機械学習アルゴリズムの分散グラフ処理フレームワークによる記述

1311216　Rathore Amogh　　岩崎研究室

1　背景

グラフはとてもフレキシブルな表現である。いろいろな分野のデータがグラフで表現できて、様々な問題が形式化できる。今はビッグデータの時代で、量が大きいデータをグラフで表現すると非常にも大きいグラフになることが普段である。そのような大きいグラフは一つの計算機で処理するのは無理である。なぜなら、一つの計算機はメモリーが限られているし、プロセサが一個なので、処理時間も長いからである。だから、グラフを分散して処理する必要がある。

分散グラフ処理はフレームワークがいくつか知られている。例えば、Apache Giraph や Pregel+ などのフレームワークが一番有名である。

近頃は、ビッグデータと同様に機械学習の分野も広がれてきた。機械学習は人間が自然に行っている学習と同様の機能をコンピュータで実現する技術のことである。基本的には、機械学習がデータにパターンを見つけてデータについて学習を行うことである。ここもデータが大きいと処理時間が長くなるので、並列的に処理を行う必要がある。

2　Vertex-centric approach によるグラフ処理

Apache Giraph や Pregel+ などは Vertex-centric approach でグラフを処理するフレームワークである。Vertex-centric approach は各頂点が同じ関数を繰り替えして実行して、それでグラフ全体が処理されるアプローチのことである。Giraph と Pregel+ はスーパーステップという概念がある。一つのスーパーステップでグラフのすべての頂点がある同じ関数を一回実行して必要があったら他の頂点へメッセージを送る。受信側の頂点はそのメッセージを次のスーパーステップで受信する。

3　目的と方針

本研究では、機械学習のアルゴリズムを分散グラフ処理フレームワークで実装し、ライブラリ化を行う。さらに、性能の評価を行う。

まずは、グラフで表現できるアルゴリズムを幾つか探して、アルゴリズムの勉強を行う。次に、アルゴリズムで使うデータを集める。それから、分散グラフフレームワークの Giraph での設計を考える。その設計に基づいて、アルゴリズムを実装して、実行してみる。次に、実装の一般化の設計を考えて、一般的な実装に変換する。そして、実行の性能を評価する。この方針で、様々のアルゴリズムを実装し、ライブラリ化を行って、性能を評価する。

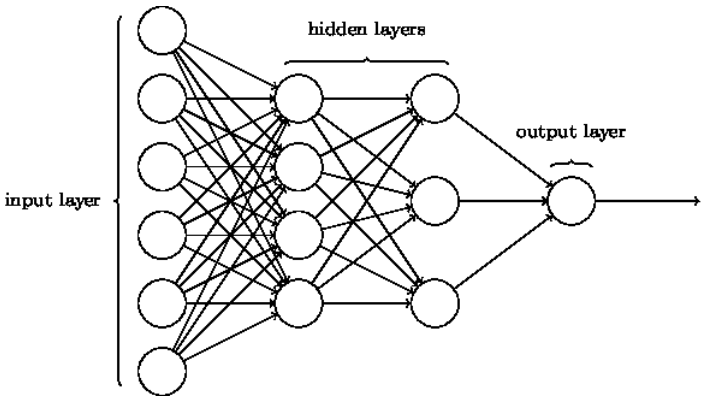

図 1　ニューラルネットワーク

4　対象とする機械学習アルゴリズム

実は機械学習には、アルゴリズムは 2 種類あって、それは教師あり学習と教師なし学習のことである。本研究には、教師ありとを教師なしを気にせずに、グラフで表現できるかどうかだけを気にして、アルゴリズムを選ぶことにする。今の段階では以下のアルゴリズムを対象とする。

- 人工ニューラルネットワーク（バックワードプロパゲーションアルゴリズム）
- K 平均法クラスタリング
- 自己組織化写像

5　人工ニューラルネットワーク

人工ニューラルネットワークは、脳機能にみられるいくつかの特性を計算機上のシミュレーションによって表現することを目指した数学モデルである。図 1から分かるように、ニューラルネットワークはレイヤーで分かれている。各レイヤーはニューロンがあって、ニューロンは次のレイヤーに辺を出している。一番目のレイヤーは入力レイヤーと呼ばれて、最後のレイヤーは出力レイヤーである。各ニューロンはアクティベートされるとき、何か計算をして、自分のアクティベーション値を求める。

6　バックワードプロパゲーションアルゴリズム

教師あり機械学習はコスト関数という関数があって、それについてパラメータの最適化を行うことがよくある。バックワードプロパゲーションはニューラルネットワークでその最適化を行うアルゴリズムの一つである。このアルゴリズムは 2 段階で分かれている。1 段階はフォーワードプロパゲーションのことである。2 段階はバックワードプロパゲーションである。

図 2から分かるように、各ニューロンと辺のウエイト

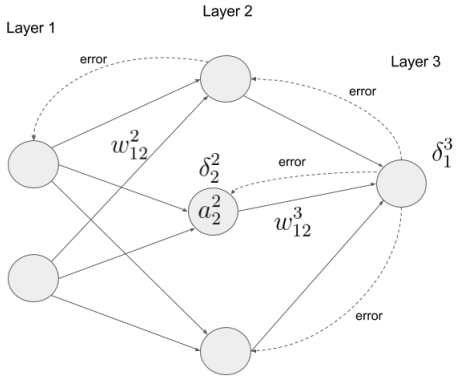

図 2　フォーワードプロパゲーション

(パラメータ) を w_{jk}^l と a_j^l で表すことがわかる。w_{jk}^l はレイヤー $l-1$ のニューロン k からレイヤー l のニューロン j に行く辺のウエート（パラメータ）の値である。a_j^l はレイヤー l のニューロン j のアクティベーションである。

フォーワードプロパゲーションでは、各レイヤーのアクティベーションを、$a_j^l = S(\sum_k (w_{jk}^l a_k^{l-1}))$ で計算する。ここで S はシグモイド関数である。入力レイヤーのアクティベーションは入力になるので、計算する必要がない。レイヤー 2 から出力レイヤーまですべてのレイヤーのアクティベーションを計算する。

フォーワードプロパゲーションが終わったら、バックワードプロパゲーションで各ニューロンに付属されてるエラーを計算する。出力レイヤーのエラーは単にトレーニングデータの実際の出力マイナスニューラルネットワークが計算した出力結果の値とする。それで、出力レイヤーは自分のエラーを一個前のレイヤーにバックプロパゲート (渡す) する。前のレイヤーの k ニューロンのエラーを $\delta_k^l = \sum_j (w_{jk}^{l+1} \delta_j^{l+1})$ x $S'(\sum_k (w_{jk}^l a_k^{l-1}))$ で計算する。

フォーワードとバックワードプロパゲーションが終わったら、Gradient Descent などの最適化アルゴリズムによって、辺のウエート (パラメータ) の値を更新して、また上のプロセスを繰り返す。

7　Giraph での実装の設計

Giraph でバックワードプロパゲーションを実装するために、ニューロンを頂点、辺をグラフの辺にする。スーパーステップ 1 でニュー力レイヤーの各ニューロンがはレイヤー 2 のニューロンに自分のアクティベーション値を送る。スーパーステップ 2 では、レイヤー 2 のニューロンは自分のアクティベーションを計算して、結果をレイヤー 3 に渡す。これを繰り返して、出力レイヤーまでのアクティベーションをすべて計算する。このフォーワードプロパゲーションには L スーパステップがかかる (L はレイヤー数)。それから、前の節で述べた通り、各レイヤーはスーパステップ L からスーパーステップ $2L-1$ でエラーをバックプロパゲートする。

これを一つのネットワークで行うと、M 個トレーニングデータに対して M 回同じ処理を行う必要がある。Giraph は並列処理フレームワークなので、M 個データに対して M 個のニューラルネットワークを作って、並列的にパラメータの最適化を行う。ここで、一個条件があって、それはすべてのニューラルネットワークの処理が終わるまでには、どのネットワークも先に次繰り返しを行うのはできないことである。なぜなら、すべてのネットワークはウエートの値をシェアーするからである。一回の繰り返しが終わったら、ウエートを更新するので、すべてのネットワークの処理結果によってウエートを更新して、次の繰り返しを行う。

8　まとめ

上の節で述べた通り、機械学習の幾つかのアルゴリズムを分散グラフ処理フレームワークで実装して、アルゴリズムの実装とフレームワークの性能を評価する。さらに、実装のライブラリ化を行うことが本研究の目的である。対象とするフレームワークはまず Giraph で、時間があったら Pregel+ も含むようにする。

9　現状と今後の予定

機械学習の基本的なことを勉強を行った。得に、ニューラルネットワークについて調べて、バックワードプロパゲーションアルゴリズムの内容を理解した。そして、教師なし学習アルゴリズムの K 平均法アルゴリズムの勉強を行った。さらに、分散グラフ処理フレームワークの Giraph と Pregel+ の構造を調べて、Giraph で最短経路問題を解く Dijkstra のアルゴリズムを実装した。そして、ニューラルネットワークとバックワードプロパゲーションアルゴリズムの Giraph で実装する設計を少し考えた。

これから、バックワードプロパゲーションの設計をもっと具体的に考える。それから、バックワードプロパゲーションで解けられる機械学習問題を探して、それに対してニューラルネットワークを Giraph で実装する。次に、実装を一般化するための設計を考えて、実装の一般化を行う。同じことを他のアルゴリズムに対して繰り返す。

参考文献

[1] Apache Giraph: http://giraph.apache.org/
[2] Pregel Paper: G. Malewicz et al., *Pregel: A System for Large-Scale Graph Processing*, SIGMOD '10, ACM, 2010, pp. 135-146
[3] Pregel+: http://www.cse.cuhk.edu.hk/pregelplus/download.html
[4] A Basic Introduction To Neural Networks http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html
[5] Matt Mazur, *A Step by Step Backpropagation Example* https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/