Computer System Design: Project

Project Goals

1. To make further consideration about the design of distributed system architecture and interactions in it. To understand the importance of modularity of computer system.
2. To implement a simplified distributed machine learning system which is able to combine scale and flexibility as much as possible. Due to the limitation of practical, you are supposed to simulate the distributed machine learning system using a single machine.

Backgrounds

"Tensorflow" is a popular machine learning tool for research and production. It offers APIs for experts to develop for desktop, mobile, web and cloud. And tensorflow makes the training of deep learning models more efficiently. If more than one device is provided, applying distributed architecture of tensorflow can really speed up the training process of deep neural networks. The distributed architecture of tensorflow contains four parts: a client, a distributed master, two workers applying for different tasks in computation during training process of a deep neural network(DNN). The interaction of these components is illustrated in Figure 1.

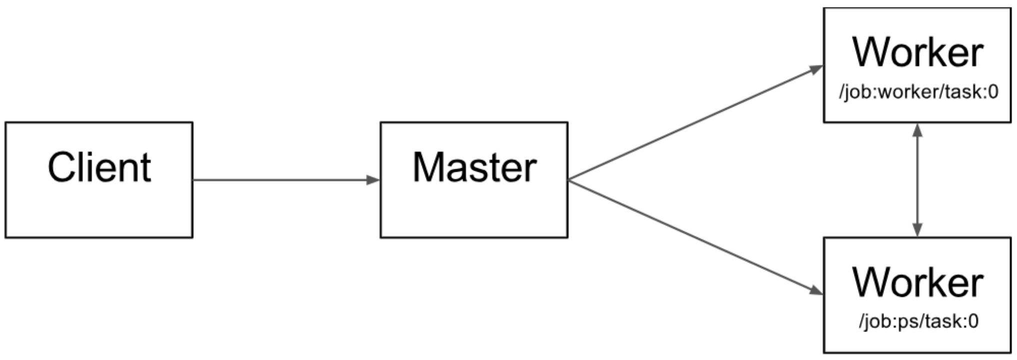

Figure 1.

In common cases, the two workers need to work on different servers, however, here, you are required to simulate the architecture on a single device.

Architecture

In your design, there are at least four parts need to be completed.

- *Client*

The client in TensorFlow is used to build the computation graph. The client creates a session, which sends the calculation graph definition to the distributed master. An example of calculation graph is described in Figure 2. When the client evaluates a node or nodes in the graph, the evaluation triggers a call to the distributed master to initiate computation. A single server can directly interact with multiple tensorflow servers, and a single server can serve multiple clients. But here we just focus on a simplified version of the distributed architecture containing only one client.

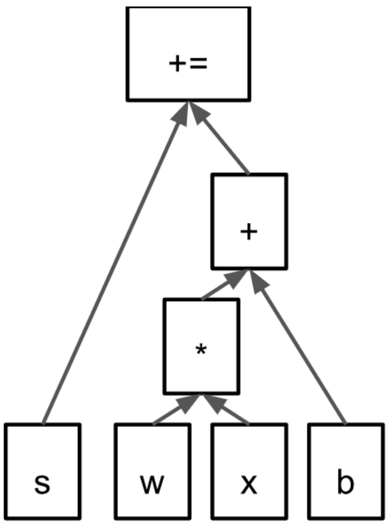

Figure 2. The client has built a graph that applies weights (w) to a feature vector (x), adds a bias term (b) and saves the result in a variable (s).

- *Distributed Master*

The master can see the overall computation graph, and it is used to do some optimizations on the computation process, i.e., common subexpression elimination and constant folding. Besides, the master is responsible for task assignment to several workers, i.e., the calculation of optimized subgraphs/subexpression.

- *Workers/Servers*

There exist two Server nodes. The first one is used to complete the calculation tasks, which is defined as worker server below. The other is parameters server, which is applied to send parameters to worker server in each training step. The distributed master makes a partition of the given calculation graph in Figure 2. Parameters server can update parameters according to the calculation results from worker server. As shown in Figure 3, model parameters are placed in parameter server and operations are placed in worker server.

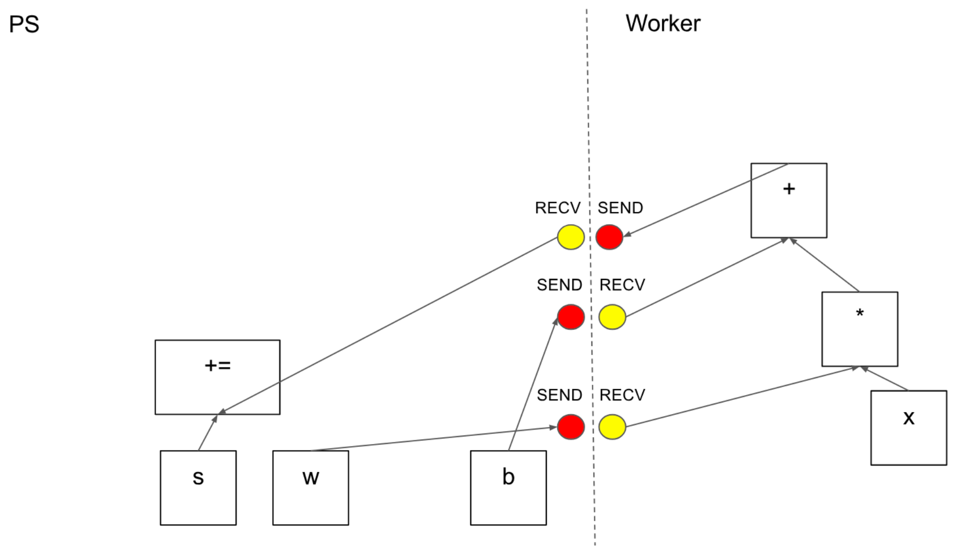

Figure 3.

Requirements

You are supposed to implement the whole architecture in distributed Tensorflow, including at least four parts illustrated above (Client, Distributed Master, Worker Server and Parameters Server). The program should be run correctly on a single machine.

The Client will receive an input vector x, and finally an output y is supposed to get from the program after several training steps. It is suggested that the training data samples (xi,yi) and testing samples are provided by yourself. We also provide some testing samples for you to verify you program. The detailed description of this dataset will be unrolled in Appendix.

In addition, what principles should be followed for updating parameters, such as w, b in Figure 2, need to be designed by yourself. Note that there are no requirements for the accuracy of your designed architecture. (if you want to know more details about the skills of updating parameters, you can find some references of “backpropagation” and “gradient descent”).

Submission Materials

Your submission should at least include the following parts:

Document which contains the following contents

(1) The architecture of your design.

(2) Functions or classes that you have implemented.

(3) Explanation of your program. Why you design in this way. Describe the advantages and disadvantages of your work.

(4) Describe the difficulties that you met and how you solve them

Source code

Providing testing samples is suggested. Furthermore, you can submit a video to show the program you make.

Grading Policy

Document (60%) + Program (40%)

Any form of cheating is not allowed, once confirmed, you will get zero for this lab, so be honest and complete your lab independently.

Deadline

Deadline: 2018.11.1 12:00:00

Please submit all the materials needed before deadline, otherwise your final grade on project will be discounted.

Appendix

- *Data File*

student_data.csv

The dataset originally came from here: http://www.ats.ucla.edu/

- *Data Description*

There are some results of student admissions to graduate school at UCLA based on three pieces of data: GRE Scores (gre), GPA Scores (gpa), Class rank (rank). The values of gre, gpa and rank comprise the input vector x. and the admission result is the output y. The admission result is expressed as 0 or 1.(1 means the student is admitted while 0 means the student is refused.)

- *Tips*

It is recommended that you may scale the data before your experiment. We notice that the range for gpa is 1.0-4.0, and the range of rank is 1-4, whereas the range for gpa scores is roughly 200-800, which is much larger. This means our data is skewed, and that makes it hard for the computation graph to handle. Let's fit our three features into a range of 0-1, by dividing the gpa and rank by 4.0, and the gpa score by 800.