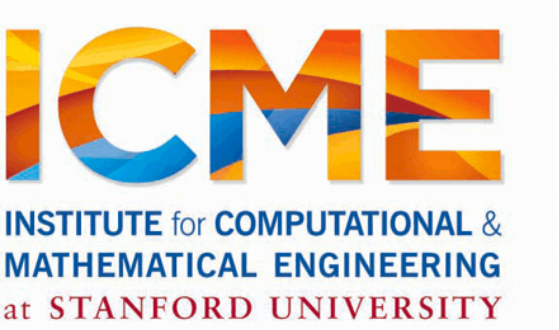
Matrix Computations and Neural Networks in Spark

Reza Zadeh

Paper: http://arxiv.org/abs/1509.02256
Joint work with many folks on paper.

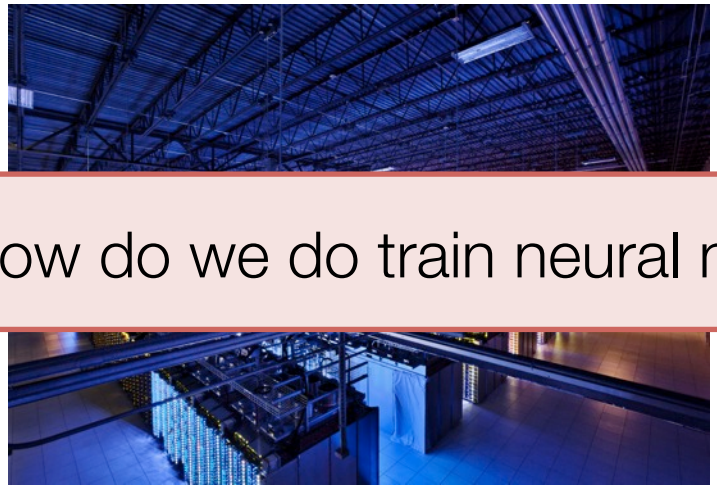


@Reza_Zadeh | http://reza-zadeh.com

Training Neural Networks

Datasets growing faster than processing speeds

Solution is to parallelize on clusters and GPUs



How do we do train neural networks on these things?

Outline

Resilient Distributed Datasets and Spark

Key idea behind mllib.linalg

The Three Dimensions of Machine Learning

Neural Networks in Spark 1.5

Local Linear Algebra

Future of Deep Learning in Spark

Traditional Network Programming

Message-passing between nodes (e.g. MPI)

Very difficult to do at scale:

- How to split problem across nodes?
  - Must consider network & data locality
- How to deal with failures? (inevitable at scale)
- Even worse: stragglers (node not failed, but slow)
- Ethernet networking not fast
- Have to write programs for each machine

Rarely used in commodity datacenters

Spark Computing Engine

Extends a programming language with a distributed collection data-structure

- » “Resilient distributed datasets” (RDD)

Open source at Apache

- » Most active community in big data, with 75+ companies contributing

Clean APIs in Java, Scala, Python, R

Resilient Distributed Datasets (RDDs)

Main idea: Resilient Distributed Datasets

» Immutable collections of objects, spread across cluster

» Statically typed: RDD[T] has objects of type T

```
val sc = new SparkContext()
val lines = sc.textFile("log.txt")   // RDD[String]

// Transform using standard collection operations
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split('\t')(2))

messages.saveAsTextFile("errors.txt")
```

→ lazily evaluated

→ kicks off a computation

MLlib: Available algorithms

classification: logistic regression, linear SVM, naïve Bayes, least squares, classification tree, neural networks

regression: generalized linear models (GLMs), regression tree

collaborative filtering: alternating least squares (ALS), non-negative matrix factorization (NMF)

clustering: k-means||

decomposition: SVD, PCA

optimization: stochastic gradient descent, L-BFGS

Key idea in mllib.linalg

Key Idea

Distribute matrix across the cluster

Ship computations involving matrix to cluster

Keep vector operations local to driver

Simple Observation

Matrices are often quadratically larger than vectors

A: n x n (matrix) $O(n^2)$

v: n x 1 (vector) O(n)

Even n = 1 million makes cluster necessary

Distributing Matrices

How to distribute a matrix across machines?

» By Entries (CoordinateMatrix)

» By Rows (RowMatrix)

» By Blocks (BlockMatrix) As of version 1.3

All of Linear Algebra to be rebuilt using these partitioning schemes

Distributing Matrices

Even the simplest operations require thinking about communication e.g. multiplication

How many different matrix multiplies needed?

» At least one per pair of {Coordinate, Row, Block, LocalDense, LocalSparse} = 10

» More because multiplies not commutative

Example mllib.linalg algorithms

- Matrix Multiplication
- Singular Value Decomposition (SVD)
- QR decomposition
- Optimization primitives
- … yours?

Simple idea goes a long way

The Three Dimensions of *Scalable* Machine Learning (and Deep Learning)

ML Objectives

Almost all machine learning objectives are optimized using this update

$$w \leftarrow w - \alpha \cdot \sum_{i=1}^{n} g(w; x_i, y_i)$$

w is a vector of dimension d
we're trying to find the best w via optimization

Scaling Dimensions

1) Data size: n

$$w \leftarrow w - \alpha \cdot \sum_{i=1}^{n} g(w; x_i, y_i)$$

2) Number of models: hyperparameters

3) Model size: d

Neural Networks Data Scaling

$$w \leftarrow w - \alpha \cdot \sum_{i=1}^{n} g(w; x_i, y_i)$$

```
val points = sc.textFile(...).map(parsePoint).cache()

// Initialize w to a random value
var w = DenseVector.fill(D){rand.nextDouble}

for (i <- 1 to ITERATIONS) {
  val gradient = points.map {
    p => ForwardBackward(p.x, p.y, w) }.reduce(_ + _)
  w -= gradient // gradient descent update
}
```

Separable Updates

Can be generalized for

- Unconstrained optimization
- Smooth or non-smooth
- LBFGS, Conjugate Gradient, Accelerated Gradient methods, …

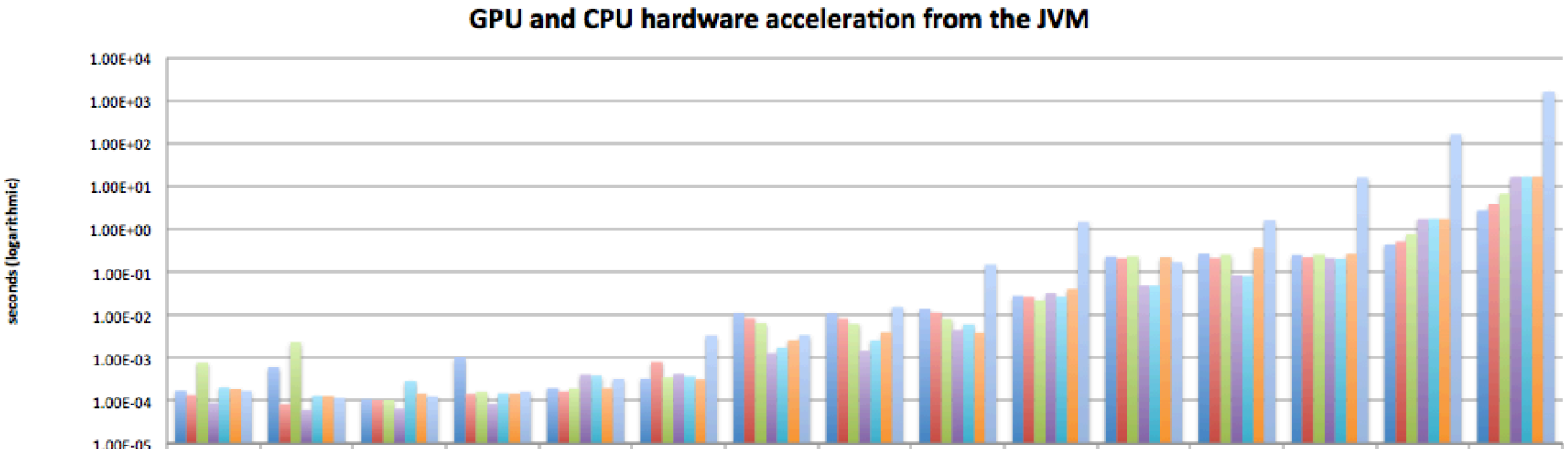
Local Linear Algebra

Why local linear algebra?

GPU and CPU hardware acceleration can be tricky on the JVM

Forward and Backward propagation in Neural Networks and many other places need local acceleration

Comprehensive Benchmarks



	1x1*1x1	10x10*10x1	10x10*10x10	100x100*100x1	100x100*100x10	100x100*100x100	1000x1000*1000x1	1000x1000*1000x10	1000x1000*1000x100	1000x1000*1000x1000	10000x10000*10000x1	10000x10000*10000x10	10000x10000*10000x100	10000x10000*10000x1000	10000x10000*10000x10000
netlib-nvblas 3 GPU	1.67E-04	5.93E-04	1.07E-04	1.01E-03	1.98E-04	3.27E-04	1.09E-02	1.09E-02	1.38E-02	2.74E-02	2.27E-01	2.62E-01	2.47E-01	4.39E-01	2.75E+00
netlib-nvblas 2 GPU	1.36E-04	8.26E-05	1.02E-04	1.43E-04	1.61E-04	8.02E-04	8.13E-03	8.02E-03	1.12E-02	2.61E-02	2.09E-01	2.12E-01	2.20E-01	5.14E-01	3.73E+00
netlib-nvblas 1 GPU	7.74E-04	2.29E-03	1.03E-04	1.58E-04	1.95E-04	3.48E-04	6.44E-03	6.23E-03	7.95E-03	2.14E-02	2.32E-01	2.51E-01	2.53E-01	7.69E-01	6.73E+00
BIDMat-MKL	8.85E-05	6.11E-05	6.46E-05	8.43E-05	3.98E-04	4.09E-04	1.27E-03	1.42E-03	4.42E-03	3.13E-02	4.79E-02	8.39E-02	2.11E-01	1.72E+00	1.65E+01
netlib-MKL	2.04E-04	1.30E-04	2.88E-04	1.48E-04	3.80E-04	3.66E-04	1.74E-03	2.54E-03	6.07E-03	2.63E-02	4.78E-02	8.09E-02	2.04E-01	1.74E+00	1.64E+01
netlib OpenBLAS	1.90E-04	1.29E-04	1.45E-04	1.45E-04	1.98E-04	3.15E-04	2.56E-03	3.96E-03	3.88E-03	4.01E-02	2.20E-01	3.63E-01	2.60E-01	1.73E+00	1.65E+01
netlib-f2jblas	1.65E-04	1.15E-04	1.26E-04	1.61E-04	3.22E-04	3.28E-03	3.35E-03	1.54E-02	1.49E-01	1.45E+00	1.66E-01	1.61E+00	1.61E+01	1.61E+02	1.61E+03

From our paper: http://arxiv.org/abs/1509.02256

Future of the Spark and Neural Networks

Coming for Neural Networks

Future versions will include

» Convolutional Neural Networks (Goal: 1.6)

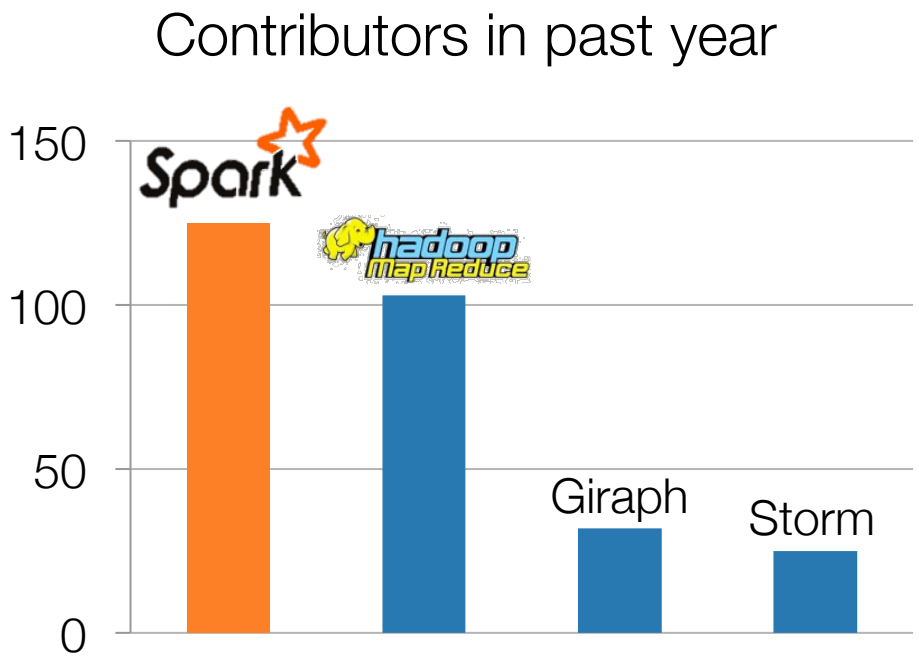
» Dropout, different layer types (Goal: 1.6)

» Recurrent Neural Networks and LSTMs (Goal: 1.7)

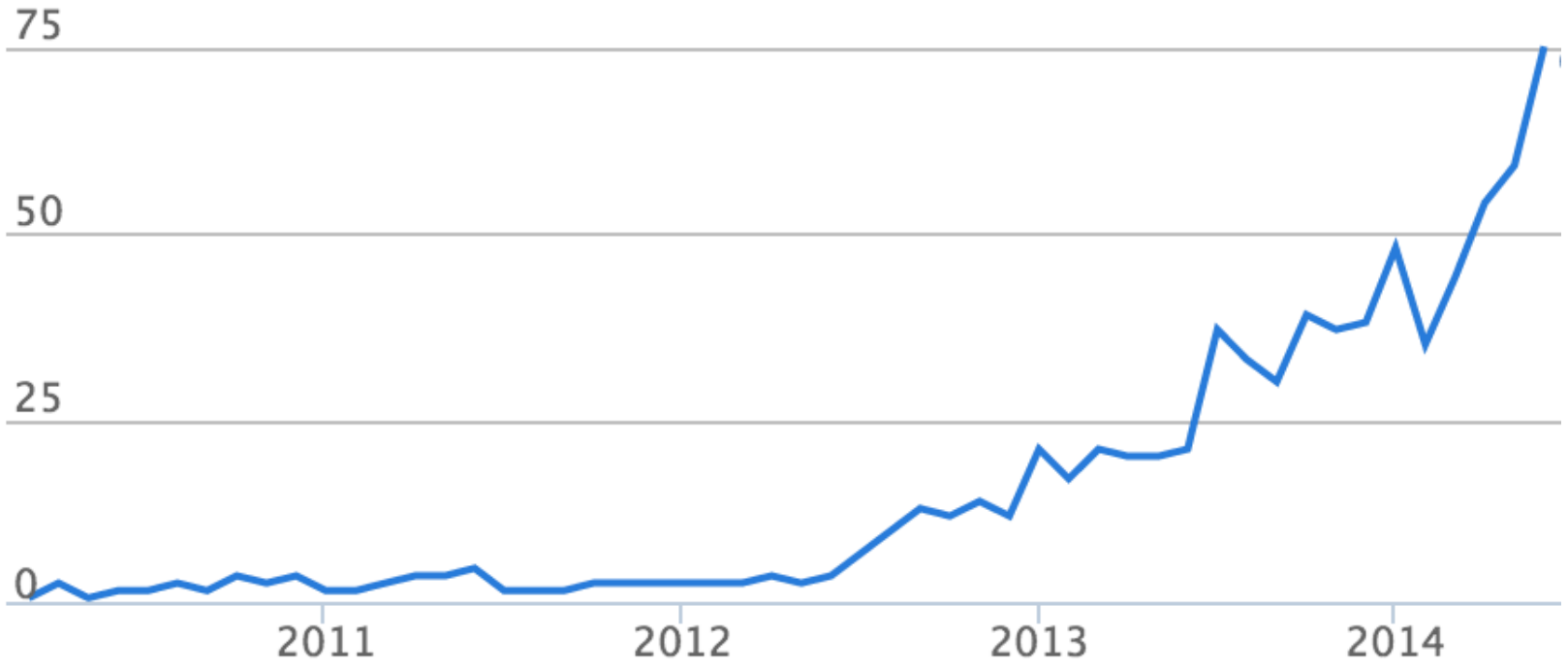
Spark Community

Most active open source community in big data

200+ developers, 50+ companies contributing



Continuing Growth



Contributors per month to Spark

source: ohloh.net

Conclusions

Spark and Research

Spark has all its roots in research, so we hope to keep incorporating new ideas!

Conclusion

Data flow engines are becoming an important platform for machine learning

More info: spark.apache.org