Master Thesis

Distributed Deep Learning

Joeri R. Hermans

Thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science of Artificial Intelligence

at

Maastricht University
Faculty of Humanities and Sciences
Department of Data Science & Knowledge Engineering
Maastricht, The Netherlands

Preface

This thesis is submitted as a final requirement for the Master of Science degree at the Department of Data Science & Knowledge Engineering of Maastricht University, The Netherlands. The subject of study originally started as a pilot project with Jean-Roch Vlimant, Maurizio Pierini, and Federico Presutti of the EP-UCM group (CMS experiment) at CERN. In order to handle the increased data rates of LHC Run 3 and High Luminosity LHC, the CMS experiment is considering to construct a new architecture for the High Level Trigger based on Deep Neural Networks. However, they would like to significantly decrease the training time of the models as well. This would allow them to tune the neural networks more frequently. As a result, we started to experiment with various state of the art distributed optimization algorithms. Which resulted in the achievements and insights presented in this thesis.

I would like to express my gratitude to several people. First and foremost, I would like to thank my promotor [redacted] for his expertise and suggestions during my research, which drastically improved the quality of this thesis. Furthermore, I would also like to thank my friends, colleagues and scientists at CERN for their support, feedback, and exchange of ideas during my stay there. It was a very motivating and inspiring time in my life. Especially the support and experience of my CERN supervisors, Zbigniew Baranowski, and Luca Canali, was proven to be invaluable on multiple occasions. I would also like to thank them for giving me the personal freedom to conduct my own research. Finally, I would like to thank my parents and grandparents who always supported me, and who gave me the chance to explore the world in this unique way.

Joeri R. Hermans
Geneva, Switzerland 2016 - 2017

Abstract

Abstract here.

Summary

Summary here.

Contents

Abbreviations and Notation

ASGD	Asynchronous Stochastic Gradient Descent
CERN	European Organization for Nuclear Research
CMS	Compact Muon Solenoid
EASGD	Elastic Averaging Stochastic Gradient Descent
HL-LHC	High Luminosity Large Hadron Collider
LHC	Large Hadron Collider
SGD	Stochastic Gradient Descent

Chapter 1

Introduction

In this chapter we introduce Distributed Deep Learning and the problems surrounding it. A more detailed description of the subject of study is given in Chapter 2. Furthermore, we make the reader more comfortable with the notation and abbreviations used throughout this thesis. Finally, we formally define the problem statement in Section 1.3, and give an outline of this thesis in Section 1.4.

1.1 Distributed Deep Learning, an introduction

Unsupervised feature learning and deep learning has shown that being able to train large models can drastically improve model performance. However, consider the problem of training a deep network with millions, or even billions of parameters. How do we achieve this without waiting for days, or even multiple weeks? Dean et al. propose a different training paradigm which allows us to train and serve a model on multiple physical machines [1]. The authors propose two novel methodologies to accomplish this. Namely, *model parallelism*, introduced in Section 1.1.1, and *data parallelism*, introduced in Section 1.1.2.

In this thesis we study *data parallelism*, since this methodology mainly focuses on the development of distributed optimization algorithms. Whereas, *model parallelism* is mainly an engineering effort, because it still follows the traditional optimization scheme, i.e., sequential gradient updates in the case the applied optimizer utilizes a gradient based approach.

1.1.1 Model Parallelism

In *model parallelism*, a single model is distributed over multiple machines [1]. The performance benefits of distributing a deep network across multiple machines mainly depends on the structure of the model. Models with a large number of parameters typically benefit from access to more CPUs and memory, up to the point where communication costs, i.e., propagation of weight updates and synchronization mechanisms, dominate [1].

Let us start with a simple example in order to illustrate this concept more clearly. Imagine having a perceptron, as depicted in Figure 1.1. In order to parallelize this efficiently, we can view a neural network as a dependency graph, where the goal is to minimize the number of synchronization mechanisms, assuming we have unlimited resources. Furthermore, a synchronization mechanism is only required when a node has more than 1 *variable* dependencies. A variable dependency is a dependency which can change in time. For example, a bias would be a *static* dependency, because the value of a bias remains constant over time. In the case for the perceptron shown in Figure 1.1, the parallelization is quite straightforward. The only synchronization mechanism which should be implemented resides in output neuron because $y \triangleq \sum_i w_i x_i$.

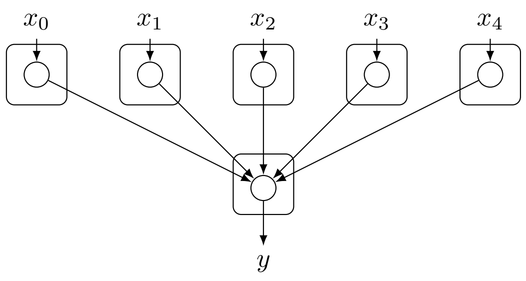

Figure 1.1: A perceptron partitioned using the *model parallelism* paradigm. In this approach every input node is responsible for accepting the input x_i from some source, and multiplying the input with the associated weight w_i. After the multiplication, the result is sent to the node which is responsible for computing y. Of course, this node requires a synchronization mechanism to ensure that the result is consistent. The synchronization mechanism does this by waiting for the results y depends on.

1.1.2 Data Parallelism

In this thesis, we focus our efforts on data parallelism. Data parallelism is an inherently different methodology of optimizing parameters. The general idea is to reduce the training time by having n different workers optimizing a model by processing n different shards (partition) of the dataset in parallel [1]. In this setting we distribute n model replicas over n processing nodes, i.e., every node (or process) holds one model replica. Then, we let the workers train their local replica using their own data shard. However, it is possible to coordinate the workers in such a way that, together, they will optimize a single objective. There are several approaches to achieve this, and these will be discussed in greater detail in Chapter 2.

Nevertheless, a popular approach to optimize this objective, is to employ a centralized *parameter server* [1, 3, 2]. A parameter server is responsible for orchestrating model updates coming from different workers. It does this by incorperating gradient updates (albeit, in an non-naive way) of the workers into a model which was replicated to the workers.

Figure 1.2

1.2 Optimization algorithms

1.2.1 Gradient Descent

1.2.2 Evolutionary Optimization

1.2.3 Particle Filters

1.3 Problem Statement

1.4 Thesis Outline

Chapter 2

Distributed Deep Learning

2.1 Model Parallelism

2.2 Data Parallelism

2.2.1 Synchronous Data Parallelism

2.2.2 Asynchronous Data Parallelism

Bibliography

[1] Jeffrey Dean et al. “Large scale distributed deep networks”. In: *Advances in neural information processing systems*. 2012, pp. 1223–1231.

[2] Benjamin Recht et al. “Hogwild: A lock-free approach to parallelizing stochastic gradient descent”. In: *Advances in Neural Information Processing Systems*. 2011, pp. 693–701.

[3] Sixin Zhang, Anna E Choromanska, and Yann LeCun. “Deep learning with elastic averaging SGD”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 685–693.

Appendices