Machine Learning for Large-Scale Data Analysis and Decision Making
80-629-17A

Distributed Machine Learning — Week #9
Today

- Distributed computing for machine learning
  - Background
  - MapReduce/Hadoop & Spark
    - Theory & Practice

- Note: Most lectures so far used stats concepts. Today we’ll turn to computer science.
Modern view of ML

• Understand massive quantities of data

• Google (4K searches/s), Twitter: (6K tweets/s), Amazon: (100s sold products/s) (source: internetlifestats.com)

• Banks, insurance companies, etc.

• Modestly-sized websites

• Both large n and large p

• Computation will scale up with the data

• Often fitting an ML models requires one or multiple operations that looks at the whole dataset
Modern view of ML

- Understand massive quantities of data

- Google (4K searches/s), Twitter: (6K tweets/s), Amazon: (100s sold products/s)
  (source: internetlifestats.com)

- Banks, insurance companies, etc.

- Modestly-sized websites

- Both large n and large p

- Computation will scale up with the data

- Often fitting an ML models requires one or multiple operations that looks at the whole dataset

\[
\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}
\]

Linear regression
Issues with massive datasets

1. Storage

2. Computation
Moore’s Law

Microprocessor Transistor Counts 1971-2011 & Moore’s Law

[https://en.wikipedia.org/wiki/Moore%27s_Law]
120 Years of Moore’s Law

Source: Ray Kurzweil, DFJ
Modern Computation paradigms
Modern Computation paradigms

- Floating point operations per second (Flop)
- Smart phone ~ 0.005 TFlops
- 1 Tera: 1,000 Giga
Modern Computation paradigms

1. “Single” computers
   - Large Computers
     - 125 435.9 TFlops

- Floating point operations per second (Flop)
- Smart phone ~ 0.005 TFlops
- 1 Tera: 1,000 Giga
Modern Computation paradigms

1. “Single” computers
   - Large Computers
     - 125 435.9 TFlops

2. Specialized hardware
   - Focusses on subset of operations
   - Graphical Processing Unit (GPU), Field Programmable Gated Array (FPGA)
     - 7.5 TFlops

- Floating point operations per second (Flop)
- Smart phone ~ 0.005 TFlops
- 1 Tera: 1,000 Giga
Modern Computation paradigms

1. “Single” computers
   - Large Computers
     - 125,435.9 TFlops

2. Specialized hardware
   - Focusses on subset of operations
     - Graphical Processing Unit (GPU), Field Programmable Gated Array (FPGA)
     - 7.5 TFlops

2. Distributed computation
   - ~100,000 TFlops (Folding@home)

- Floating point operations per second (Flop)
- Smart phone ~ 0.005 TFlops
- 1 Tera: 1,000 Giga
Modern Computation paradigms

1. **“Single” computers**
   - Large Computers
   - 125 435.9 TFlops

2. **Specialized hardware**
   - Focusses on subset of operations
   - Graphical Processing Unit (GPU), Field Programmable Gated Array (FPGA)
   - 7.5 TFlops

2. **Distributed computation**
   - ~100 000 TFlops (Folding@home)

- Floating point operations per second (Flop)
- Smart phone ~ 0.005 TFlops
- 1 Tera: 1,000 Giga
Distributed Computing
Distributed Computing

- Faster computers can help
Distributed Computing

- Faster computers can help
- What about a large of “slow” computers working together?
  - Divide the computation into small problems
    1. All (slow) computers solve a small problem at the same time
    2. Combine the solution of small problems into initial solution
Distributed Computing

- Faster computers can help
- What about a large of “slow” computers working together?
- Divide the computation into small problems

1. All (slow) computers solve a small problem at the same time
2. Combine the solution of small problems into initial solution
Simple example

• You are tasked with counting the number of houses in Montreal

1. Centralized (single computer):
   • Ask a marathon runner to jog around the city and count
   • Build a system to count houses from satellite imagery
Simple example

• You are tasked with counting the number of houses in Montreal

1. Centralized (single computer):
   • Ask a marathon runner to jog around the city and count
   • Build a system to count houses from satellite imagery

2. Distributed (many computers):
   • Ask 1,000 people to each count houses from a small geographical area
   • Once they are done they report their result at your HQ
Tool for distributed computing (for machine learning)

  - Builds on MapReduce ideas
  - More flexible computation graphs
- High-level APIs
  - MLlib
Outline

- MapReduce
  - Fundamentals and bag-of-words example
- Spark
  - Fundamentals & MLlib
  - Live examples
MapReduce

- From Google engineers

  “MapReduce: Simplified Data Processing on Large Clusters”, Jeffrey Dean and Sanjay Ghemawat, 2004

- Now also known as (Apache) Hadoop

- Google built large-scale computation from commodity hardware

- Specific distributed interface

- Useful for algorithms that can be expressed using this interface
MapReduce

- Two types of tasks:
  
  A. Map: Solve a subproblem
  
  B. Reduce: Combine the results of map workers
TASK: Create a document's bag-of-word representation

A. Map
- The black dog
- A black cat
- The blue cat

B. Reduce
TASK: Create a document's bag-of-word representation

The black dog
A black cat
The blue cat

A. Map

B. Reduce

The black dog
A black cat
The blue cat
The black dog
A black cat
The blue cat

A. Map

B. Reduce

The, 1
black, 1
dog, 1

A, 1
black, 1
cat, 1

The, 1
blue, 1
cat, 1
A. Map

The black dog
A black cat
The blue cat

B. Reduce

Sort by key

The, 1
dog, 1

A, 1
black, 1
cat, 1

The, 1
blue, 1
cat, 1
TASK: Create a document's bag-of-word representation.
Some details

- Typically the number of subproblems is higher than the number of available machines

  - \( \sim \)linear speed-up wrt to the number of machines

- If a node crashes, need to recompute its subproblem

- Input/Output

  - Data is read from disk when beginning

  - Data is written to disk at the end
MapReduce is quite versatile

• When I was at Google the saying was:

  “If your problem cannot be framed as MapReduce you haven’t thought hard enough about it.”

• A few examples of “map-reduceable” problems:

  • Sorting, filtering, distinct values, basic statistics

  • Finding common friends, sql-like queries, sentiment analysis
MapReduce for machine learning

- Training linear regression
  - Reminder: there is a closed form solution
    \[
    w = (X^TX)^{-1}X^TY
    \]
MapReduce for machine learning

• Training linear regression

• Reminder: there is a closed form solution

\[ w = (X^TX)^{-1}X^TY \]

\[ w = \left( \sum_{ij} X_i^T X_j \right)^{-1} \left( \sum_{i} X_i^T Y_i \right) \]
MapReduce for machine learning

- Training linear regression

- Reminder: there is a closed form solution

\[ w = (X^TX)^{-1}X^TY \]

\[ w = \left( \sum_{ij} X_i^T X_j \right)^{-1} \left( \sum_i X_i^T Y_i \right) \]

- Each term in the sums can be computed independently
MapReduce for machine learning

- Training linear regression

- Reminder: there is a closed form solution

\[
\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}
\]

\[
\mathbf{w} = (\sum_{ij} \mathbf{x}_i^\top \mathbf{x}_j)^{-1} (\sum_i \mathbf{x}_i^\top \mathbf{y}_i)
\]

- Each term in the sums can be computer independently

A. Map

\[
\begin{bmatrix}
\mathbf{X}_0^\top
\mathbf{X}_1
\end{bmatrix}
\]
MapReduce for machine learning

• Training linear regression

• Reminder: there is a closed form solution

$$w = (X^TX)^{-1}X^TY$$

$$w = (\sum_{ij} X_i^TX_j)^{-1}(\sum_{i} X_i^TY_i)$$

• Each term in the sums can be computed independently

• Other models we studies have a closed form solution (e.g., Naive Bayes and LDA)
MapReduce for machine learning

- Training linear regression

- Reminder: there is a closed form solution
  \[ w = (X^T X)^{-1} X^T Y \]

- Other models we studies have a closed form solution (e.g., Naive Bayes and LDA)

- Hyper-parameter search
  - A neural network with 2 hidden layers and 5 hidden units per layer and another with 3 hidden layers and 10 hidden units

Each term in the sums can be computed independently.
Shortcomings of MapReduce

• Many models are fitted with iterative algorithms

  • Gradient descent:
    1. Find the gradient for the current set parameters
    2. Update the parameters with the gradient

• Not ideal for MapReduce

  • Would require several iterations of MapReduce
  • Each time the data is read/written from/to the disk
(Apache) Spark

- Advantages over MapReduce

  1. Less restrictive computations graph (DAG instead of Map then Reduce)
     - Doesn’t have to write to disk in-between operations

  2. Richer set of transformations
     - map, filter, cartesian, union, intersection, distinct, etc.

  3. In-memory processing
Spark History

- Started in Berkeley’s AMPLab (2009)
- Version 1.0 2014
  - Based on Resilient Distributed Datasets (RDDs)
- Version 2.0 June 2016
- Version 2.3 February 2018
- We will be using pySpark
- Best (current) documentation:
  2. Project docs: https://spark.apache.org/docs/latest/
Resilient Distributed Datasets (RDDs)

- A data abstraction
  - Collection of partitions. Partitions are the distribution unit.
  - Operations on RDDs are (automatically) distributed.
- RDDs support two types of operations:
  1. Transformations
     - Transform a dataset and return it
  2. Actions
     - Compute a result based on an RDD
     - These operations can then be “chained” into complex execution flows
DataFrames

- An extra abstraction on top of RDDs
- Encodes rows as a set of columns
  - Each column has a defined type
- Useful for (pre-processed) machine learning datasets
- Same name as `data.frame` (R) or `pandas.DataFrame`
  - Similar type of abstraction but for distributed datasets
- Two types of actions (for our needs): transformers, estimators.
Spark’s “Hello World”

data = spark.read.format("libsvm").load("hdfs://...")

model = LogisticRegression(regParam=0.01).fit(data)
Spark’s “Hello World”

```python
DataFrame → data = spark.read.format("libsvm").load("hdfs://...")

define the model

model = LogisticRegression(regParam=0.01).fit(data)
```
Spark’s “Hello World”

```python
from pyspark.ml.classification import LogisticRegression
from pyspark.ml import Pipeline

# Load data from HDFS
data = spark.read.format("libsvm").load("hdfs://...")

# Create a pipeline with a Logistic Regression estimator
estimator = LogisticRegression(regParam=0.01)

# Fit the model to the data
model = estimator.fit(data)
```

---

DataFrame → data = spark.read.format("libsvm").load("hdfs://...")

model = LogisticRegression(regParam=0.01).fit(data)
Parallel gradient descent

- Logistic Regression

\[ y = \frac{1}{1 + \exp(-w_0 - w_1 x_1 - w_2 x_2 - \cdots - w_p x_p)} \]
Parallel gradient descent

- Logistic Regression

\[ y = \frac{1}{1 + \exp(-w_0 - w_1x_1 - w_2x_2 - \ldots - w_px_p)} \]

- No closed-form solution, can use gradients

\[ \frac{\partial \text{Loss}(Y, X, w)}{\partial w_i} \]
Parallel gradient descent

- Logistic Regression

\[ y = \frac{1}{1 + \exp(-w_0 - w_1 x_1 - w_2 x_2 - \ldots - w_p x_p)} \]

- No closed-form solution, can use gradients

\[ \frac{\partial \text{Loss}(Y, X, w)}{\partial w_i} \]

- Loss functions are often decomposable

\[ \frac{\partial}{\partial w_i} \sum_j \text{Loss}(Y_j, X_j, w) \]
Parallel gradient descent

- Logistic Regression

\[ y = \frac{1}{1 + \exp(-w_0 - w_1 x_1 - w_2 x_2 - \ldots - w_p x_p)} \]

- No closed-form solution, can use gradients

\[ \frac{\partial \text{Loss}(Y, X, w)}{\partial w_i} \]

- Loss functions are often decomposable

\[ \frac{\partial \sum_j \text{Loss}(Y_j, X_j, w)}{\partial w_i} \]
Demo

- Fitting Logistic Regression

- Dataset:
  - Ad clicks given ad and user features
  - 45M observations, 39 features (13 continuous, 26 categorical)

[https://www.kaggle.com/c/criteo-display-ad-challenge]
Resources for 80-629 (think of your project)

• Ask me for an account
  • Notebook access
  • Available kernels: python2, python3, pySpark
• Your account is on a Google cloud cluster
Takeaways

• Distributed computing is useful:
  • for large-scale data
  • for faster computing

• Current frameworks (e.g., spark) offer easy access to popular ML models + algorithms
  • Useful speedups by decomposing the computation into a number of identical smaller pieces

• Still requires some engineering/coding
Administrative matters

• Final presentation
  In front of class or poster session?

• Final exam
  With or without documentation?