Big-data Machine Learning

Chih-Jen Lin
Department of Computer Science
National Taiwan University

Talk at eBay Data Summit, October 25, 2014

Outline

1. Introduction
2. Algorithms for distributed data analytics
3. Discussion and conclusions

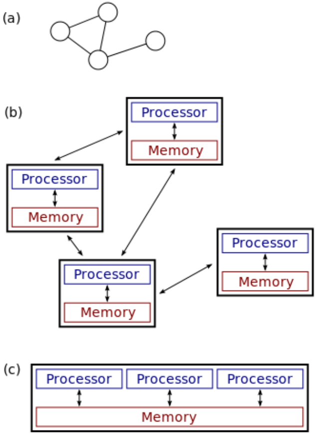
Outline

Big Data

- People talk about volume, velocity, variety, value and veracity
- All are important, but today we will focus on volume

Big-data Machine Learning

- We consider the situation that data are larger than the capacity of a computer
- So in a sense we are talking about distributed data mining or machine learning
- Today we won't talk about things such as GPU-based machine learning



(a), (b): distributed systems

Image from Wikimedia

From Small to Big Data

Two important differences:

Negative side:

- Methods for big data analytics are not quite ready, not even mentioned to integrated tools

Positive side:

- Some (Halevy et al., 2009) argue that the almost unlimited data make us easier to mine information

I will discuss the first difference

Possible Advantages of Distributed Data Analytics

Parallel data loading

- Reading several TB data from disk is slow
- Using 100 machines, each has $1/100$ data in its local disk $\Rightarrow$ $1/100$ loading time
- But having data ready in these 100 machines is another issue

Fault tolerance

- Some data replicated across machines: if one fails, others are still available

Possible Advantages of Distributed Data Analytics (Cont'd)

Workflow not interrupted

- If data are already distributedly stored, it's not convenient to reduce some to one machine for analysis

Possible Disadvantages of Distributed Data Analytics

- More complicated (of course)

  Note that you can always subsample data to one machine for deep analysis
- Communication and synchronization

  Everybody says moving computation to data, but this isn't that easy

Going Distributed or Not Isn't Easy to Decide

- Quote from Yann LeCun (KDnuggets News 14:n05)
  "I have seen people insisting on using Hadoop for datasets that could easily fit on a flash drive and could easily be processed on a laptop."
- Now disk and RAM are large. You may load several TB of data once and conveniently conduct all analysis
- The decision is application dependent

Challenges in Distributed Environments

We must consider

- Computation time
- Loading time
- Communication/synchronization time

In the past, we focus only on the computation time

Loading time

We use the following example to illustrate that computation time is not the only concern

- Using a linear classifier LIBLINEAR (Fan et al., 2008) to train the rcv1 document data sets (Lewis et al., 2004).
- # instances: 677,399, # features: 47,236
- On a typical PC: Total time: 50.88 seconds. Loading time: 43.51 seconds

In fact, 2 seconds are enough to get stable test accuracy

loading time $\gg$ running time

Loading Time (Cont'd)

- To see why this happens, let's discuss the complexity
- Assume the memory hierarchy contains only disk and number of instances is l
- Loading time: $l \times$ (a big constant)

  Running time: $l^q \times$ (some constant), where $q \geq 1$.
- Traditionally running time is larger because of using nonlinear algorithms (i.e., $q > 1$)
- But when l is large, we may use a linear algorithm (i.e., $q = 1$) for efficiency $\Rightarrow$ loading time may dominate

Small Analysis versus Big Analysis

- Big data, small analysis

  versus

  Big data, big analysis
- If you need a single record from a huge set, it's reasonably easy
- For example, accessing your high-speed rail reservation is fast
- However, if you want to analyze the whole set by accessing data several time, it can be much harder

Outline

1. Introduction
2. Algorithms for distributed data analytics
   - Example: a distributed Newton method for logistic regression
   - Other existing implementations
   - Successful applications
3. Discussion and conclusions

Existing Algorithms on One Machine

- Most existing data mining/machine learning methods were designed without considering data access and communication of intermediate results
- They iteratively use data by assuming they are readily available
- Example: doing least-square regression isn't easy in a distributed environment

Algorithms for Distributed Data Analytics

This is an on-going research topic.

Roughly there are two types of approaches

1. Parallelize existing (single-machine) algorithms
2. Design new algorithms particularly for distributed settings

Of course there are things in between

Outline

Logistic Regression

- Training data $\{y_i, \boldsymbol{x}_i\}, \boldsymbol{x}_i \in R^n, i = 1, \ldots, l, \; y_i = \pm 1$
- l: # of data, n: # of features
- Regularized logistic regression

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}),$$

where

$$f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l} \log\left(1 + e^{-y_i \boldsymbol{w}^T\boldsymbol{x}_i}\right).$$

- C: regularization parameter decided by users
- Twice differentiable, so we can use Newton methods

Newton Methods

- Newton direction

$$\min_{\boldsymbol{s}} \quad \nabla f(\boldsymbol{w}^k)^T \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T \nabla^2 f(\boldsymbol{w}^k)\boldsymbol{s}$$

- This is the same as solving Newton linear system

$$\nabla^2 f(\boldsymbol{w}^k)\boldsymbol{s} = -\nabla f(\boldsymbol{w}^k)$$

- Hessian matrix $\nabla^2 f(\boldsymbol{w}^k)$ too large to be stored

$$\nabla^2 f(\boldsymbol{w}^k) : n \times n, \quad n : \text{ number of features}$$

- But Hessian has a special form

$$\nabla^2 f(\boldsymbol{w}) = \mathcal{I} + CX^T DX,$$

Newton Methods (Cont'd)

- X: data matrix. D diagonal with

$$D_{ii} = \frac{e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}}{(1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i})^2}$$

- Using Conjugate Gradient (CG) to solve the linear system. Only Hessian-vector products are needed

$$\nabla^2 f(\boldsymbol{w})\boldsymbol{s} = \boldsymbol{s} + C \cdot X^T(D(X\boldsymbol{s}))$$

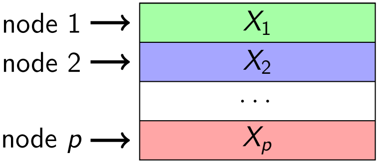
- Therefore, we have a Hessian-free approach
- Other details; see Lin et al. (2008) and the software LIBLINEAR

Parallel Hessian-vector Product

- Hessian-vector products are the computational bottleneck

$$X^TDX\boldsymbol{s}$$

- Data matrix X is now distributedly stored



$$X^TDX\boldsymbol{s} = X_1^TD_1X_1\boldsymbol{s} + \cdots + X_p^TD_pX_p\boldsymbol{s}$$

Distributed LIBLINEAR

- The above method is now available as an extension of the software LIBLINEAR
- See `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/distributed-liblinear`
- We support both MPI and Spark
- The development is still in an early stage.

Outline

1. Introduction

2. Algorithms for distributed data analytics
   - Example: a distributed Newton method for logistic regression
   - Other existing implementations
   - Successful applications

3. Discussion and conclusions

Vowpal_Wabbit (Langford et al., 2007)

- It started as a linear classification package on a single computer
- After version 6.0, Hadoop support has been provided
- A hybrid approach: parallel stochastic gradient initially and switch to LBFGS (quasi Newton)
- In Agarwal et al. (2014), they train 17B samples with 16M features on 1K nodes by 10 passes $\Rightarrow$ 70 minutes

Sibyl from Google

- Based on the algorithm in Collins et al. (2002)
- Idea:

$$f(\boldsymbol{w}) \approx A(\boldsymbol{w}) = \sum_{i=1}^{n} A_i(w_i)$$

Minimize

$$A_i(w_i), \forall i$$

in parallel

Outline

Example of Distributed Machine Learning

- Computational advertising (in particular, click-through rate prediction) is an area that heavily uses distributed linear classification

  We will explain what CTR prediction is
- See also applications mentioned in Google's Sibyl talk

Example: CTR Prediction

- Definition of CTR:

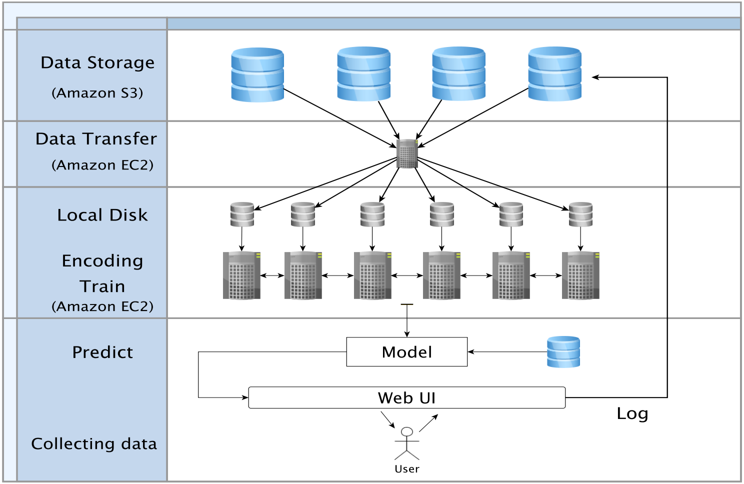
$$\text{CTR} = \frac{\#\ \text{clicks}}{\#\ \text{impressions}}.$$

- A sequence of events

Not clicked	Features of user
Clicked	Features of user
Not clicked	Features of user
...	...

- A binary classification problem.

Example: CTR Prediction (Cont'd)



Outline

Design Considerations for Distributed Training

- On one computer, often we do batch rather than online learning

  Online and streaming learning may be more useful for big-data applications
- The example (Newton method) we showed is a synchronous parallel algorithms

  Maybe asynchronous ones are better for big data?

System Issues

- Platforms are still being actively developed. We must check technical details
- Example: in developing distributed Newton methods on Spark (an in-memory cluster-computing platform), we must consider details of
  - Spark
  - Scala
- For example, you want to know
  - the difference between **mapPartitions** and **map** in Spark, and
  - the slower **for** loop than **while** loop in Scala

Workflow Issues

- Data analytics is often only part of the workflow of a big-data application
- By workflow, I mean things from raw data to final use of the results
- Other steps (e.g., feature generation or data movements) may be more complicated or more time consuming than the analytics step

Open-source Developments

- Open-source developments are very important for big data analytics
- How it works:

  The company must do an application X. They consider an open-source tool Y. But Y is not enough for X. Then their engineers improve Y and submit pull requests
- Through this process, core developers of a project are formed. They are from various companies

Conclusions

- Big-data machine learning is in its infancy
- It's challenging to development algorithms and tools in a distributed environment
- To start, we should take algorithms, systems, and applications together into consideration
- Hopefully we will get some breakthroughs in the near future