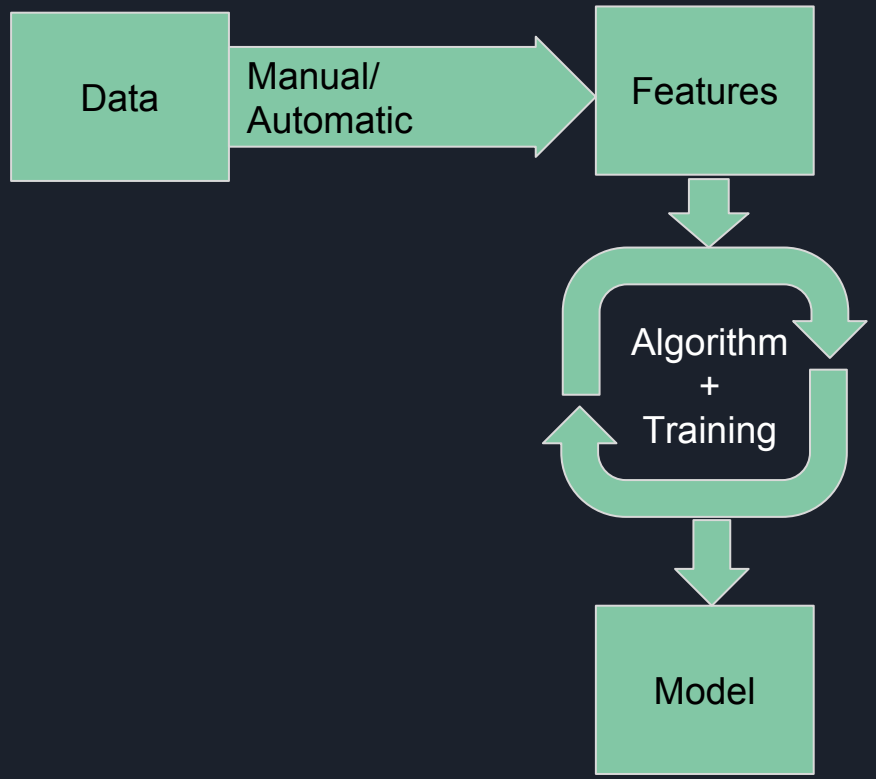
Distributed Machine Learning

Group Members: Ajesh, Evangelos, Rav

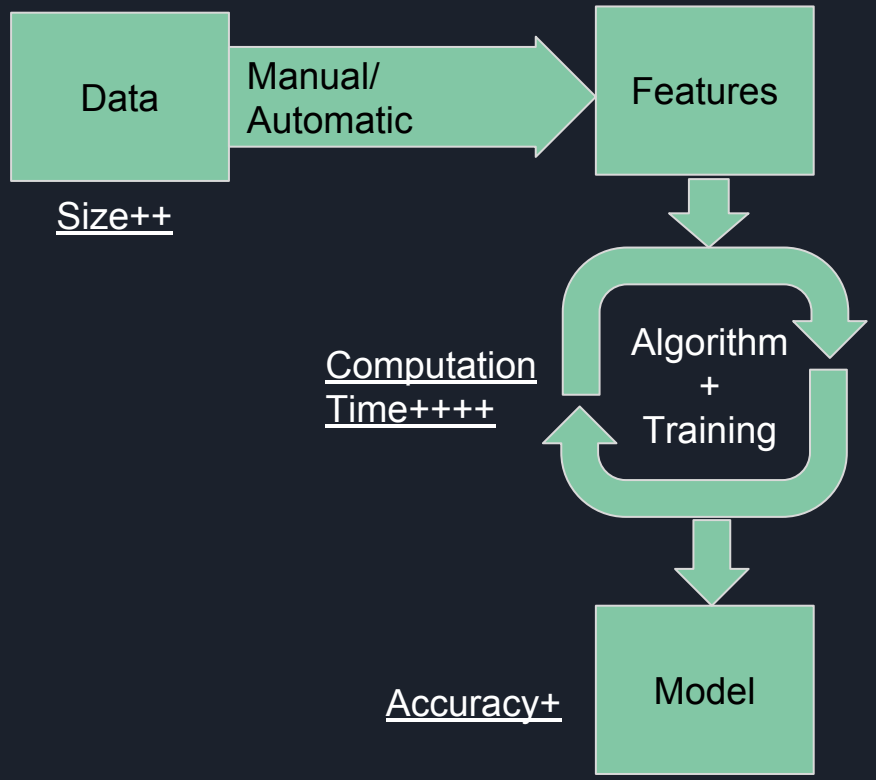
What is Machine Learning?

- Model
  - Computation architecture
  - Parameters
- Data
  - Training set
  - Features



What is Machine Learning?

- Model
  - Computation architecture
  - Parameters
- Data
  - Training set
  - Features



Motivations for DML

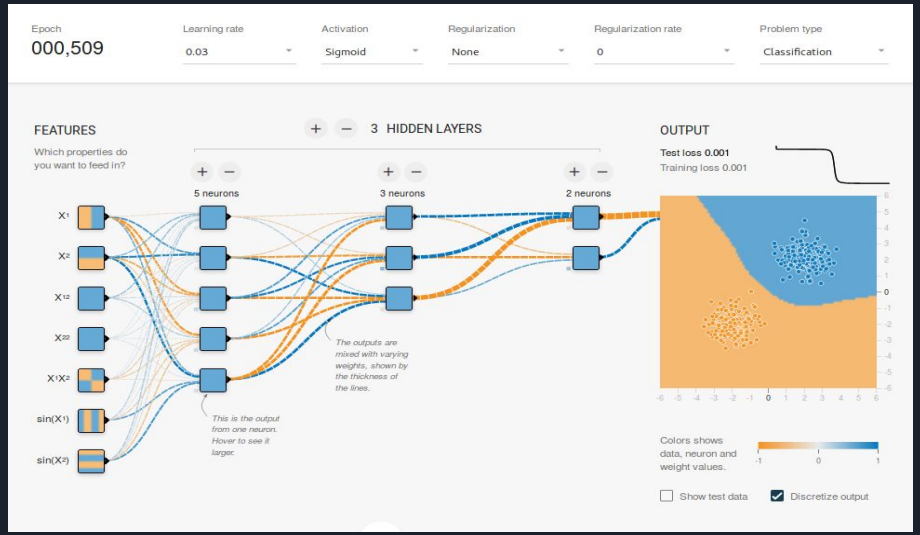
- Data size (TB - PB)
- Large Volume of Data increases Model accuracy
- A lot of Computations
- Data Locality
- Data privacy / security

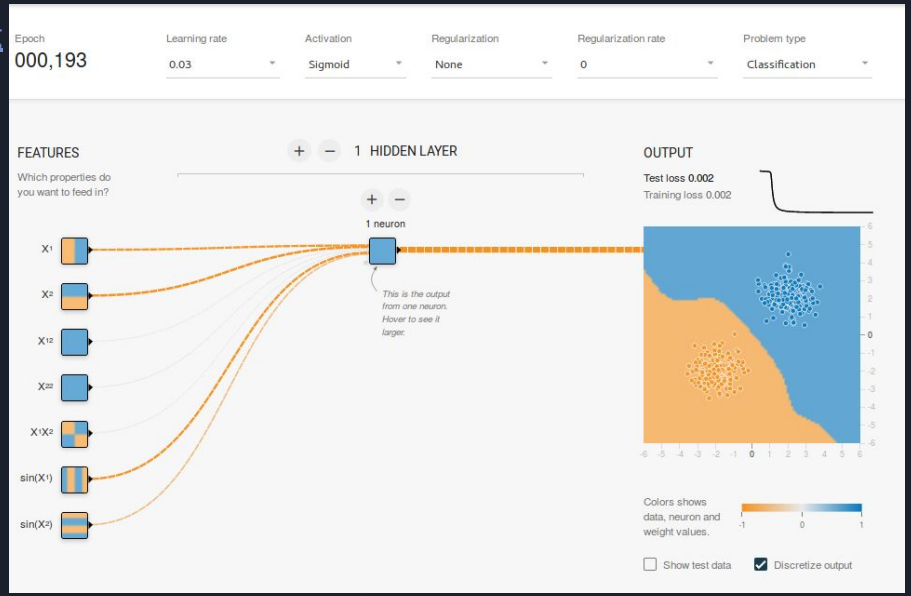
Challenges of DML

- Communication cost
  - Latency
  - Bandwidth
- Diverse Resource Capabilities
  - Differing CPU Threads
  - RAM available
  - Power Consumption
- Poor Resource Utilization
- Distribute data for workers
- Difficult to Develop
- Error Tolerance
- Current ML Algorithms are not naturally distributed
- Balancing consistency with performance and accuracy

Motivating Example

Neural Net http://playground.tensorflow.org





Strategies for DML

- Data Parallelism
  - Parameter server scheme
- Model Parallelism
  - Distributed networks (NN)
  - Pipelined computation
- Hybrid Approach

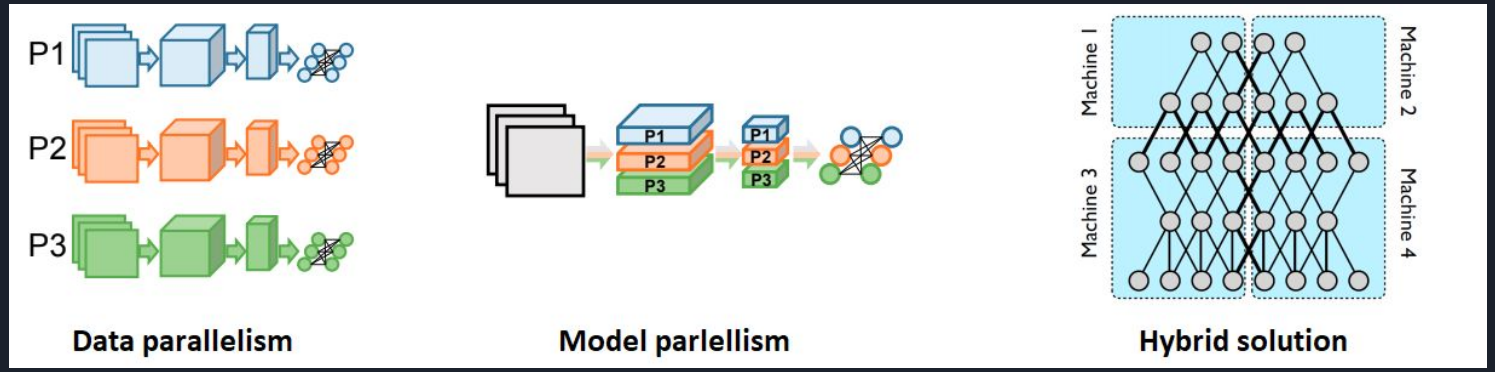
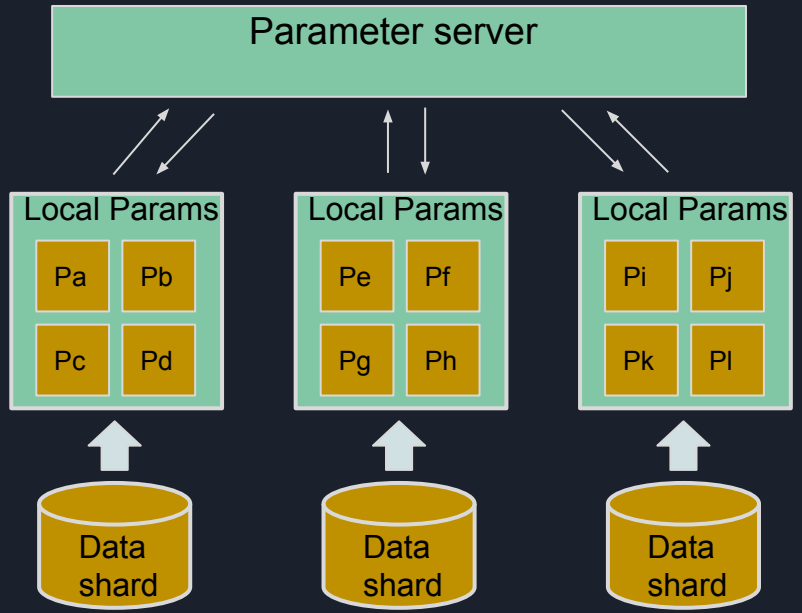


Image Courtesy of [3] TAL BEN-NUN and TORSTEN HOEFLER, ETH Zurich, Switzerland

Data-Parallel: Parameter-Server

- 1 point of reference for Global Model Parameters
- Each worker has a partition of the data and computes a partial update of all parameters
- Takes advantage of Error Tolerance in ML Algorithms



Parameter-Server: Synchronization

- Bulk Synchronous Parallel
  - Synchronize after each iteration
- Stale Synchronous Parallel
  - Synchronizes after the fastest node passes the slowest in s iterations
- Adaptive Synchronous Parallel
  - Zhang et al suggests changing the stale threshold based on CPU occupancy and have a minimum number of executions for the slowest [2]

Model Parallel: Distributed networks

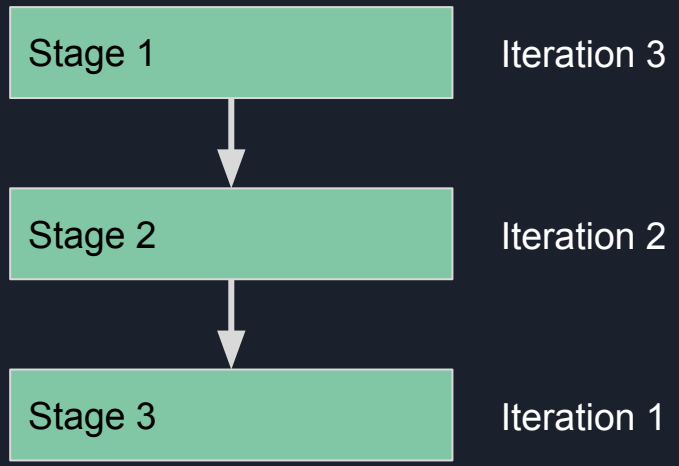
- Computation is divided among nodes
  - Different layers or parts of a layer in a NN
- Model is split among nodes
- Every node has access to the entire dataset
- Every node is responsible of its part of the parameter space
- Heavy communication is usually required
  - DDNN early exit [4]
- Issue: Under-utilized nodes

Model Parallel: Pipelined computation

- Overall computation in iterative steps
  - Layers of neural networks
- Subset of nodes responsible for part of computation
- Temporal division
  - One iteration goes through multiple stages in subsequent time intervals
- Easier communication
  - Messages flow from one stage to the next only
  - Simple synchronization
- Bigger throughput and latency
- Susceptible to bottlenecks

Pipelined computation

- 3 different iterations being executed at the same time
- One iteration goes through 3 seperate barrier-synchronized steps



Early Implementations

- Hand-crafted implementations
  - C++
  - MPI
- Distributed computing frameworks
  - MapReduce
- Adapting ML libraries to distributed computation
  - Caffe (DNN)
- Adapting distributed computation libraries to ML
- Simple strategies

DML-specific frameworks

- Specialized libraries
  - Spark/MLlib
  - TensorFlow
  - GraphLab
- Data flow paradigm
  - Execution DAG
  - Graph coloring
- Industrial solutions
  - Difficult to implement though
- Novel strategies
  - Hybrid parallelism
  - Parameter server variants
  - Relaxing consistency and accuracy for speed
  - Distributed gradient descent

Recent Developments

- Mature industrial solutions
  - Easier to set up production-level DML
- Novel systems
  - DDNN - edge computing
  - ASP
- Bridging the gap between ease of use and performance
  - MLI
  - MXNet

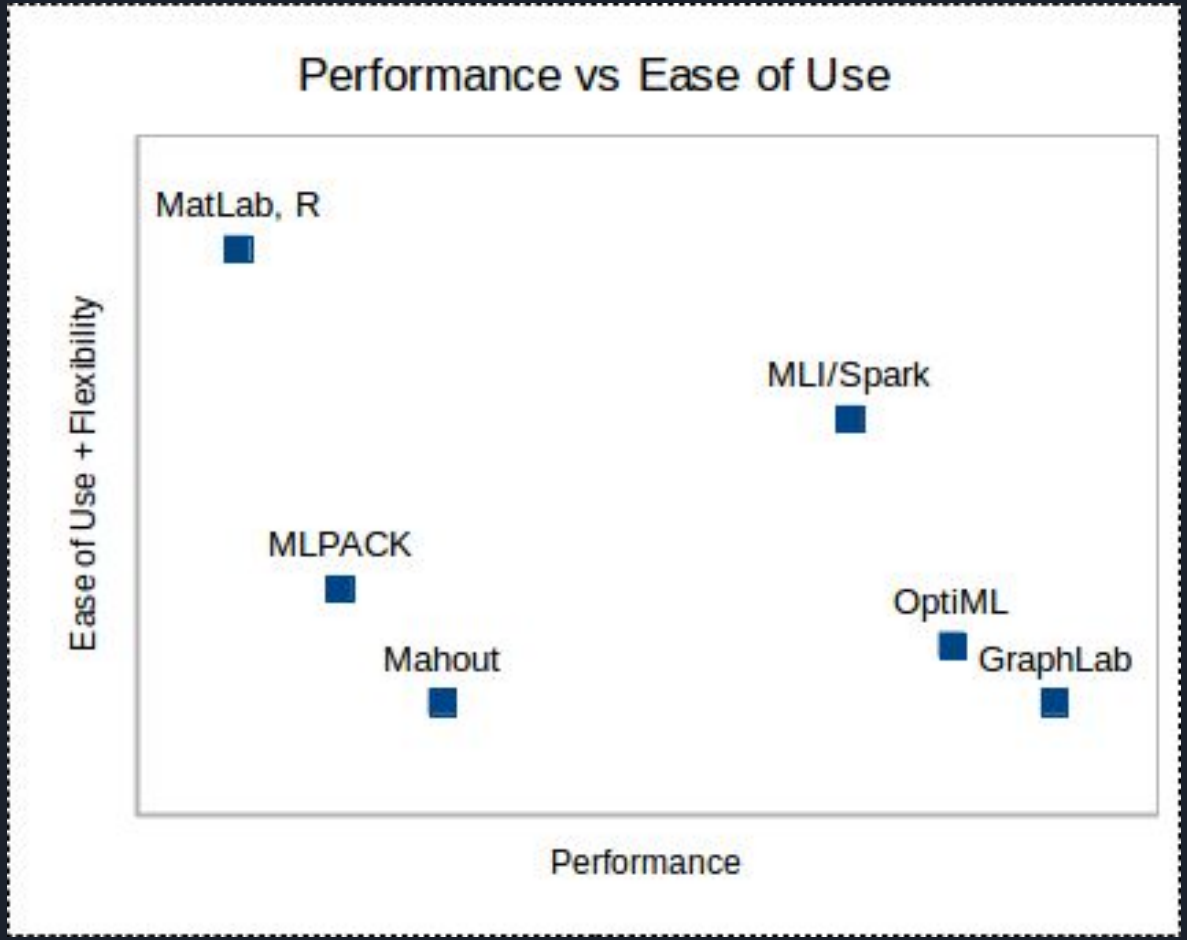


Image Recreated from: Sparks et al., MLI: An API for Distributed Machine Learning

Implementations

- Data-Parallel
  - Parameter Server
    - Petuum/PMLS - C++
    - YahooLDA
    - MXNet
- Model-Parallel
  - Strads
- Specialized libraries
  - GraphLab
  - Spark/MLLib
  - TensorFlow
  - PyTorch

Thank You
Any Questions?

References

[1] K. Zhang, S. Alqahtani, and M. Demirbas, “A Comparison of Distributed Machine Learning Platforms,” in 2017 26th International Conference on Computer Communication and Networks (ICCCN), 2017, pp. 1–9.

[2] J. Zhang et al., “An Adaptive Synchronous Parallel Strategy for Distributed Machine Learning,” IEEE Access, vol. 6, pp. 19222–19230, 2018.

[3] T. Ben-Nun and T. Hoefler, “Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis,” arXiv:1802.09941 [cs], Feb. 2018.

[4] S. Teerapittayanon, B. McDanel, and H. T. Kung, “Distributed Deep Neural Networks Over the Cloud, the Edge and End Devices,” in 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 2017, pp. 328–339.

[5] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, “Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud,” Proc. VLDB Endow., vol. 5, no. 8, pp. 716–727, Apr. 2012.

[6] T. Kraska, A. Talwalkar, and J. Duchi, “MLbase: A Distributed Machine-learning System,” p. 7.

[7] E. R. Sparks et al., “MLI: An API for Distributed Machine Learning,” arXiv:1310.5426 [cs, stat], Oct. 2013.

[8] X. Meng et al., “MLlib: Machine Learning in Apache Spark,” Journal of Machine Learning Research, vol. 17, no. 34, pp. 1–7, 2016.

[9] S. Lee, J. K. Kim, X. Zheng, Q. Ho, G. A. Gibson, and E. P. Xing, “On Model Parallelization and Scheduling Strategies for Distributed Machine Learning,” in Advances in Neural Information Processing Systems 27, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2834–2842.