GPU Acceleration for Machine Learning

John Canny*^

* Computer Science Division

University of California, Berkeley

^ Google Research, 2016

Outline

BIDMach on single machines

BIDMach on clusters

DNNs for Power-Law data

MCMC for massive datasets

Personal History

Yahoo [Chen, Pavlov, Canny, KDD 2009]*

Ebay [Chen, Canny, SIGIR 2011]**

Quantcast 2011-2013

Microsoft 2014

Yahoo 2015 [KDD 2015]

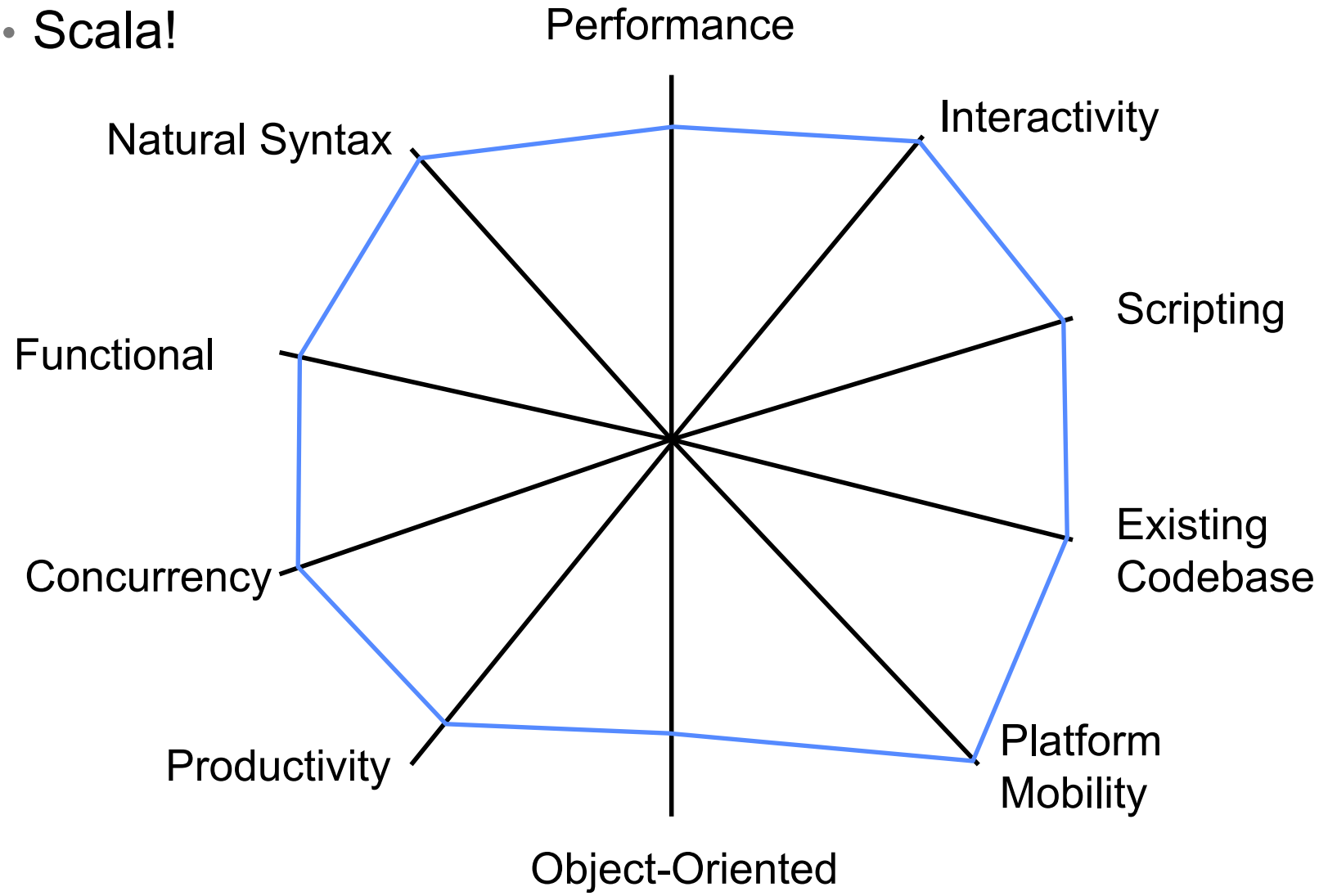
Google 2016-

* Best application paper prize

** Best paper honorable mention

Desiderata for an ML Toolkit

- Scala!



The Scala Language

- Scala is a JVM language, performance similar to Java.
- But has a REPL, similar to interpreted languages.
- Functional + Object Oriented + Actors.
- High productivity, >> C, >= Python.
- Open syntax (arbitrary operator definitions), great for DSLs.
- Base language for Apache Spark

Performance in Perspective

System	Matrix A + B
C	1600 Mflops
Julia	800 Mflops
Scala	500 Mflops
Lua	5 Mflops
Python	1 Mflops

Sandy Bridge, 2.2 GHz CPU, 1 thread

Matrix A * B (BLAS)	
CPU	GPU
400 Gflops	5 Tflops

Dual Haswell (20-core) 2.4 GHz
Titan-X GPU

Philosophy of BIDMach

Roofline design: Explicit accounting of performance limits:

- ALU speed
- Memory latency and throughput
- Network latency and throughput
- Secondary storage latency and throughput

Explorations

Codesign: Hardware-aware kernels:

- Can fit tons of state in GPU registers:
  - Natural language parser (EMNLP 2013) 1 tflop
  - Fused-kernel Word2Vec (draft paper) 200 gflops
  - Auction simulator (IEEE Big Data 2015) 400x CPU
  - Backward-step + ADAGRAD
  - Random Forests
  - Fused LSTM kernels

GPU programming workflow

Coding GPU kernels in CUDA C is hard (C is hard, massively multi-threaded programming is hard).

Automatic code generation would be great, but we don't understand the design space ***yet***

Coding CUDA kernels can be *greatly* accelerated by:

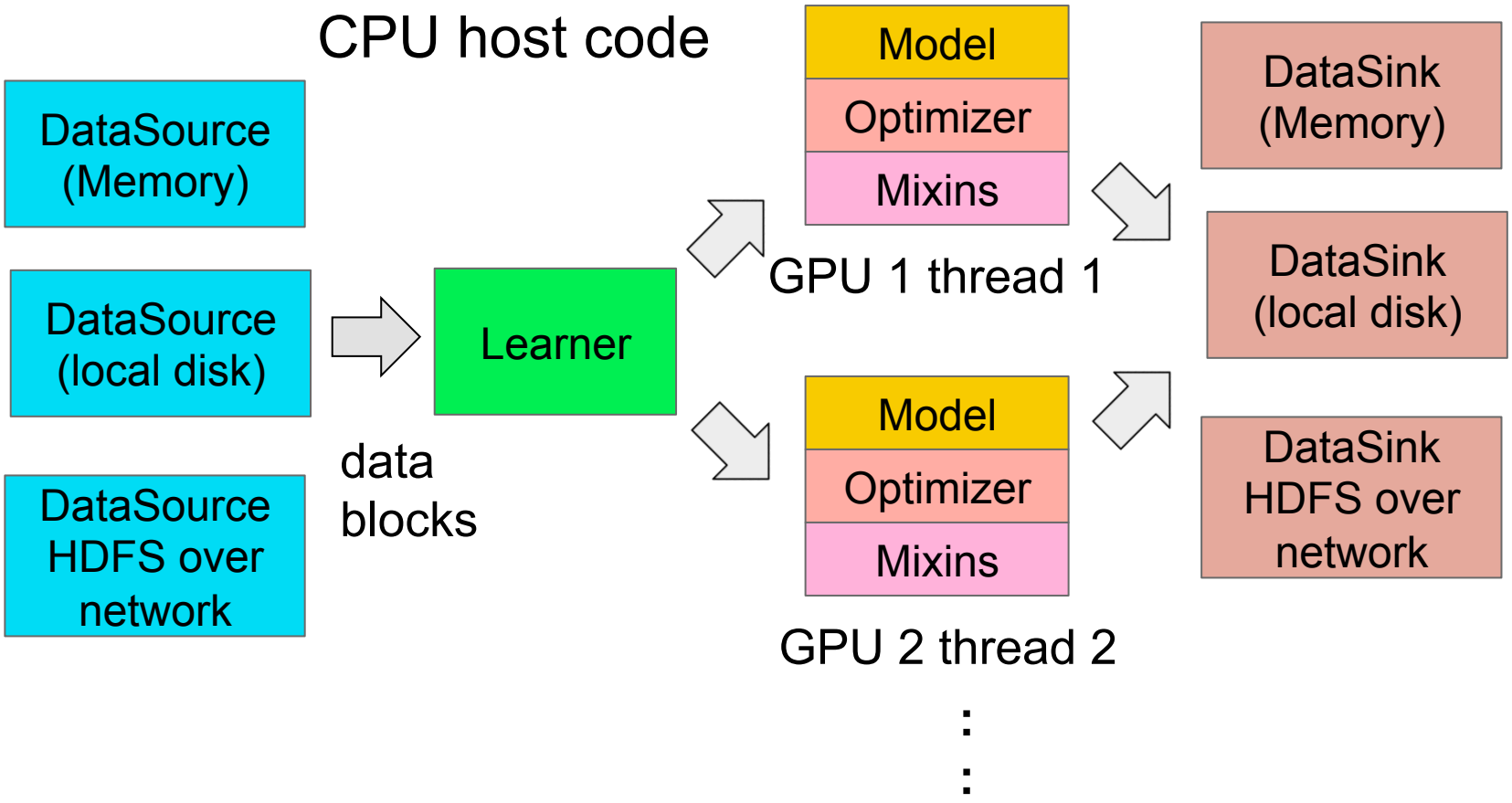
- Writing a "GPU-like" Scala-language program.
- Debug with: Eclipse, interpreter, scripts,…
- Port to CUDA. Scala model provides *ground truth*.

Stackless Viterbi parser, Auction simulator,...

A Rooflined Machine Learning Toolkit

BIDMACH

Zhao+Canny SIAM DM 13, KDD 13, KDD 2015, IEEE BigData 2015



BIDMach ML Algorithms

1. Regression (logistic, linear)
2. Support Vector Machines
3. k-Means Clustering
4. Topic Modeling - Latent Dirichlet Allocation
5. Collaborative Filtering
6. NMF – Non-Negative Matrix Factorization
7. Factorization Machines
8. Random Forests
9. IPTW (Causal Estimation)
10. ICA
11. Word2Vec
12. Discrete Graphical models
13. Scalable SVD
14. 1D Neural Networks, LSTMs etc.

= Likely the fastest implementation available

Benchmarks

Systems (single node)

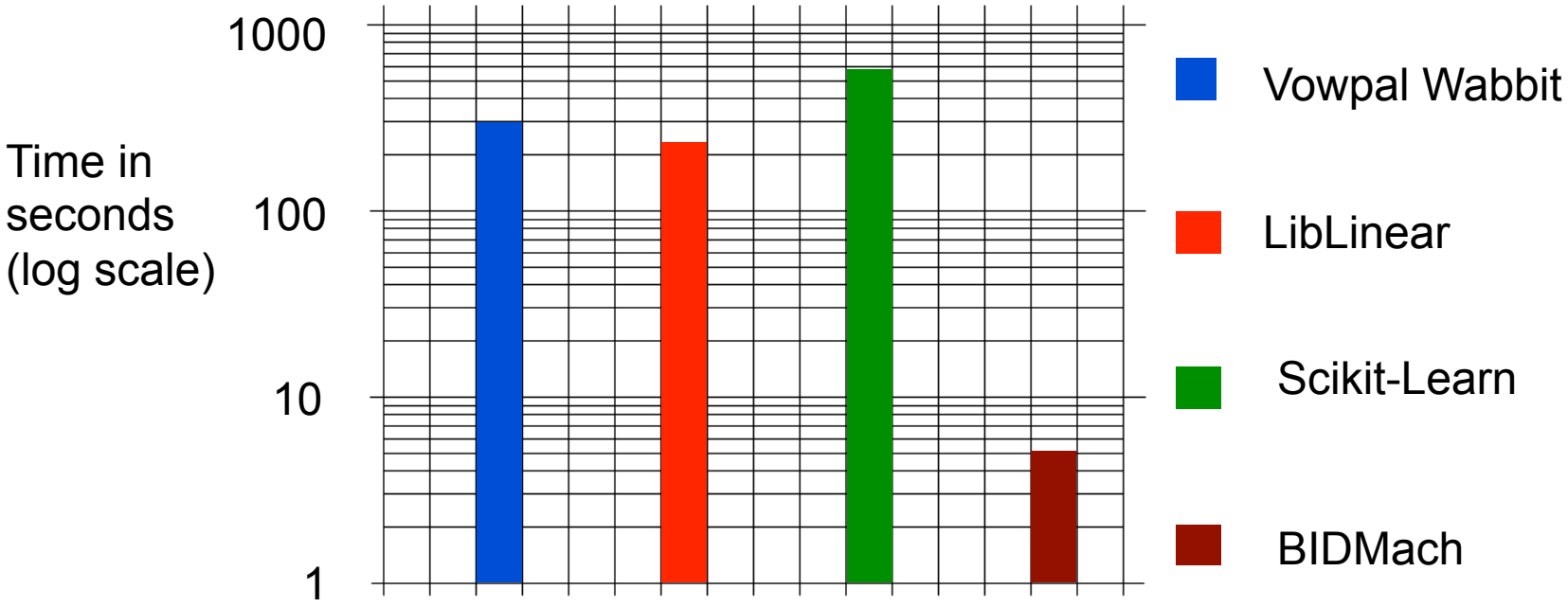
- BIDMach
- VW (Vowpal Wabbit) from Yahoo/Microsoft
- Scikit-Learn
- LibLinear

Cluster Systems

- Spark v1.2 and v1.5
- Graphlab (academic version)
- Petuum Parameter Server
- Yahoo's LDA cluster

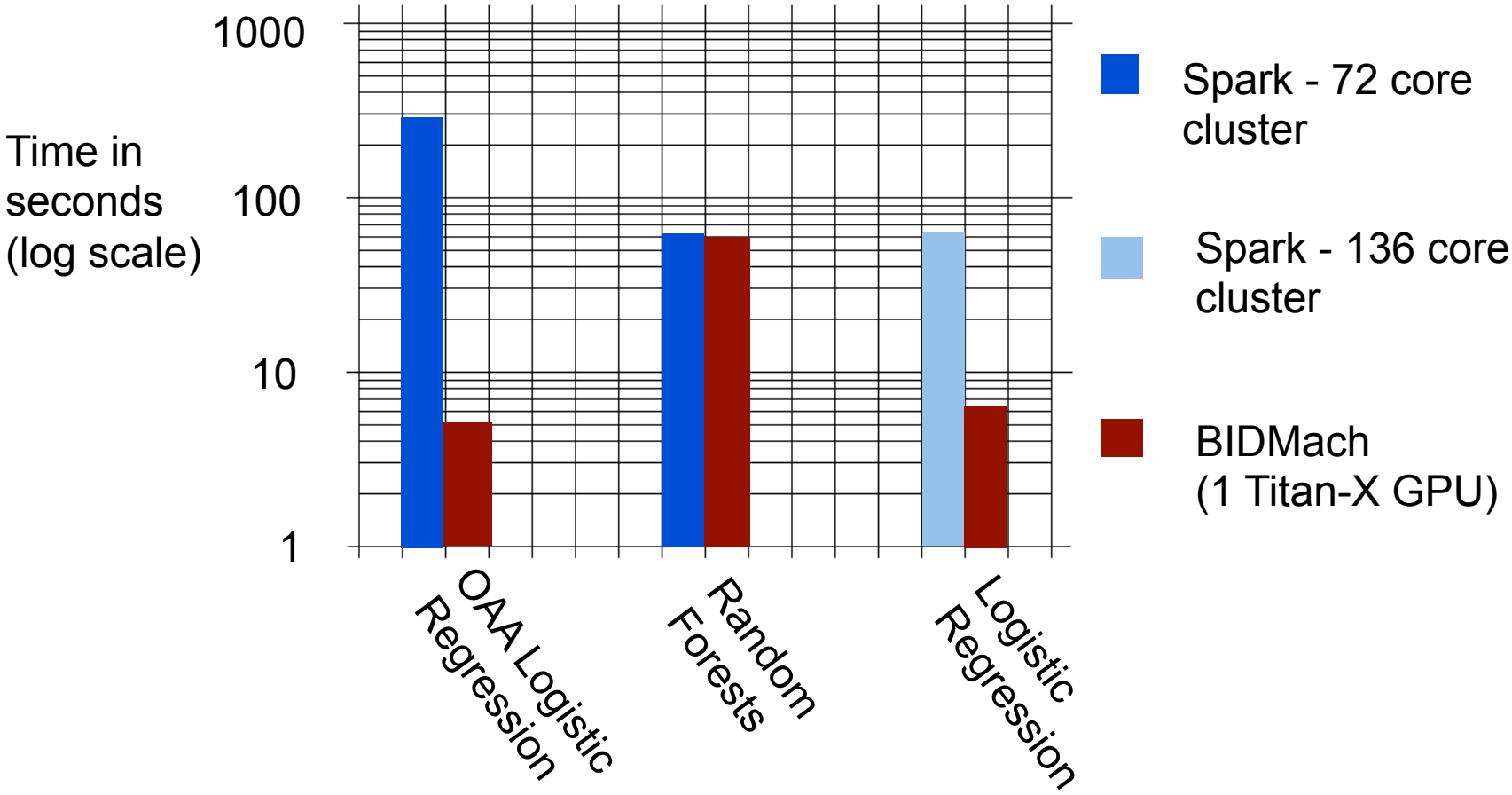
Benchmarks

Single-Machine Benchmarks: Multilabel classification
RCV1 dataset: Text Classification, 103 topics (0.5GB).



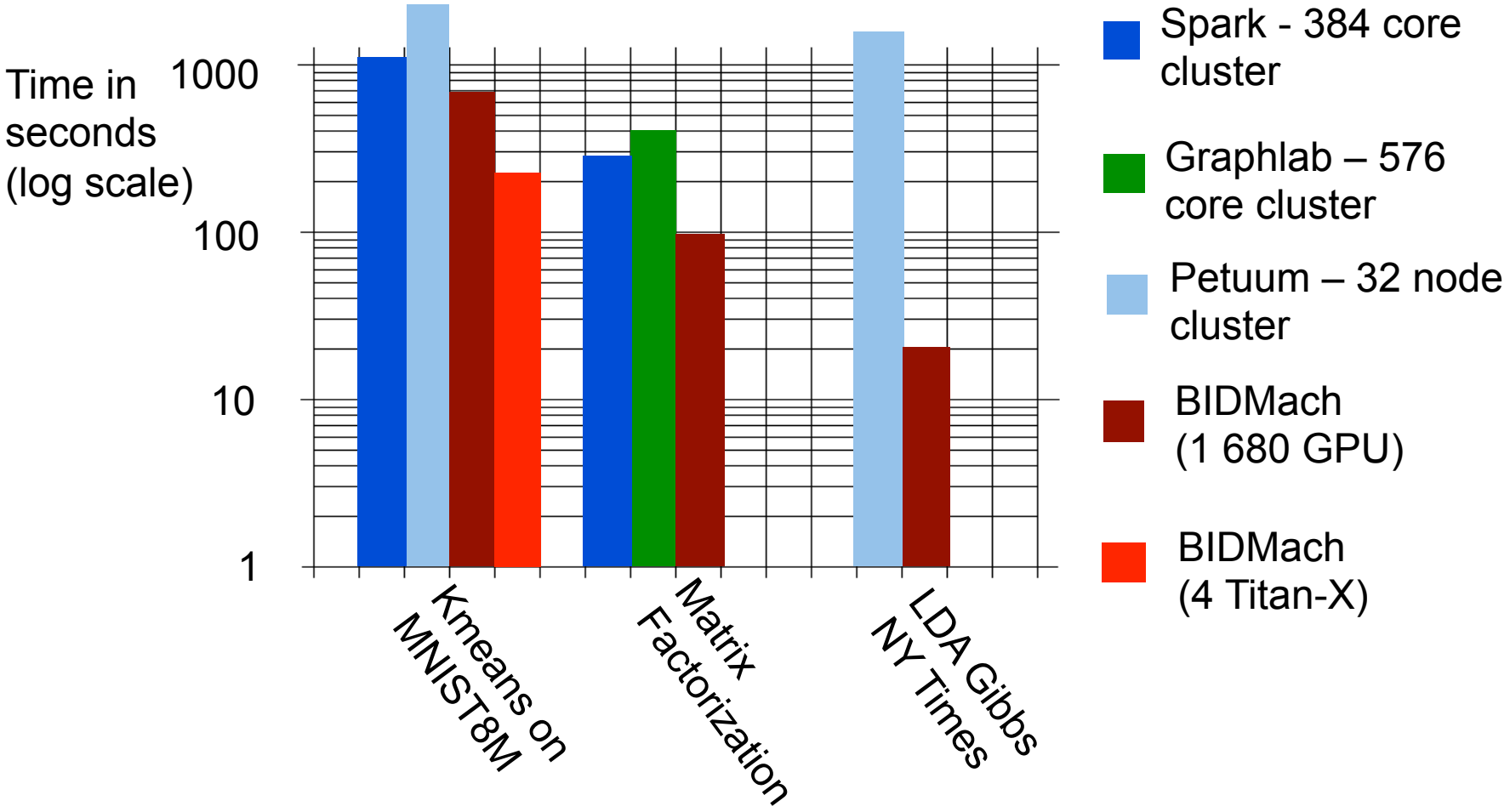
Benchmarks

Benchmarks vs Clusters: Tasks as indicated



Benchmarks

Unsupervised Learning: Tasks as indicated



Outline

BIDMach on single machines

BIDMach on clusters

DNNs for Power-Law data

MCMC for massive datasets

Allreduce vs. Parameter Servers

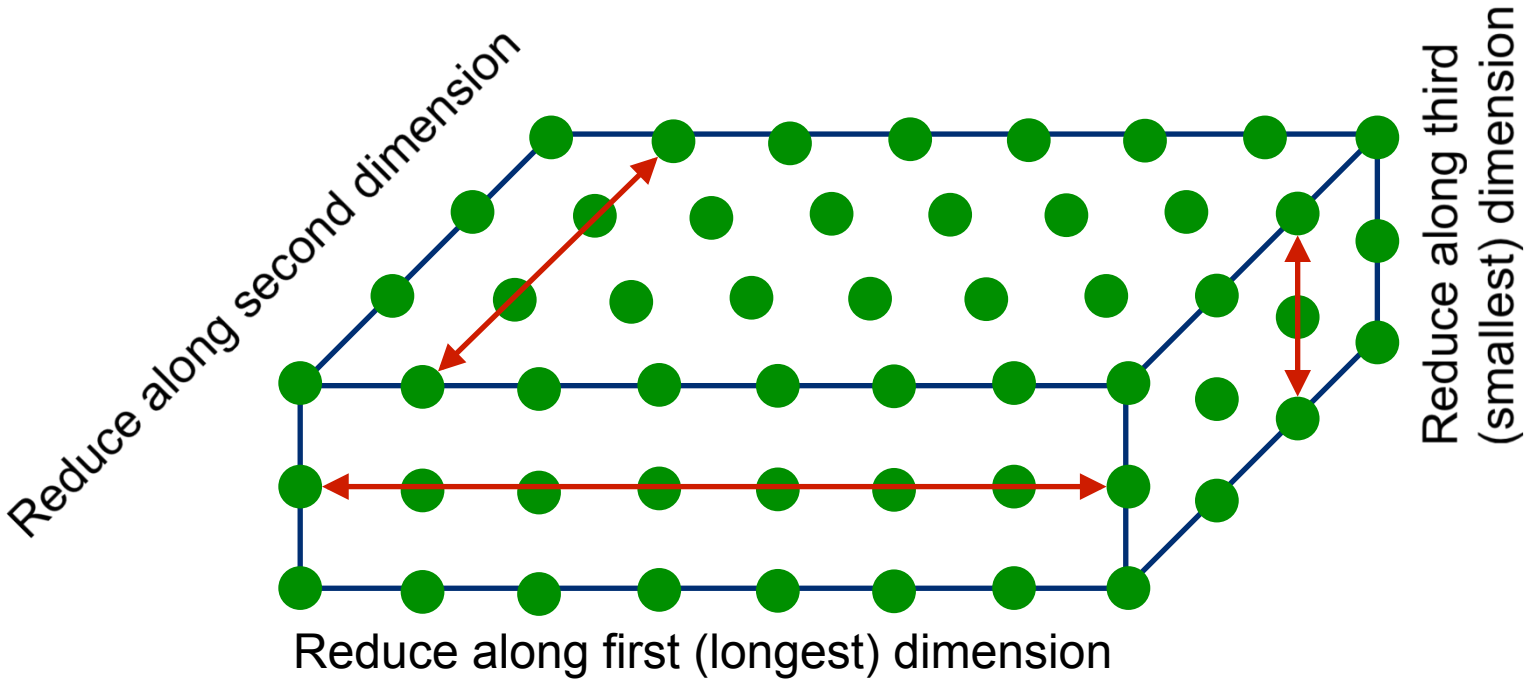
- Allreduce:
  - + B/W optimal, peer-to-peer (good scaling), no queueing, locks
  - - synchronous only, dense data only
- Parameter Servers:
  - + Sparse updates, asynchronous
  - - resource-hungry (large # servers), high staleness, complex
- Sparse Allreduce (Kylix)
  - Peer-to-peer, sparse updates, simple, B/W optimal, client asynchronous, fault tolerant.

Kylix: A Sparse AllReduce for Commodity Clusters

ICPP 2014

Hypercube allreduce mitigates latency

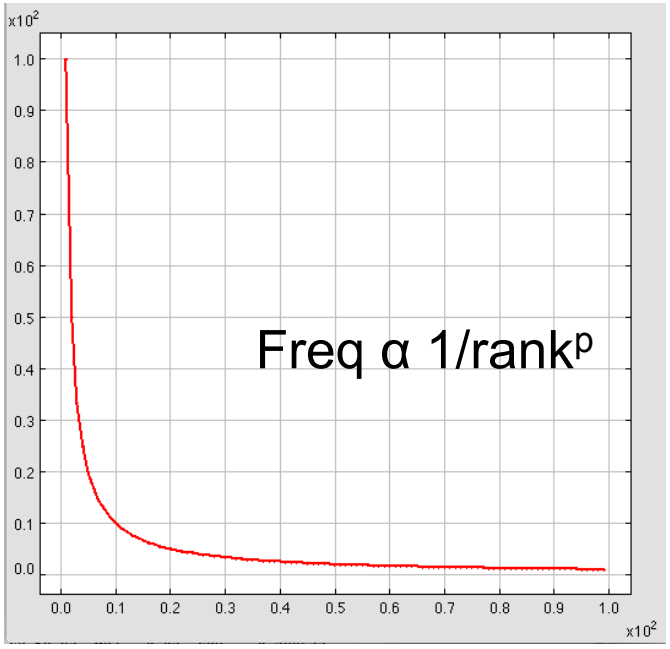
- Group size along each dimension controls message in order to achieve **optimal message size.**
- Data vectors overlap in each reduction leading to a **reduction in message volume** with each layer.



Power-Law Features

Big Data about people (text, web, social media) usually follow power law statistics:

Feature frequency

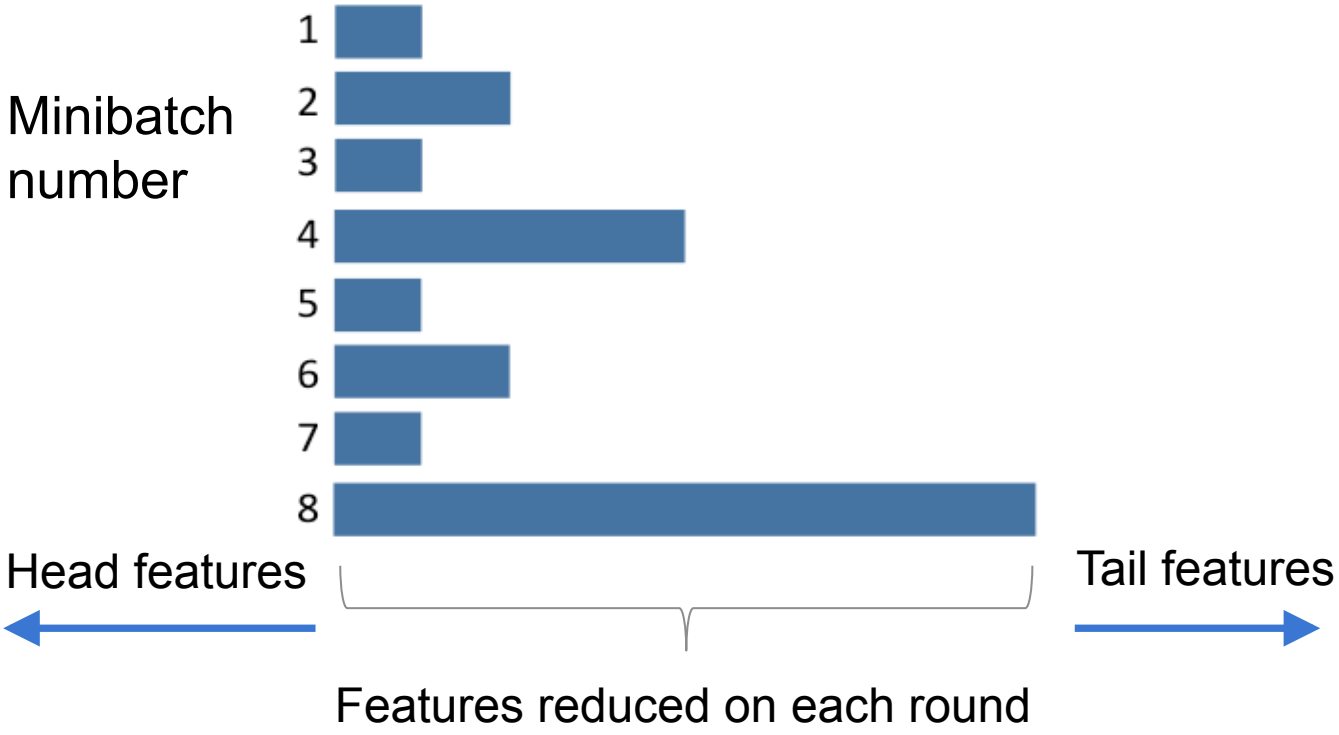


Feature rank

Feature sorted by frequency descending

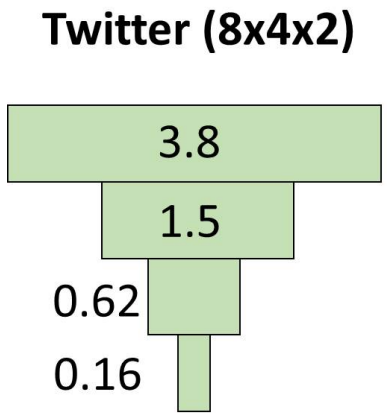
Minimizing Network Bandwidth

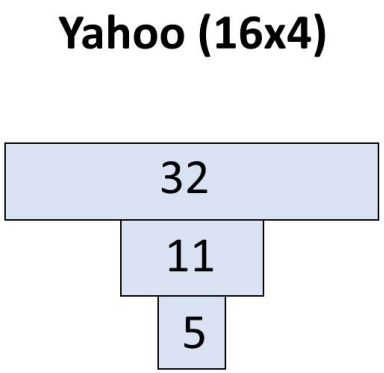
Graded updates refresh each feature at a rate inversely proportional to its rank. This is *proportional* (for power law date) to the rate at which the feature is updated by SGD.



Data Volumes with Sparse Data

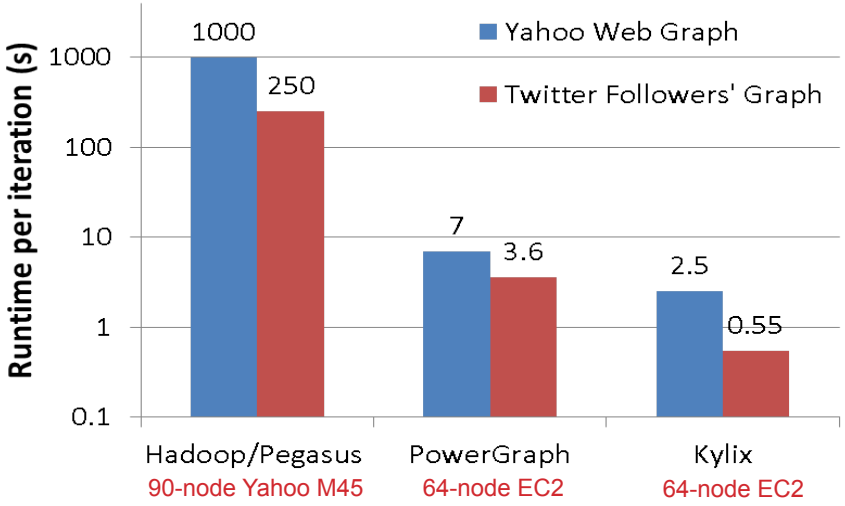
- Total communication across all layers a small constant larger than the top layer, which is close to optimal.
- Communication volume across layers has a characteristic **Kylix** shape.





Experiments (PageRank)

- Twitter Followers' Graph
  - 1.4 billion edges and 40 million vertices
- Yahoo Web Graph
  - 6.7 billion edges and 1.4 billion vertices
- EC2 cluster compute node (cc2.8xlarge)



BIDMach-on-Spark

- Spark is a powerful platform for data manipulation in Scala.
  - But only batch updates, immutable objects, unoptimized ML
- BIDMach-on-Spark adds
  - Minibatch updates – faster convergence
  - Peer-to-peer, hierarchical Allreduce (Kylix)
  - GPU support

BIDMach-on-Spark Benchmarks

Logistic Regression (Criteo 20 GB Dataset).

System	Algorithm	Passes	AUC	Time(s)
Spark 17x m3.2xlarge	LR-LBFGS	3	0.68	3300
BIDMach 1x g2.xlarge	LR-SGD	3	0.74	3000
BIDMach 17x g2.xlarge	LR-SGD	3	0.74	220

BIDMach-on-Spark cluster running periodic Kylix Allreduce.

BIDMach-on-Spark Benchmarks

KMeans on the MNIST 8M dataset (about 26 GB).

System (node type)	Nodes	Inertia	Time(s)
Spark (m3.2xlarge)	97	1.5E6	1130
Petuum (m3.2xlarge)	17	??	2500
BIDMach (g2.xlarge)	1	1.5E6	460
BIDMach (g2.xlarge)	17	1.5E6	60

BIDMach-on-Spark cluster running batch Allreduce.

All systems running 10 iterations with 1024 centers

Outline

BIDMach on single machines

BIDMach on clusters

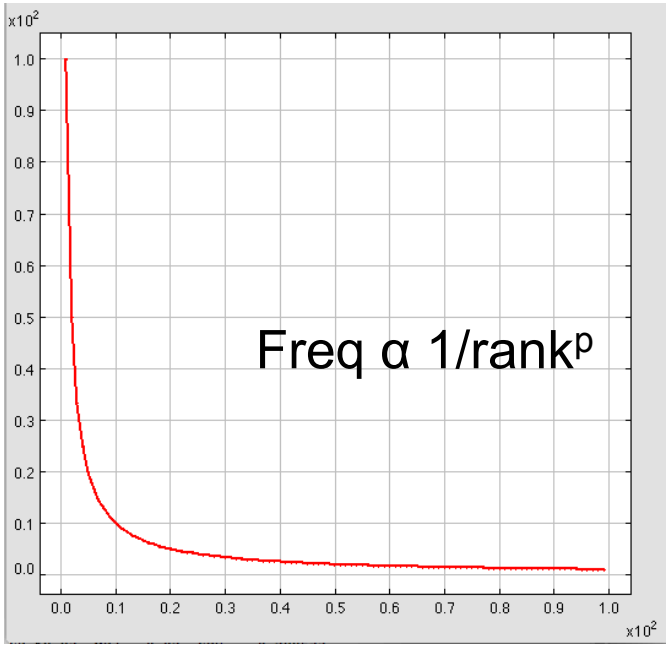
DNNs for Power-Law data

MCMC for massive datasets

Power-Law Features

Big Data about people (text, web, social media) follow power law statistics:

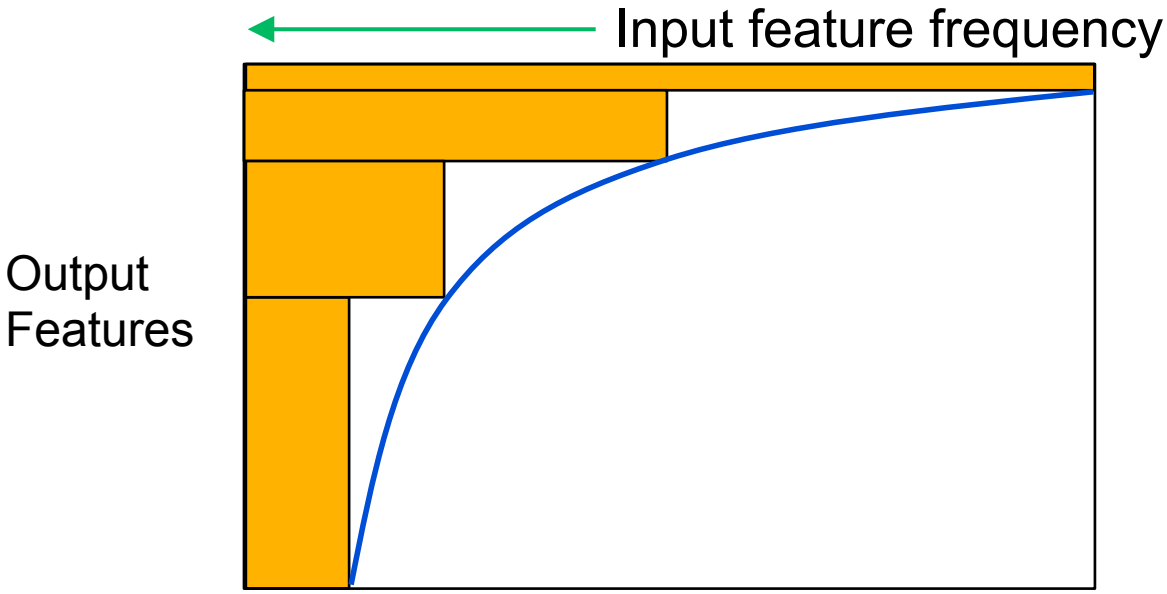
Feature frequency



Feature rank

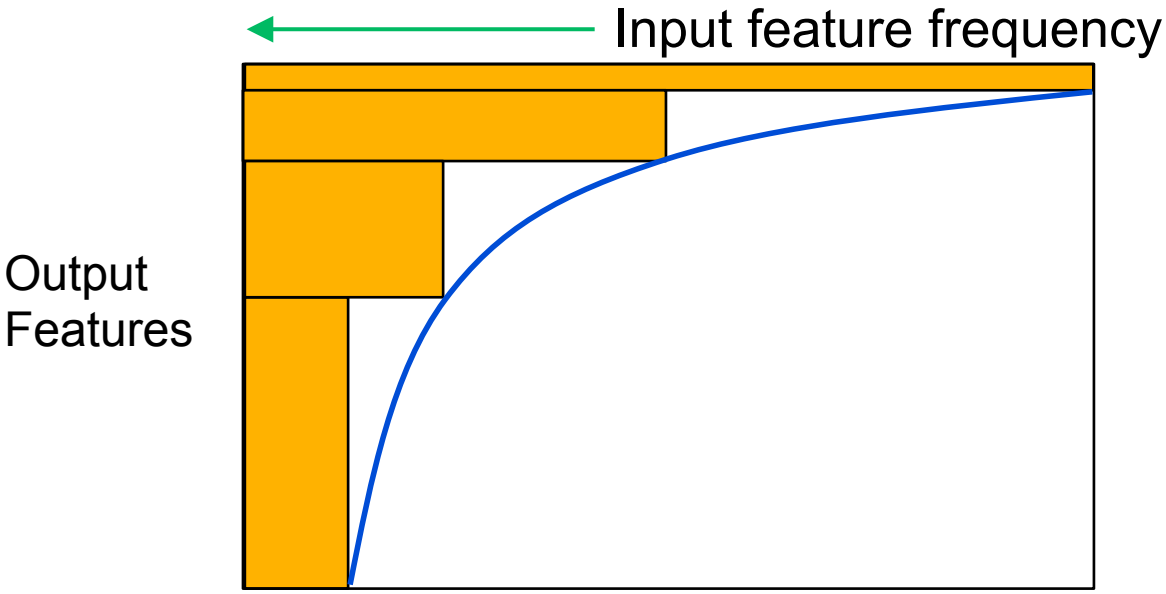
Feature sorted by frequency descending

DNNs for Power-law data



- “Powerlayers” include linear maps built from rectangular tiles with power law shape.
- Used as input layers in regression problems or as input/ output layers in sequence LSTMs.

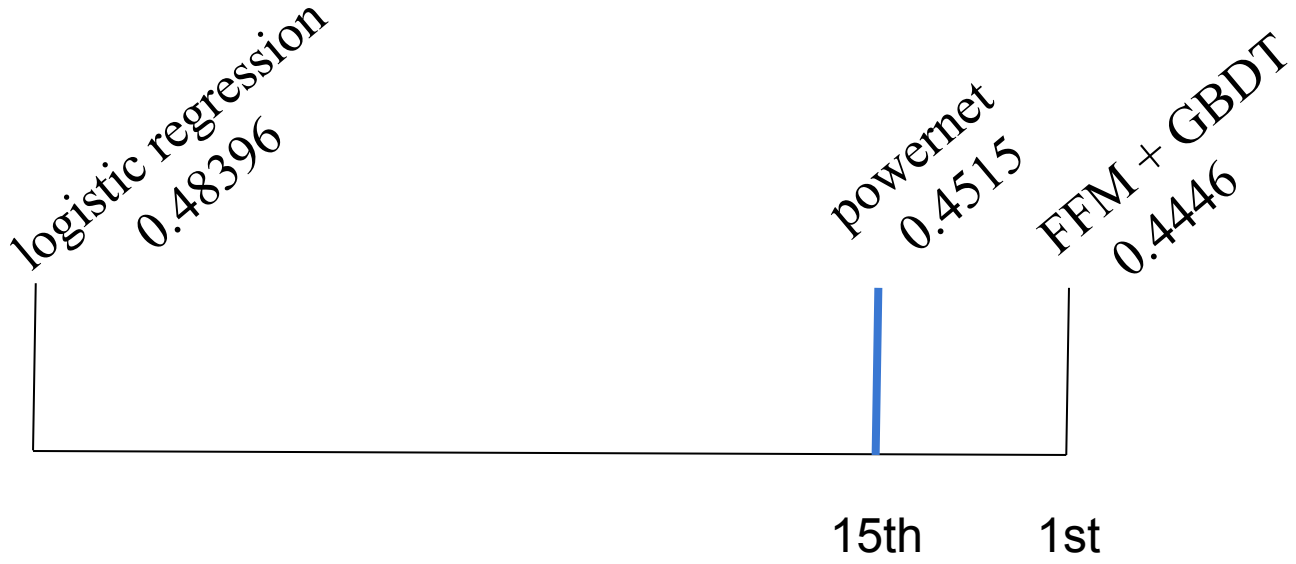
DNNs for Power-law data



- We can solve the following optimization problem:
  - Which N coefficients should be keep to produce the best low-dimensional approximation to the original data?
- The solution uses an SVD of the full matrix. For typical data:
  - The sequence of singular value magnitudes follows a power-law.
- It follows that the envelope of non-zeros follows a power-law.

Performance on Criteo 20 GB

- Criteo released a clickthrough dataset which was used for a Kaggle competition.
- The dataset has 37 million distinct features, about 2.5 billion features total.
- Preliminary results on a simple 8-layer, full-connected network with power-law input layer:



Outline

BIDMach on single machines

BIDMach on clusters

DNNs for Power-Law data

MCMC for massive datasets

MCMC for Massive Datasets

Why?

- We want to build good models.
- Model parameter spaces complex, multimodal.
- We want to exploit cluster computing as search, (Elastic Averaging SGD).
- MCMC methods keep us on track in searching the parameter space, allowing aggressive moves.

MCMC for Massive Datasets

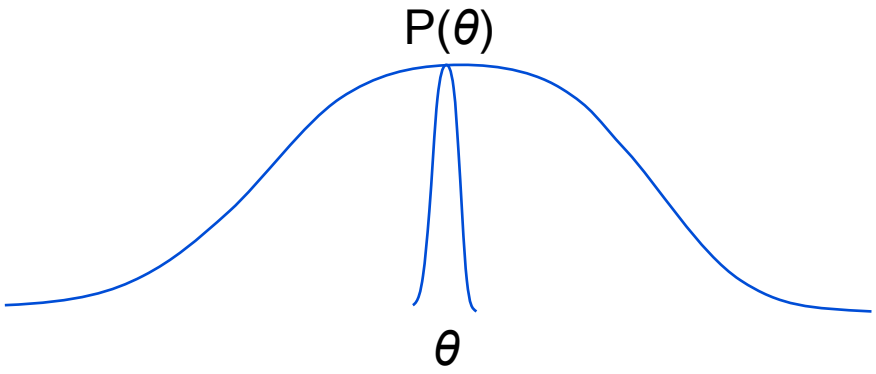
Bernstein Von-Mises Theorem:

P(θ) is the likelihood of model parameters θ. It is asymptotically normal with variance 1/N for N datapoints.

Not that useful (or practical) to just sample from P(θ).

MCMC for Massive Datasets

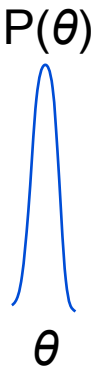
Heating/Annealing



Heating scales the log likelihood. Typically smooths the likelihood landscape, improves accuracy of large steps.

MCMC for Massive Datasets

Scaling the step size:



We cant take large steps (relative to the posterior) using the information in a small minibatch (not enough information to find the mode).

But we can take smaller steps.

Metropolis-Hastings

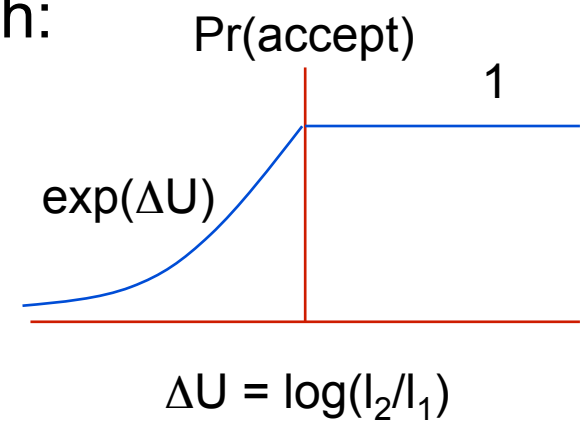
- From some state θ, propose a new state θ'.

- Based on a simple test on the likelihoods $p(\theta)$ and $p(\theta')$, decide to either accept (move to) θ' or stay at θ.

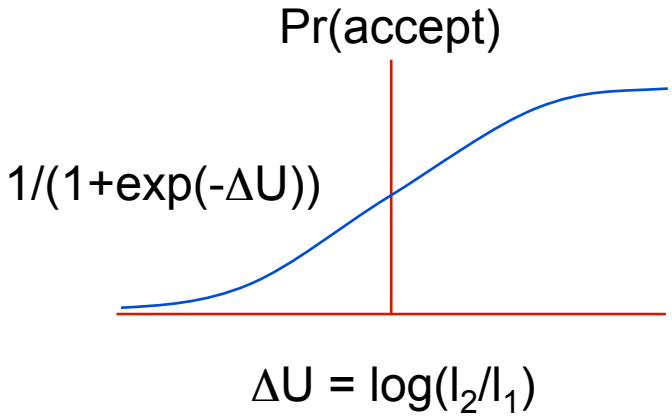
Ensures that the sequences of samples θ come from the target distribution.

Minibatch Metropolis Hastings

The classical MH test has an acceptance probability which is asymmetric and non-smooth:

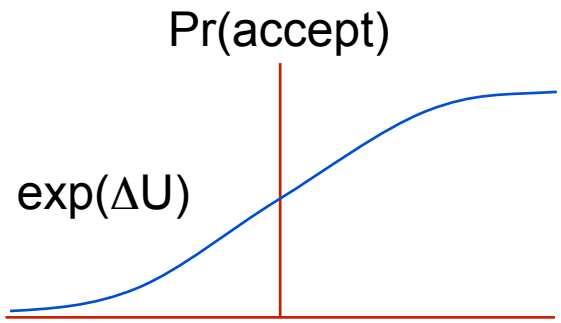


An alternative smooth, symmetric distribution is the logistic function (Barker's test):

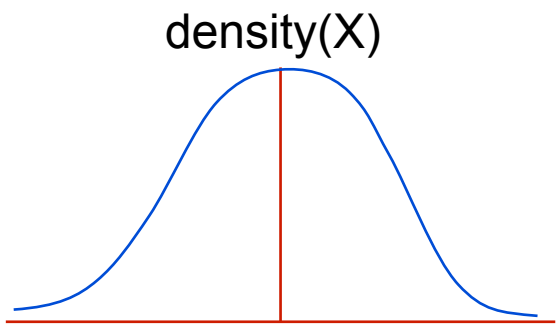


Minibatch Metropolis Hastings

Testing against the smooth distribution can be done using a random variable X whose CDF is the acceptance curve:



$\Delta U = \log(l_2/l_1)$

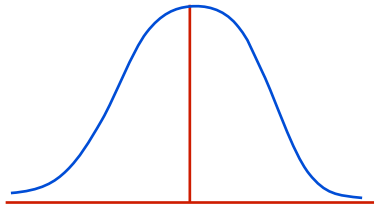


X

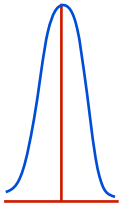
Accept if
$\Delta U + X > 0$

This allows us to use the minibatch-induced variance in likelihood estimates to provide the variation for the MH test.

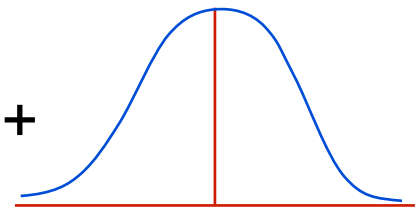
ΔU +



+



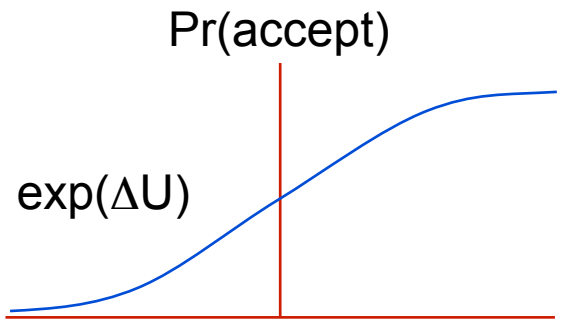
= ΔU +



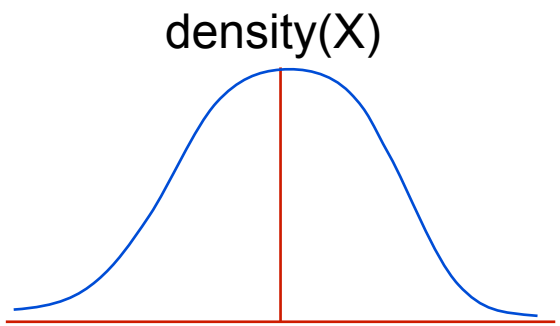
Noise_U (normal) $X_{\text{correction}}$ X (logistic')

Minibatch Metropolis Hastings

Testing against the smooth distribution can be done using a random variable X whose CDF is the acceptance curve:



$\Delta U = \log(l_2/l_1)$

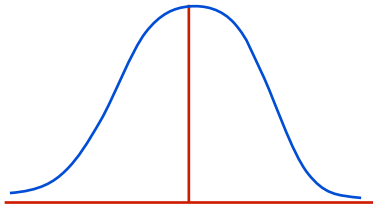


X

Accept if
$\Delta U + X > 0$

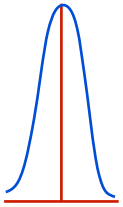
This allows us to use the minibatch-induced variance in likelihood estimates to provide the variation for the MH test.

ΔU +



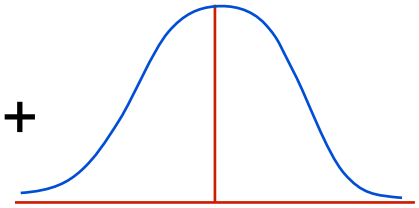
Noise_U (normal)

+



$X_{correction}$

= ΔU +



X (logistic')

Minibatch Metropolis Hastings

- As long as the variance condition is satisfied:
  - Temperature is high enough, OR
  - Step size is small enough

We can perform an M-H test with any desired minibatch size.

Achieves arbitrary speedups over previous approaches.

Allows risky explorations with parallel optimization moves.

Opportunistic MCMC

With a fast MH test in hand, we can explore non-conservative moves during optimization:

- Max: Each machine in a group moves toward the best parameter value in the group.
- Mix: Each machine moves toward the average parameter value (Elastic Averaging SGD).

Summary

BIDMach on single machines

BIDMach on clusters

DNNs for Power-Law data

MCMC for massive datasets