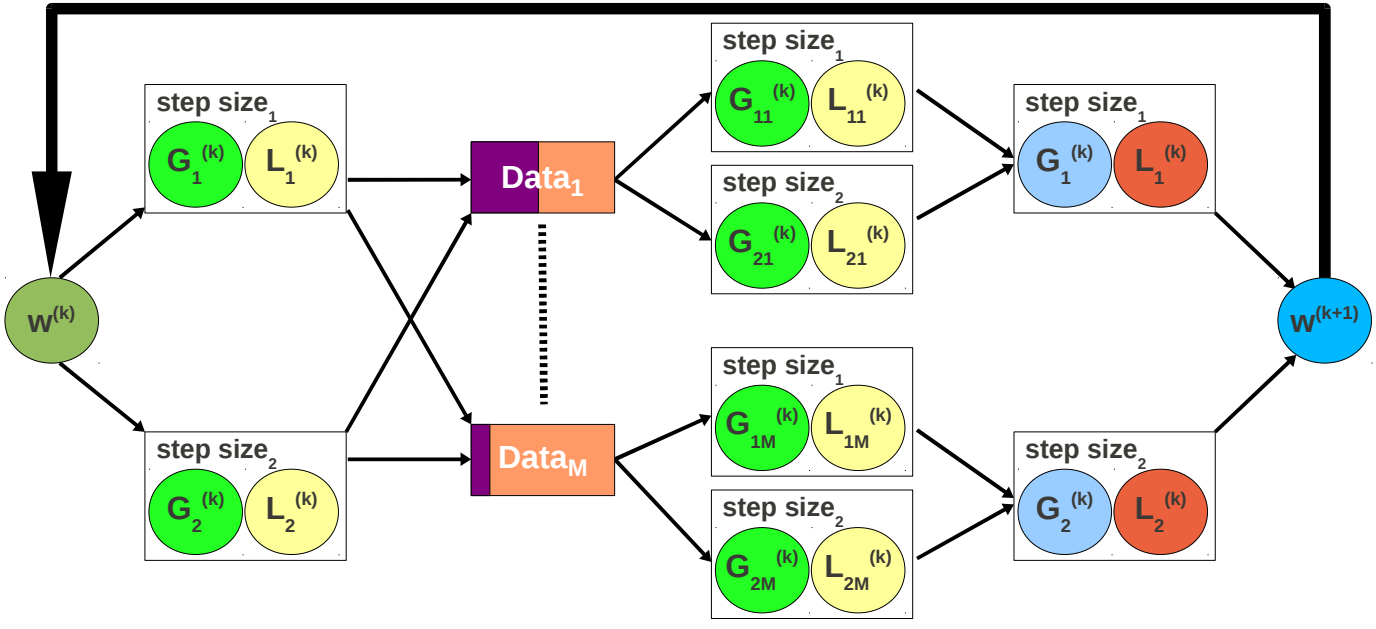


- loss estimate for original $w^{(1)}$ converge
- loss estimate for $w^{(2)}$ is smaller than loss estimate for $w^{(1)}$, i.e., $L^{(2)} < L^{(1)}$

Approximate SGD

- select $ws_4^{(2)}$ as starting model for next iteration

Enhanced Distributed Gradient Descent

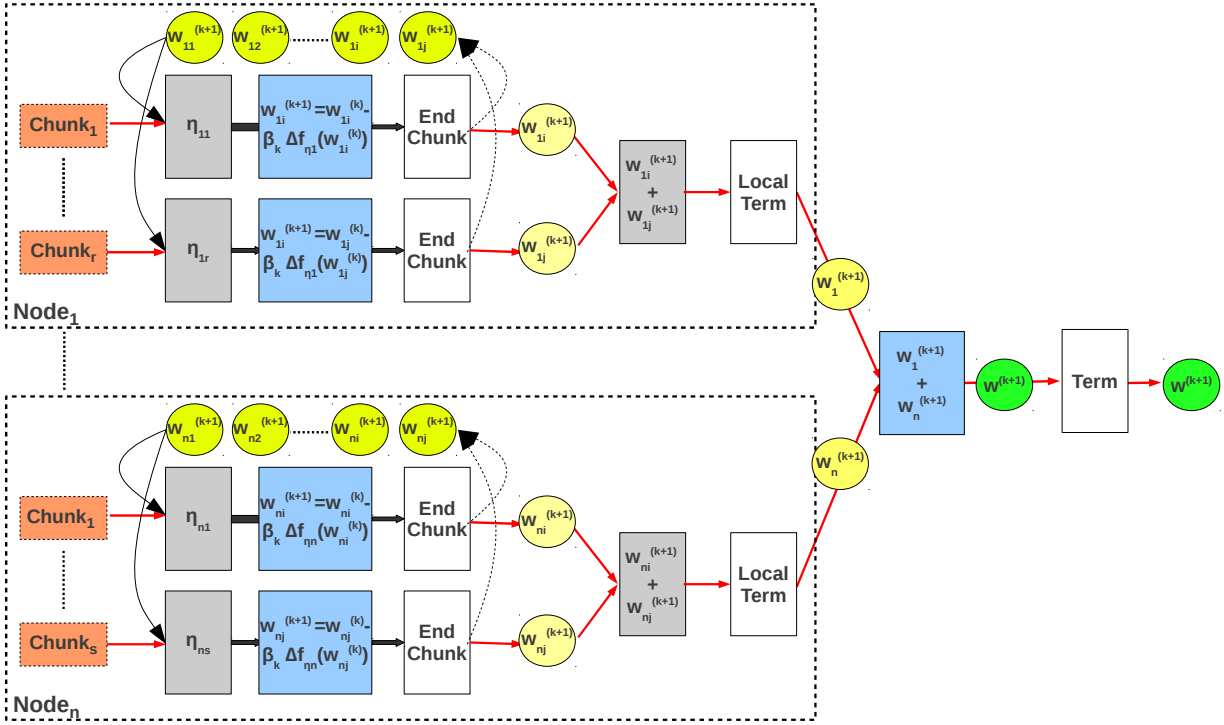


- Data partitioning parallelism
- Model merging: linear loss functions (commutative & associative)

Agenda

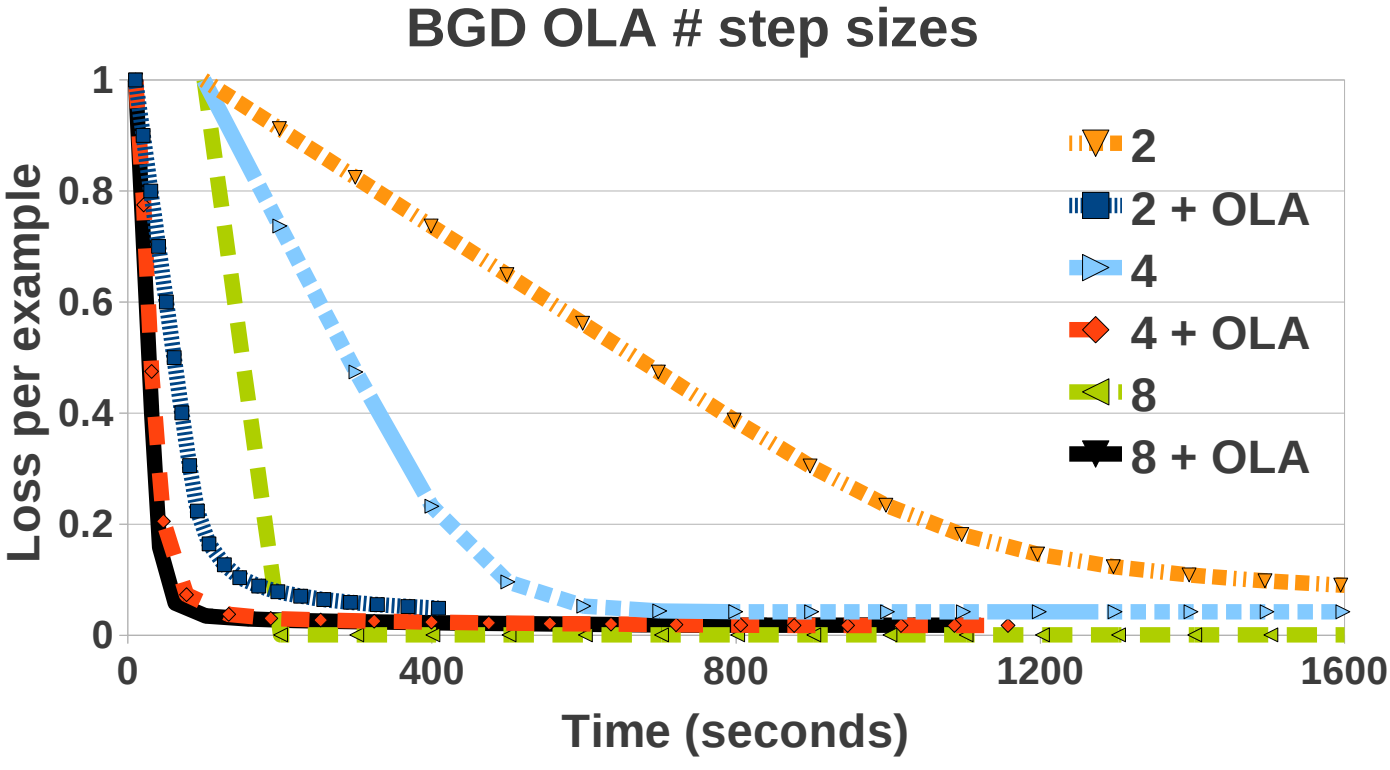
- Gradient Descent Optimization
- Speculative Iterations
- Intra-Iteration Approximation
- Experiments
- Conclusions

Implementation as GLA in GLADE



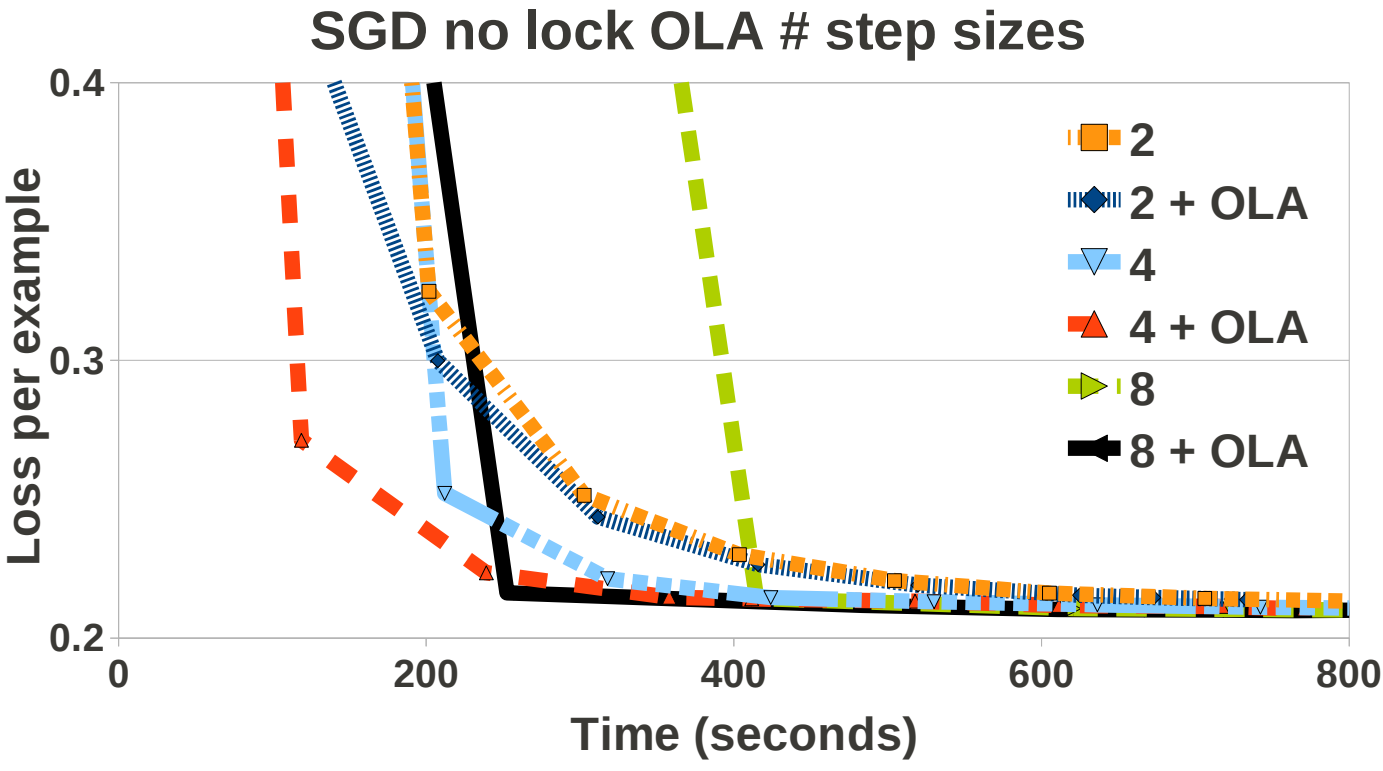
- Extended UDA interface with support for chunk and thread/process aggregation/OLA

SVM on `classify50M`



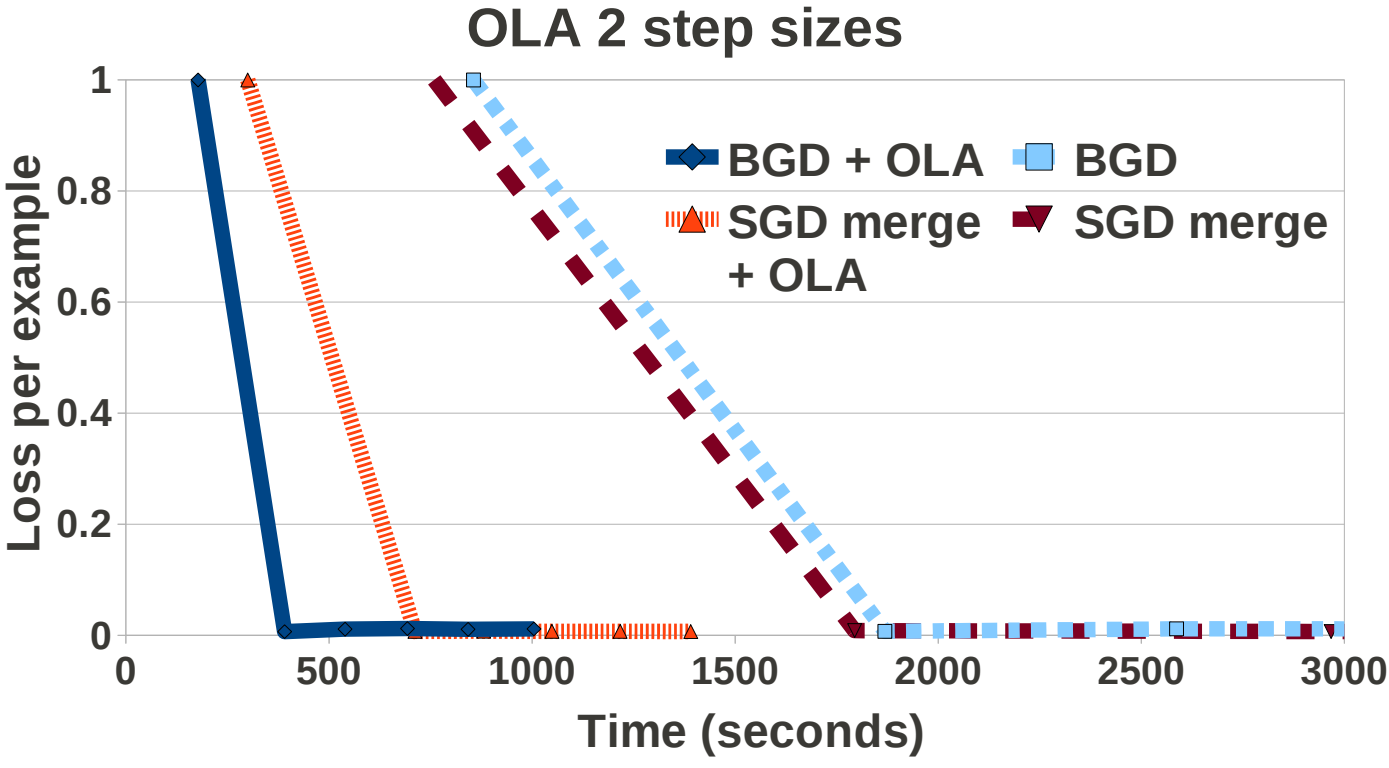
- 50M examples, 200 dimensions, 136GB

LR on `classify50M`



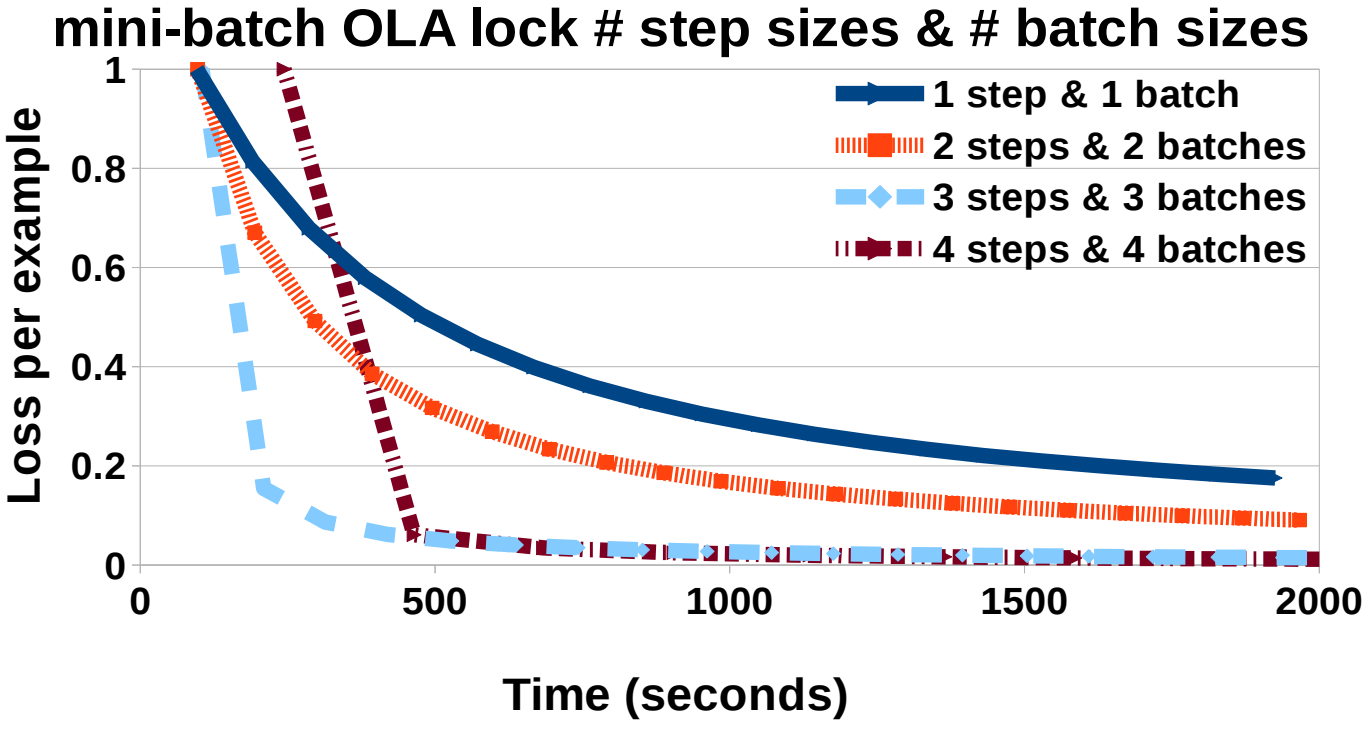
- 50M examples, 200 dimensions, 136GB

SVM on `splice`



- 50M examples, 13M dimensions, 3.2TB

Two-Parameters LR on `classify50M`



- 50M examples, 200 dimensions, 136GB

Conclusions

- Gradient Descent Optimization
- Speculative Iterations
- Intra-Iteration Approximation
- Experiments

Questions ???