Drs. Sha & Liu
{feisha,yanliu.cs}@usc.edu

November 6, 2014
Outline

1. Administrative matters
2. Big Data: Motivating Examples
3. Technique 1: Parallel Algorithms
4. Technique 2: Locality-sensitive Hashing
5. Technique 3: Distributed Optimization
6. Tips for Mini-project
Survey Results

Lectures and Discussion Forum

How do you evaluate the quality of lectures?

Answered: 68  Skipped: 1

How do you evaluate the quality of TA activity in the discussion forum?

Answered: 68  Skipped: 1
Survey Results

Homework

How do you evaluate the quality of algorithmic problems in the assignments?

- Extremely satisfied: [ ]
- Somewhat satisfied: [ ]
- No opinion: [ ]
- Somewhat dissatisfied: [ ]
- Extremely dissatisfied: [ ]

How do you evaluate the quality of programming problems in the assignments?

- Extremely satisfied: [ ]
- Somewhat satisfied: [ ]
- No opinion: [ ]
- Somewhat dissatisfied: [ ]
- Extremely dissatisfied: [ ]
Survey Results

TA and Workload

How do you evaluate the quality of TA office hours?

- Extremely satisfied
- Somewhat satisfied
- No opinion
- Somewhat dissatisfied
- Extremely dissatisfied

Answered: 68  Skipped: 1

How do you evaluate the overall workload of the assignments?

- Light
- Appropriate
- Heavy

Answered: 68  Skipped: 1
Survey Results

Comments to Instructors
1. Less math; do derivation on blackboard
2. A lot of typos on slides
3. Slow down
4. Homework synchronization issues

Comments to TAs
1. Larger room (addressed)
2. More recitation sessions
3. Some communication issues because of dividing homeworks among two sets of TAs.
4. Homeworks are time-consuming
Outline

1. Administrative matters

2. Big Data: Motivating Examples

3. Technique 1: Parallel Algorithms

4. Technique 2: Locality-sensitive Hashing

5. Technique 3: Distributed Optimization

6. Tips for Mini-project
Motivating Examples

(a) Computational Advertising

(b) Recommendation System

(c) Crowd sourcing

(d) Scientific discovery
Summary of Techniques

A variety of techniques have been developed: some have been discussed and others to be discussed in this class

- Parallel algorithms: will be discussed
- Online algorithms: e.g., perception
- Stochastic algorithms: e.g., stochastic gradient descent
- Locality sensitive hashing: will be discussed
- Distributed optimization: will be discussed
- Greedy algorithms: e.g. greedy search algorithm for solving Lasso
- First-order algorithms: e.g., NESTA
Outline

1. Administrative matters
2. Big Data: Motivating Examples
3. Technique 1: Parallel Algorithms
4. Technique 2: Locality-sensitive Hashing
5. Technique 3: Distributed Optimization
6. Tips for Mini-project
Parallel Algorithms

We are at the beginning of the multicore era (Moore’s law)

[Sutter and Larus, 2005] points out that multicore mostly benefits concurrent applications

- Best match if the data can be subdivided and stay local to the cores:

\[
\sum f(x_i) = f(x_1) + f(x_2) + \ldots + f(x_n).
\]

- Permit the learning algorithm to access the learning problem only through a statistical query oracle. E.g. given a function of \( f(x, y) \), it returns an estimate of the expectation of \( f(x, y) \).
Parallel Algorithm Examples: Linear Regression

Remember in linear regression, we have the following solution of \( y = \theta^T x \) as

\[
\theta^* = (X^T X)^{-1} X^T \vec{y}
\]

\[
\begin{align*}
A &= X^T X \\
b &= X^T \vec{y} \\
\theta^* &= (A)^{-1} b
\end{align*}
\]

\[
\begin{align*}
A &= \sum_{i=1}^{m} (x_i x_i^T) \\
b &= \sum_{i=1}^{m} (x_i y_i) \\
\theta^* &= (A)^{-1} b
\end{align*}
\]
Parallel Algorithm Examples: Naive Bayes

Remember in Naive Bayes, we need to estimate $P(x_j = k | y = 1)$ and $P(y)$ from the training data. In other words, we need to calculate the following terms:

$$
\sum_{\text{subgroup}} 1\{x_j = k | y = 1\}, \quad \sum_{\text{subgroup}} 1\{x_j = k | y = 0\}, \quad \sum_{\text{subgroup}} 1\{y = 1\}, \quad \sum_{\text{subgroup}} 1\{y = 0\}
$$
Outline

1. Administrative matters
2. Big Data: Motivating Examples
3. Technique 1: Parallel Algorithms
4. Technique 2: Locality-sensitive Hashing
5. Technique 3: Distributed Optimization
6. Tips for Mini-project
Locality-sensitive hashing (LSH): Introduction

Difficulties faced by machine learning applications, where exist
- High dimensional data feature (e.g. vocabulary token histogram)
- Large size of training data (e.g. personal emails of thousands of users)

Reduce the feature dimension effectively:
- From the original features $x \rightarrow R^d$
  To the transformed features $\phi(x) \rightarrow R^m$, where $m << d$ through function $\phi(x)$
- With little information loss:
  preserve norm $\|x\|_\phi = \|x\|_2$, and
  preserve inner product $\langle \phi(x), \phi(y) \rangle = \langle x, y \rangle$. 
Locality-sensitive hashing (LSH)

Given a collection of objects $C$ and similarity function $sim(x, y)$ that maps pairs of objects $x, y \in C$, a locality sensitive hash function family $F$ operates on $C$, such that for any $x, y \in C$,

$$Prob_{h \in F}[h(x) = h(y)] = sim(x, y).$$

Below are two examples of locality sensitive hash function family:

1. Min-wise independent permutations for Jaccard coefficients:
   $$Prob_{h \in F}[\min\{\pi(x)\} = \min\{\pi(y)\}] = \frac{|x \cap y|}{|x \cup y|}.$$

2. Random projection for cosine similarity:
   $$Prob_{h \in F}[h(x) = h(y)] = 1 - \frac{\theta(x, y)}{\pi},$$
   where $\theta(x, y)$ is the angle between vector $x$ and $y$. 

Drs. Sha & Liu  
{feisha,yanliu.cs}@usc.edu  
CSCI567 Machine Learning (Fall 2014)  
November 6, 2014 16 / 25
A k-shingle (or k-gram) is a sequence of k tokens that appears in the document.

Represent a doc by the set of hash values of its k-shingles.

A natural document similarity measure is then the Jaccard similarity:

\[ \text{sim}(x, y) = \frac{|x \cap y|}{|x \cup y|}. \]
The random projection method of LSH is designed to approximate the cosine distance between vectors.

The basic idea of this technique is to choose a random hyperplane (defined by a normal unit vector $v$) at the outset and use the hyperplane to hash input vectors.

Given an input vector $x$ and a hyperplane defined by $r$, we let $h(x) = \text{sign}(v \cdot r)$. That is, $h(x) = +/ - 1$ depending on which side of the hyperplane lies.
Outline

1 Administrative matters
2 Big Data: Motivating Examples
3 Technique 1: Parallel Algorithms
4 Technique 2: Locality-sensitive Hashing
5 Technique 3: Distributed Optimization
6 Tips for Mini-project
Distributed Optimization

1. Massive data can be stored or even collected in distributed manner.
   - Example 1: cloud computing, Hadoop/MapReduce
   - Example 2: training data of extremely large size.

2. Increasing computing resources, parallel computation.
   - e.g., features with high dimensions.

Distributed optimization for large scale problem (Dean et al. 2012)
Distributed SGD - One shot averaging

Settings:

- Divide the data into $K$ disjoint sets such that $S = (s_1, ..., s_K)$.
- Suppose we have $K$ machines.

Algorithm:

```python
for each machine $i \in \{1, ..., K\}$ parallel do:
    compute $\hat{w}_i = \text{SGD}(s_i, T, \eta, w_0)$

return $\frac{1}{K} \sum_{i=1}^{K} \hat{w}_i$
```

where $T$ is the maximum number of iterations, $\eta$ is the step size, and $w_0$ is the initial model parameter.

*This method is simple, and no communication until the end!*
Sometimes, it is impossible to store all the parameters in single machine. i.e. for social network with billions of edges, the size of model parameters can go up to few thousand terabyte.

Communication between clients and servers. Clients process data and while servers synchronize parameters and perform global updates.
Tips for Mini-project

Outline

1. Administrative matters
2. Big Data: Motivating Examples
3. Technique 1: Parallel Algorithms
4. Technique 2: Locality-sensitive Hashing
5. Technique 3: Distributed Optimization
6. Tips for Mini-project
**Tips**

- Start early! The data is fairly large and you will need some time to be able to handle it.
- Start with simpler algorithms like linear algorithms so that you make at least a submission.
- Make sure that you use online algorithms like stochastic gradient descent if you cannot load the data into memory.
- Always use all of the available cores in your system via the multi-threading packages. (e.g. OpenMP in C++)
- If you don’t have enough time, tune your algorithms in a small amount of data, then run the selected algorithm on the entire data.
Resources

**Scikit-Learn**  http://scikit-learn.org/
Easy to use, general purpose Python library for machine learning. Good for large scale learning. It has many of the important algorithms implemented.

**Vowpal-Wabbit**  http://hunch.net/~vw/
Very fast C++ library for most of the fundamental algorithms.

**Shogun**  http://www.shogun-toolbox.org/
Written in C++, with interface for many languages. Mainly for large-scale learning.

**Graphlab**  http://graphlab.com
Very fast implementation of graphical models and graph-based algorithms.

**ParameterServer**  http://parameterserver.org
Distributed machine learning algorithms

And many others. See a comparison of features in here:
http://www.shogun-toolbox.org/page/features/.