



Distributed ML Intro

Ильнур Шугаев

Table of Contents

1. Large Scale ML

- Learning with gradient descent

- Learning with large training sets

2. Distributed ML

3. General Purpose Distributed Computing

4. Natively Distributed ML Systems

Problem formulation¹²

1 Dataset $X^n = \{z_i\}_{i=1}^n$, где $z_i = (x_i, y_i) \sim P(z)$

2 Loss function $l(\hat{y}, y)$

3 Function

$$f_w \in \mathcal{F} \quad : \quad \frac{1}{n} \sum_{i=1}^n l(f_w(x_i), y_i) \quad \rightarrow \quad \min_w$$

¹ [Léon Bottou and Olivier Bousquet](#). "The tradeoffs of large scale learning". In: *Advances in neural information processing systems*. 2008, pp. 161–168.

² [Léon Bottou](#). "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

Loss

- Expected risk

$$E(f) = \int l(f(x), y) dP(z)$$

- Empirical risk

$$E_n(f) = \frac{1}{n} \sum_{i=1}^n l(f(x_i), y_i)$$

Gradient descent

- GD

$$w_{t+1} = w_t - \gamma \frac{1}{n} \sum_{i=1}^n \nabla_w Q(z_i, w_t) \quad (1)$$

- SGD

$$w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t) \quad (2)$$

It is hoped that (2) behaves like its expectation (1)

Online learning

- ✓ SGD can process examples on the fly in a deployed system
- ✓ SGD directly optimizes the expected risk, since the examples are randomly drawn from $P(z)$

Approximations

$$f_w \in \mathcal{F} \quad : \quad \frac{1}{n} \sum_{i=1}^n l(f_w(x_i), y_i) \quad \rightarrow \quad \min_w$$

- $f^* = \arg \min_f E(f)$ – best possible prediction function
- $f_{\mathcal{F}}^* = \arg \min_{f \in \mathcal{F}} E(f)$ – best function in the family \mathcal{F}
- $f_n = \arg \min_{f \in \mathcal{F}} E_n(f)$ – empirical optimum
- \tilde{f}_n minimizes the objective function with a predefined accuracy $E_n(\tilde{f}_n) < E_n(f_n) + \rho$

The tradeoffs of large scale learning

Excess error $\mathcal{E} = \mathbb{E} [E(\tilde{f}_n) - E(f^*)]$ can be decomposed in three terms

$$\mathcal{E} = \mathbb{E} [E(f_{\mathcal{F}}^*) - E(f^*)] + \mathbb{E} [E(f_n) - E(f_{\mathcal{F}}^*)] + \mathbb{E} [E(\tilde{f}_n) - E(f_n)]$$

- $\mathcal{E}_{app} = \mathbb{E} [E(f_{\mathcal{F}}^*) - E(f^*)]$ — approximation error
- $\mathcal{E}_{est} = \mathbb{E} [E(f_n) - E(f_{\mathcal{F}}^*)]$ — estimation error
- $\mathcal{E}_{opt} = \mathbb{E} [E(\tilde{f}_n) - E(f_n)]$ — optimization error

Optimization

Two cases

$$\min_{\mathcal{F}, \rho, n} \mathcal{E} = \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}} \quad \text{subject to} \quad \begin{cases} n \leq n_{\text{max}} \\ T(\mathcal{F}, \rho, n) \leq T_{\text{max}} \end{cases}$$

- ✓ *Small-scale* constrained by the maximal number of examples. Choose ρ arbitrary small and set $n = n_{\text{max}}$.
- ✓ *Large-scale* constrained by the maximal computing time. Process more examples during allowed time.

Typical variations when \mathcal{F} , n , and ρ increase

		\mathcal{F}	n	ρ
\mathcal{E}_{app}	(approximation error)	\searrow		
\mathcal{E}_{est}	(estimation error)	\nearrow	\searrow	
\mathcal{E}_{opt}	(optimization error)	\cdots	\cdots	\nearrow
T	(computation time)	\nearrow	\nearrow	\searrow

Resume

The data sizes have grown faster than the speed of processors. In this context, the capabilities of statistical machine learning methods is limited by the computing time rather than the sample size.

Table of Contents

1. Large Scale ML

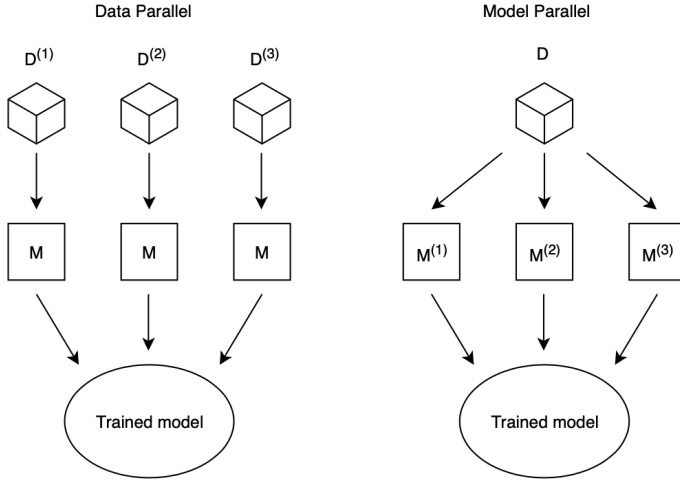
2. Distributed ML

Parallelism in Distributed Machine Learning
Ecosystem

3. General Purpose Distributed Computing

4. Natively Distributed ML Systems

Data & Model Parallelism³



³Joost Verbraeken et al. "A Survey on Distributed Machine Learning". In: *arXiv preprint arXiv:1912.09789* (2019).

Ecosystem

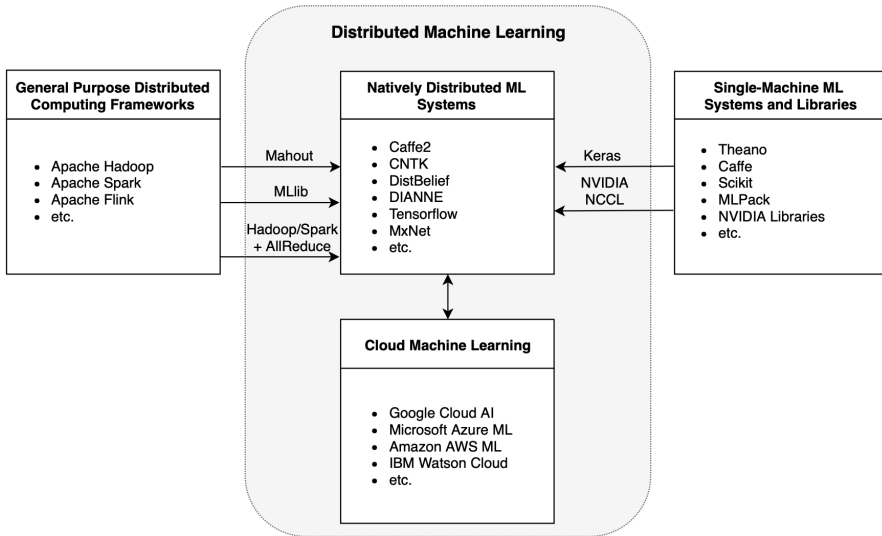


Table of Contents

1. Large Scale ML

2. Distributed ML

3. General Purpose Distributed Computing

MapReduce

MapReduce Parallel SGD

SparkNet

Spark MLlib

4. Natively Distributed ML Systems

MapReduce for ML⁴

Statistical Query Model

- Exact implementation of ML algorithms, not parallel approximations to algorithms

⁴Cheng-Tao Chu et al. "Map-reduce for machine learning on multicore". In: *Advances in neural information processing systems*. 2007, pp. 281–288.

Adopted Algorithms

LR

Logistic Regression

NNs

Neural Networks

k-Means

Clustering

PR

PageRank

MapReduce gives us Data Parallelism

Limitations

- Exact implementation of ML algorithms could be slow
 - it requires many passes through the dataset for convergence
 - it requires many synchronization sweeps (i.e. MapReduce iterations)
- Broadcasting parameters of LR, NN could be slow
- Using HDFS at each iteration is slow

How to speed up SGD?

Parallel SGD⁵

Algorithm 3.1 SGD($X^n = \{z_i\}_{i=1}^n, T, \gamma, w_0$)

for $t = 1, \dots, T$ **do**

 Draw $t \in \{1, \dots, n\}$ uniformly at random.

$w_t \leftarrow w_{t-1} - \gamma \nabla_w Q(z_t, w_{t-1})$

return w_t

⁵Martin Zinkevich et al. "Parallelized stochastic gradient descent". In: *Advances in neural information processing systems*. 2010, pp. 2595–2603.

Parallel SGD⁶

Algorithm 3.2 ParallelSGD($X^n = \{z_i\}_{i=1}^n, T, \gamma, w_0, K$)

for $i = 1, \dots, K$ **parallel do**

$v_i \leftarrow \text{SGD}(X^n, T, \gamma, w_0)$

Aggregate from all computers $v \leftarrow \frac{1}{K} \sum_{i=1}^K v_i$

return v

If T is much less than n , then it is only necessary for a machine to have access to the data it actually touches.

⁶Zinkevich et al., “Parallelized stochastic gradient descent”.

SimuParallelSGD⁷

Algorithm 3.3 SimuParallelSGD($X^n = \{z_i\}_{i=1}^n, \gamma, K$)

$T \leftarrow \lfloor n/K \rfloor$

Randomly partition the X^n , giving T examples to each machine

for $i = 1, \dots, K$ **parallel do**

 Randomly shuffle the data on machine i

 Initialize $w_{i,0} = 0$

for $t = 1, \dots, T$ **do**

 Get the t -th example on the i -th machine (this machine), $z^{i,t}$

$w_{i,t} \leftarrow w_{i,t-1} - \gamma \nabla_w Q(z^{i,t}, w_{i,t-1})$

Aggregate from all computers $v \leftarrow \frac{1}{K} \sum_{i=1}^K w_{i,t}$

return v

⁷Zinkevich et al., "Parallelized stochastic gradient descent".

SimuParallelSGD

Experiments

- **Dataset:** 2.5M Training examples, 600K Test examples. $x_i \in \mathbb{R}^{2^{18}}$
- **Approach:**
 1. Trained up to 100 models, each on an independent, random permutation of the full training data.
 2. During training, the model is stored on disk after $k = 10^4 \cdot 2^i$ updates.
 3. Then models obtained for each i are averaged and evaluated.
 4. This approach evaluates performance for the algorithm after each machine has seen k samples.
- **Metrics:** normalized RMSE (1.0 is the RMSE obtained by training in one sequential pass)

SimuParallelSGD

Experiment Results

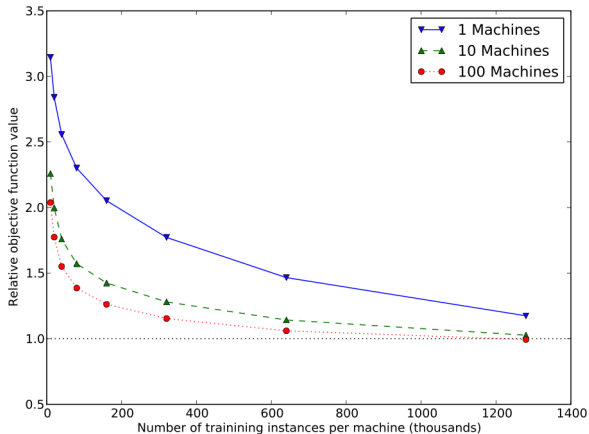


Figure: Relative training error

SimuParallelSGD

Conclusion

- ✓ *Data-parallel* SGD
- ✓ Highly suitable for parallel, large-scale ML (communication at the very end)

SparkNet⁸

Motivation & Benefits

Motivation:

- Main goal is to address limitations of training Neural Networks using MapReduce

Benefits:

- Model training is integrated in existing data-processing pipelines

⁸Philipp Moritz et al. "Sparknet: Training deep networks in spark". In: *arXiv preprint arXiv:1511.06051* (2015).

Implementation

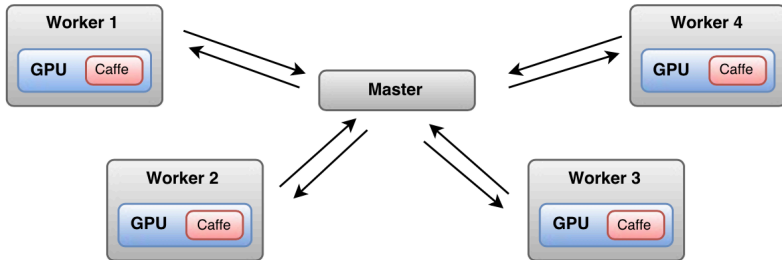


Figure: SparkNet architecture

Parallelizing SGD

Serial SGD

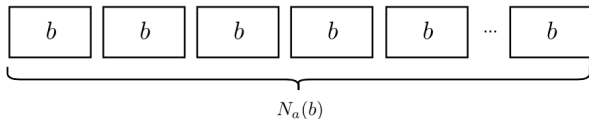


Figure: Serial run of SGD. Each block corresponds to a single SGD update with batch size b

The quantity $N_a(b)$ is the number of iterations required to achieve an accuracy of a

Parallelizing SGD

Naive parallelization

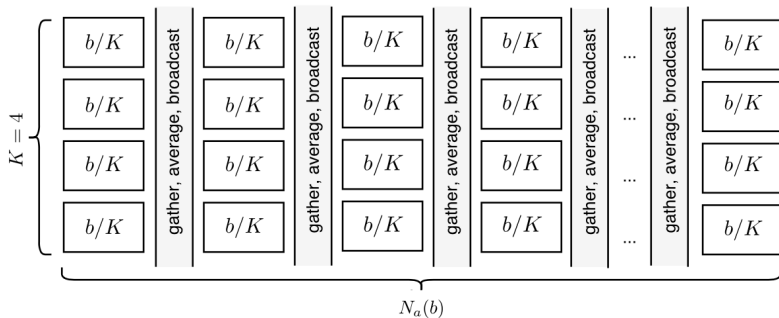


Figure: At each iteration, each batch of size b is divided among the K machines. this is equivalent to the serial run of SGD in Figure 3 and so the number of iterations is the same — $N_a(b)$.

Parallelizing SGD

SparkNet's parallelization

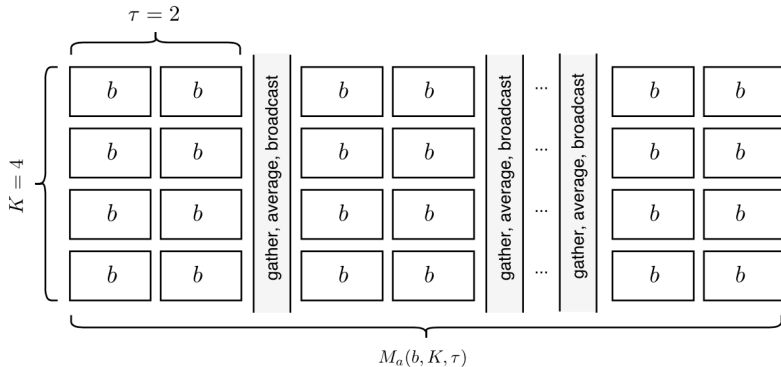


Figure: The quantity $M_a(b, K, \tau)$ is the number of rounds (of τ iterations) required to obtain an accuracy of a . The total number of parallel iterations of SGD under SparkNet's parallelization scheme required to obtain an accuracy of a is then $\tau M_a(b, K, \tau)$.

Evaluation

ImageNet speedup

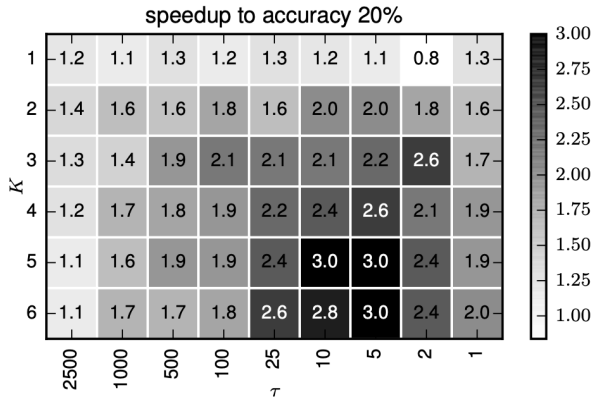


Figure: Speedup $\tau M_a(b, K, \tau) / N_a(b)$ given by SparkNet's parallelization scheme relative to training on a single machine to obtain an accuracy of $a = 20\%$

Cls&Reg

...

Cluster

k-Means, LDA, GMM

ML algorithms

RecSys

ALS

DimRed

SVD, PCA

⁹Xiangrui Meng et al. "Mllib: Machine learning in apache spark". In: *The Journal of Machine Learning Research* 17:1 (2016), pp. 1235–1241.

Table of Contents

1. Large Scale ML

2. Distributed ML

3. General Purpose Distributed Computing

4. Natively Distributed ML Systems

- Parameter Server

- DistBelief

- TensorFlow

- Horovod

Parameter Server¹⁰

Classic Parameter Server

- All parameters are stored in the PS
- Workers send gradients to PS and get updated parameters once in a while

Key observation

Mutable state is crucial when training very large models, because it becomes possible to make in-place updates to very large parameters, and propagate those updates to parallel training steps as quickly as possible.

¹⁰ Mu Li et al. "Scaling distributed machine learning with the parameter server". In: *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*. 2014, pp. 583–598.

DistBelief¹¹

DistBelief provides both Data and Model parallelism.

¹¹Jeffrey Dean et al. "Large scale distributed deep networks". In: *Advances in neural information processing systems*. 2012, pp. 1223–1231.

Model parallelism

- User defines the computation that takes place at each node in each layer of the model, and the messages that should be passed during the upward and downward phases of computation
- User may partition the model across several machines

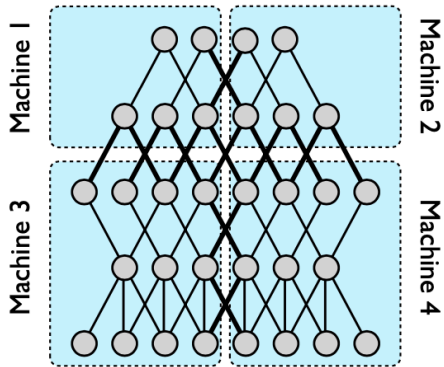


Figure: A 5 layer DNN with local connectivity, partitioned across 4 machines. Nodes with edges that cross partition boundaries will need to have their state transmitted between machines.

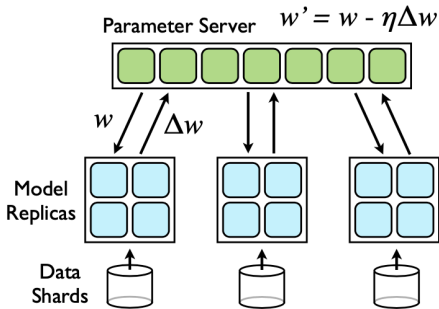
Model parallelism

Models with a large number of parameters or high computational demands typically benefit from access to more CPUs and memory, up to the point where communication costs dominate.

Distributed optimization algorithm

Downpour SGD

- We divide the training data into a number of subsets and run a copy of the model on each of these subsets
- Before processing each mini-batch, a model replica asks the parameter server for an updated copy of parameters
- We can limit replica to push updates only each n_{push} steps and ask for updates only each n_{fetch} steps



Experiments

Model parallelism benchmarks

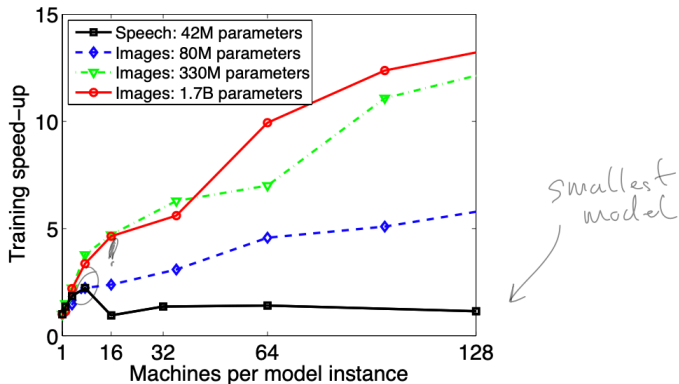


Figure: Training speed-up for four different deep networks as a function of machines allocated to a single DistBelief model instance

Experiments

Optimization method comparisons

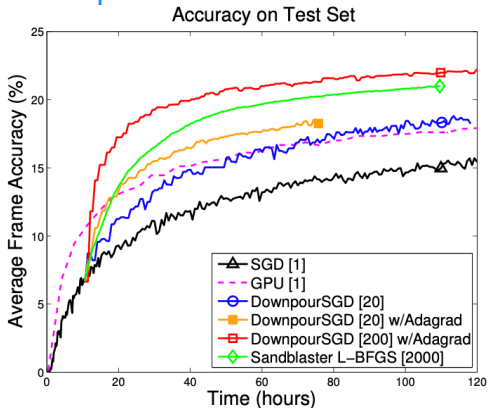


Figure: ImageNet. Classification accuracy on the hold out test set as a function of training time. Downpour and Sandblaster experiments initialized using the same 10 hour warmstart of simple SGD.

Experiments

Optimization method comparisons

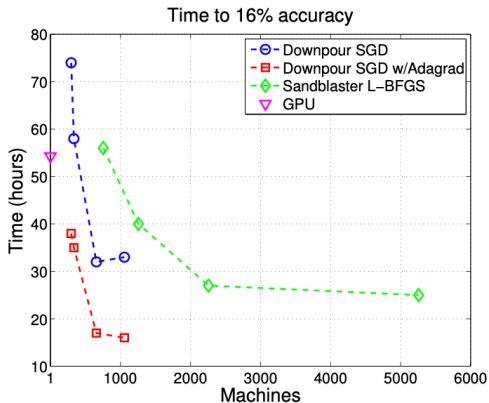


Figure: ImageNet. Time to reach a fixed accuracy (16%)

Conclusions

- Downpour SGD, a highly asynchronous variant of SGD works surprisingly well for training nonconvex deep learning models
- Methods can use a cluster of machines to train even modestly sized deep networks significantly faster than a GPU, and without the GPU's limitation on the maximum size of the model

TensorFlow¹² aka DistBelief 2.0

Basic concepts

- DataFlow graph (nodes, tensors, variables, operations)
- Kernel, Device
- Session
- Client, master, workers

¹² [Martín Abadi et al.](#) "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467* (2016); [Martín Abadi et al.](#) "Tensorflow: A system for large-scale machine learning". In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 265–283.

Multi-Device Execution

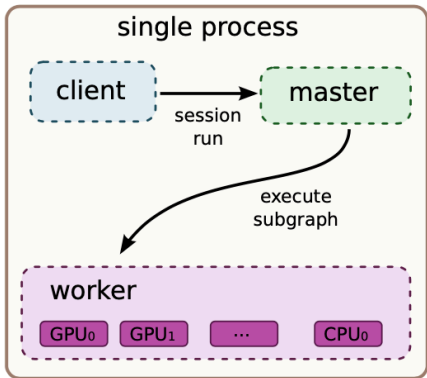


Figure: Single machine structure

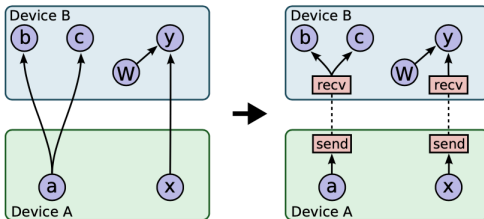


Figure: Cross-device communication

Distributed Execution

- Distributed execution of a graph is very similar to multi- device execution
- TensorFlow supports both (A)synchronous Data Parallelism and Model Parallelism

Limitations of (TensorFlow) Parameter Server

- If one parameter server is used, it will likely become a networking or computational bottleneck.
- If multiple parameter servers are used, the communication pattern becomes “all-to-all” which may saturate network interconnects.

Ring Allreduce¹⁷ & Horovod¹⁸

- Originally proposed and implemented at Baidu¹³
- Open-sourced at Uber¹⁴¹⁵¹⁶

¹³<http://andrew.gibiansky.com/>

¹⁴<https://github.com/horovod/horovod/blob/master/docs/spark.rst>

¹⁵<https://docs.databricks.com/applications/deep-learning/distributed-training/index.html>

¹⁶https://youtu.be/0l3i_QKj8sY

¹⁷Baidu Research. *Baidu: Ring Allreduce*. <http://andrew.gibiansky.com/>. 2017.









¹⁸Alexander Sergeev and Mike Del Balso. "Horovod: fast and easy distributed deep learning in TensorFlow". In: *arXiv preprint arXiv:1802.05799* (2018).








Remarks

- Literature with more math: Book,¹⁹ Papers²⁰
- What about non-SGD ML algorithms?

¹⁹ [Levent Bottou](#), [Frank E Curtis](#), and [Jorge Nocedal](#). "Optimization methods for large-scale machine learning". In: *Siam Review* 60.2 (2018), pp. 223–311; [Elad Hazan et al.](#) "Introduction to online convex optimization". In: *Foundations and Trends® in Optimization* 2.3-4 (2016), pp. 157–325.

²⁰ [Zinkevich et al.](#), "Parallelized stochastic gradient descent"; [Bottou and Bousquet](#), "The tradeoffs of large scale learning"; [Bottou](#), "Large-scale machine learning with stochastic gradient descent".

-  Abadi, Marten et al. "Tensorflow: A system for large-scale machine learning". In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 265–283.
-  Abadi, Marten et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467* (2016).
-  Bottou, Léon. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.
-  Bottou, Léon and Olivier Bousquet. "The tradeoffs of large scale learning". In: *Advances in neural information processing systems*. 2008, pp. 161–168.
-  Bottou, Léon, Frank E Curtis, and Jorge Nocedal. "Optimization methods for large-scale machine learning". In: *Siam Review* 60.2 (2018), pp. 223–311.
-  Chu, Cheng-Tao et al. "Map-reduce for machine learning on multicore". In: *Advances in neural information processing systems*. 2007, pp. 281–288.
-  Dean, Jeffrey et al. "Large scale distributed deep networks". In: *Advances in neural information processing systems*. 2012, pp. 1223–1231.
-  Hazan, Elad et al. "Introduction to online convex optimization". In: *Foundations and Trends® in Optimization* 2.3-4 (2016), pp. 157–325.

-  Li, Mu et al. "Scaling distributed machine learning with the parameter server". In: *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*. 2014, pp. 583–598.
-  Meng, Xiangrui et al. "Mllib: Machine learning in apache spark". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1235–1241.
-  Moritz, Philipp et al. "Sparknet: Training deep networks in spark". In: *arXiv preprint arXiv:1511.06051* (2015).
-  Research, Baidu. *Baidu: Ring Allreduce*. <http://andrew.gibiansky.com/>. 2017.
-  Sergeev, Alexander and Mike Del Balso. "Horovod: fast and easy distributed deep learning in TensorFlow". In: *arXiv preprint arXiv:1802.05799* (2018).
-  Verbraeken, Joost et al. "A Survey on Distributed Machine Learning". In: *arXiv preprint arXiv:1912.09789* (2019).
-  Zinkevich, Martin et al. "Parallelized stochastic gradient descent". In: *Advances in neural information processing systems*. 2010, pp. 2595–2603.



Вопросы?

Ильнур Шугаев