Matrix Computations and Neural Networks in Spark

Reza Zadeh

Joint work with many folks on paper.
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

```scala
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: \( n \times n \) (matrix) \( O(n^2) \)

v: \( n \times 1 \) (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 \{\text{Coordinate}, \text{Row}, \text{Block}, \text{LocalDense}, \text{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

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