Parallel PLSI on Spark

Hongwei Wang, Bicheng Zhang

Department of Computer Science
University of Illinois at Urbana-Champaign
{hwang172, zhang368}@illinois.edu

1 Introduction

Probabilistic Latent Semantic Indexing (PLSI) [5] is a category of topic models, which aims to find hidden topics in a corpus of documents. It is widely used in information retrieval, natural language processing and machine learning. As the number of documents increase, it is necessary to parallelize the computation in multiple nodes, and a popular way of doing it is MapReduce [4], a distributed framework proposed by Google and open sourced by Apache Hadoop [1].

However, the MapReduce implementation in Hadoop is not that efficient for PLSI computation. The reason is that PLSI is an iterative process, and the output of a Reduce phase will become the input of next Map phase, while in Hadoop, it always writes the output of Map and Reduce to the disk, which incurs a large overhead between each iteration.

Recently, a new distributed framework called Spark [2], was proposed to solve the problem of iterative MapReduce jobs by introducing a new distributed memory abstraction called resilient distributed datasets (RDD) [6], which supports in-memory computation and efficiently data reuse. It maintains a fault tolerant memory cache on each node so that the output of MapReduce can keep in memory for next iteration.

Since currently Spark does not support PLSI [3], our contribution is to implement PLSI on Spark.

2 PLSI

PLSI models the the probability of each observed document and word pairs \((d, w)\) as a mixture of conditional independent multinomial distribution with each topic as a hidden variable. PLSI assumes the process of generating a pair of \((d, w)\) as follows:

- Choose a document with the probability of \(P(d)\).
- Given the document \(d\), choose a topic \(z\) with probability \(P(z|d)\).
- Given the topic \(z\), choose a word \(w\) with probability \(P(w|z)\).

So the joint probability of a \((d, w)\) pair is

\[
P(d, w) = P(d)P(w|d) = P(d) \sum_z P(w|z)P(z|d)
\]
The joint distribution for the whole corpus is

$$
\prod_{d,w} P(d,w) = \prod_{d,w} P(d)P(w|d) = \prod_{d,w} P(d) \sum_z P(w|z)P(z|d)
$$

We are interested in estimating the parameters $P(z|d)$ and $P(w|z)$ given the corpus. $P(z|d)$ represents the topic distribution in a document, and $P(w|z)$ represents the word distribution within a topic.

Let $\theta$ denotes all the parameters, and the log maximum likelihood is

$$
\log L(\theta) = \sum_{d,w} n(d,w)\log P(d,w; \theta)
= \sum_{d,w} n(d,w)\log P(w|d; \theta)P(d)
= \sum_{d,w} n(d,w)\log P(w|d; \theta) + \sum_{d,w} n(d,w)\log P(d)
= \sum_{d,w} n(d,w)\log \sum_z P(w|z)P(z|d) + \sum_{d,w} n(d,w)\log P(d)
$$

Since

$$
\sum_{d,w} n(d,w)\log P(d)
$$

does not contain $\theta$, we only need to find

$$
\theta_{ML} = \arg \max_\theta \sum_{d,w} n(d,w)\log \sum_z P(w|z)P(z|d)
$$

Here we use expectation-maximization (EM) algorithm. The iterative procedure is defined as $\theta_t = \arg \max_\theta Q(\theta, \theta_{t-1})$, where

$$
Q(\theta, \theta_{t-1}) = \sum_{d,w} n(d,w)\sum_z P(z|d,w; \theta_{t-1}) \log P(w|d; \theta)
= \sum_{d,w} n(d,w)\sum_z P(z|d,w; \theta_{t-1})[\log P(w|z) + \log P(z|d)]
$$

Fig. 1. PLSI plate notation
**E step:**

\[
P(z|d, w; \theta_{t-1}) = \frac{P_{t-1}(z|d)P_{t-1}(w|z)}{\sum_{z'} P_{t-1}(z'|d)P_{t-1}(w|z')}
\]

We can compute \( Q(\theta, \theta_{t-1}) \) with the above \( P(z|d, w; \theta_{t-1}) \).

**M step:**

We need to compute

\[
\theta_t = \arg \max_{\theta} Q(\theta, \theta_{t-1})
\]

where \( \theta \) denotes the parameters \( P(w|z) \) and \( P(z|d) \) for each \( w, z, d \) with the constraint of

\[
\sum_w P(w|z) = 1 \quad \text{(1)}
\]

\[
\sum_z P(z|d) = 1 \quad \text{(2)}
\]

To solve the optimization problem, we construct the Lagrange function by introducing Lagrange multipliers.

\[
H = Q(\theta, \theta_{t-1}) + \sum_z \alpha_z (1 - \sum_w P(w|z)) + \sum_d \beta_d (1 - \sum_z P(z|d))
\]

We have

\[
\frac{\partial H}{\partial P(w|z)} = \frac{\sum_d n(d, w)P(z|d, w; \theta_{t-1})}{P(w|z)} - \alpha_z
\]

\[
\frac{\partial H}{\partial P(z|d)} = \frac{\sum_w n(d, w)P(z|d, w; \theta_{t-1})}{P(z|d)} - \beta_d
\]

Set

\[
\frac{\partial H}{\partial P(w|z)} = 0
\]

\[
\frac{\partial H}{\partial P(z|d)} = 0
\]

i.e.

\[
\sum_d n(d, w)P(z|d, w; \theta_{t-1}) - \alpha_z = 0 \quad \text{(3)}
\]

\[
\sum_w n(d, w)P(z|d, w; \theta_{t-1}) - \beta_d = 0 \quad \text{(4)}
\]

With (1)(2)(3)(4), we can compute

\[
P(w|z) = \frac{\sum_d n(d, w)P(z|d, w; \theta_{t-1})}{\sum_d n(d, w')P(z|d, w'; \theta_{t-1})}
\]

\[
P(z|d) = \frac{\sum_w n(d, w)P(z|d, w; \theta_{t-1})}{\sum_w n(d, w)P(z'|d, w; \theta_{t-1})}
\]
3 Progress

We have completed the formula derivation of PLSI in Section 2, and implemented a local version of PLSI with Python.

4 Experiment

In our experiment, we would like to set up Spark on 5 instances in Amazon EC2, and evaluate our parallel PLSI algorithm on a sample of Wikipedia data.

5 Difficulties and Future Work

The difficulty of this project is how to parallelize and distribute the computation in MapReduce jobs and implement them on Apache Spark. Based on our current progress, we think we can complete it on time.

References