Parallelizing Latent Dirichlet Allocation (LDA)

Virgile Audi (vaudi@g.harvard.edu)
Nicolas Drizard (nicolasdrizard@g.harvard.edu)
Charles Liu (cliu02@g.harvard.edu)

November 4, 2015

1 Introduction

1.1 Background

Latent Dirichlet Allocation was a topic model proposed in 2003 that allows sets of observations to be explained by topics automatically discovered by the LDA technique. As stated in the article:

> The goal is to find short descriptions of the members of a collection that enable efficient processing of large collections while preserving the essential statistical relationships that are useful for basic tasks such as classification, novelty detection, summarization, and similarity and relevance judgments.

The technique has widely been used in natural language processing.

1.2 Motivation

There is currently no public version in python that is parallelized (or in Cython). We think such functionality would be useful to the community, as LDA is a conceptually simple algorithm but provides impressive results.

2 Objectives

2.1 Functionality

The algorithm is well-defined, we are just trying to parallelize it. We'd like to provide a Cython implementation as well, and apply more novel parallelization techniques such as Hogwild's lock-free approach to gradient descent.

2.2 Performance

We will benchmark our algorithm against two existing libraries:

- Serial Python library: `https://pypi.python.org/pypi/lda`
- Parallel C++ library: `https://code.google.com/p/plda/`

3 Design Overview

3.1 Technologies Used

As this is somewhat an embarassingly parallel problem, we want to look at two approaches:

- Python multiprocessing
- Cython + Python multithreading

3.2 Parallelism Details

There are three areas where this can be parallelized:

- Gradient descent using the Hogwild approach
- Variational inference
- Gibbs sampler

4 Verification

As there are serial implementations available we will compare with those over Yelp and Guttenberg data (and larger data sets we have yet to find).

5 Schedule

- Functional, serial LDA implementation
- Parallelized gradient descent using Hogwild
- Parallelized update procedure (Gibbs sampling and variational inference)

6 Work Distribution

Each member will write a serial LDA to understand the algorithm and compare performance. Since there are three parallelization parts, each member will focus on one.