UNIVERSITÀ DEL SALENTO

Facoltà di Ingegneria

Corso di Laurea Magistrale in Computer Engineering

Advanced Control Techniques Project

Multinomial Logistic Regression for a Supervised Learning problem

Professor: **Giuseppe Notarstefano**

Students: **Alfarano Gianluca,**
Basile Davide,
Capoccia Leonardo,
Larini Ludovico

Academic year 2017/2018

Abstract

In this report, it will be shown how to solve a Multinomial Logistic Regression (also known as *Softmax Regression*) using a distributed method. The sub-gradient method has been used to distribute calculations among a configurable number of agents. A portion of the dataset is given to each agent and used to minimize a cost function related to the portion of the dataset in its possession.

Contents

List of Figures

Introduction

In the past there was a single *Mainframe* that executed all digital operations. After, it was born the *Personal Computer* and more people could execute the same operations in private. Today exist the *Microcontrollers* that permit to make smart an infinity of devices. More algorithms have been created to connect these devices for distribute the computation.
This work borns to implement a scenario in which there are some agents that estimate a cost function using their own information and those of the other agents; they use a *Distribuited Sub-gradient Method* to update their own estimate and, in particular, they resolve a *Multinomial Logistic Regression*.
After some test in *MATLAB*, it is used *MPI* implemented with *Python*.
The present work is divided into two chapters. In the Chapter 1 it is introduced the theory behind the problem and it is visualized and commented the implementation code. In the Chapter 2 there are the results of simulations with some considerations.

Chapter 1

Chapter 1 Problem and its implementation

First-chapter for problem set-up and description of the implemented solution.

1.1 Theory of the problem

1.1.1 Distributed Subgradient Methods for Multi-Agent Optimization

In this problem there are m agents that cooperatively minimize a common additive cost. The optimization general problem is:

$$\textit{minimize} \quad \sum_{i=1}^{m} f_i\left(x\right) \qquad \textit{subject to} \quad x \in \mathbb{R}^n, \tag{1.1}$$

where $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}$ is the cost function of agent i, known by this agent only, and $x \in \mathbb{R}$ is a decision vector. It is assumes that:

- The cost function is convex;
- the agents are distributed over a time-varying topology;
- the graph (V, E_∞) is connected, where E_∞ is the set of edges (j, i) representing agent pairs communicating directly infinitely many times;
- there isn't communication delay.

Every agent i generates and maintains estimates of the optimal decision vector based on information concerning its own cost function and exchanges this estimate with its directly neighbors at discrete times $t_0, t_1, t_2, ...$ Moreover, each agent i has a vector of weights $a^i(k) \in \mathbb{R}^m$ at any time t_k; for each time, the scalar $a_i^j(k)$ is zero if the agent i doesn't directly comunicate with j, else it is the weight assigned from the agent i to the information x^j obtained from j during the time interval (t_k, t_{k+1}). The estimates are updated according to the update rule: [?]

$$x^i\left(k+1\right) = \sum_{j=1}^{m} a_j^i\left(k\right) x^i\left(k\right) - a^i\left(k\right) d_i\left(k\right) \tag{1.2}$$

where $\alpha^i(k) > 0$ is the (diminishing) stepsize used by agent i and the vector $d_i(k)$ is a subgradient of agent i objective function $f_i(x)$ at $x = x^i(k)$.

1.1.2 Multinomial Logistic Regression

$$f_i\left(\omega\right) := \left|\left|h_\omega\left(x^{(i)}\right) - y^{(i)}\right|\right|^2 \tag{1.3}$$

$$f\left(\omega\right) := \sum_{i=1}^{N} f_i\left(\omega\right) \tag{1.4}$$

$$\omega^* := \arg\min_{\omega} f\left(\omega\right) \tag{1.5}$$

$$h_\theta = \frac{1}{\sum_{j=1}^{K} exp\left(\theta^{(j)\top} x\right)} \begin{bmatrix} exp\left(\theta^{(1)\top} x\right) \\ \vdots \\ exp\left(\theta^{(K)\top} x\right) \end{bmatrix} \tag{1.6}$$

$$\theta^* = \arg\min_{\theta} - \sum_{i=1}^{N} g_i(\theta) \tag{1.7}$$

$$g_i\left(\theta\right) := \sum_{k=1}^{K} 1\{y^{(i)} = e_k\} \log\left(\frac{exp(\theta^{(k)\top} x^{(i)})}{\sum_{k=1}^{K} exp(\theta^{(j)\top} x)}\right) \tag{1.8}$$

1.1.3 Pseudocode

1.2 Code Implementation

Algorithm 1

Stop Rules:
$||x_{k+1} - x_k|| \leq \varepsilon \qquad \varepsilon$ fixed
Number of maximum iterations reached
Start:
Fix initial conditions for each node $x_i(0) = [0 \quad ... \quad 0]^T$
Define the Adjancency Matrix, Weights Matrix, $\alpha^i = \alpha$ constant for each iteration
while No stop rule is true, each node i does: **do**
    calculate ∇f_i
    for each neighbor j **do**
        $x_i(k+1) = x_i(k+1) + a^i_j(k)x^j(k)$
    end for
    $x_i(k+1) = x_i(k+1) - \alpha\nabla f_i$
end while
Result:
Each node i should converge to x^*
The minimum of function is $\sum_{i=1}^{m} f_i(x^*)$

Chapter 2

Chapter 2 Results of simulations

Second chapter for description of the results (simulations and experiments where applicable).

Citation [?]

Conclusions

In this work it has been resolved a Multinomial Logistic Regression problem using MPI in Python. Each agent used a Distributed Sub-gradient method to update its own estimate of optimal solution.

Bibliography

[1] Mario Rossi and Giulio Bianchi. Just an example. *La Repubblica*, 2011.