Final Report - P1

Stochastic Gradient Descent - SGD

Shangwen Yan | N17091204 | sy2160

Introduction & Background

Gradient Descent

Gradient Descent is a optimization method that is used for finding the optimal (best) **weight vector** which could minimize the loss.

Typically when we want to predict the tendency or find the rule of a specific scenario problem in real world, we would **build a model** first to desceibe the problem in a mathematical way. Fitting a model is done by either directly solving for the “best” setting of model parameters or by iteratively refining the model weights typically from some random initialization.The objective is to find the optimal (best) weight vector (parameters of the model) that could make the model describe the problem well.

In the process of finding the optimal parameters, we need to choose a **loss function**, i.e. MSE Loss, Cross Entropy Loss, Negative Log Loss, etc. to measure the difference between real value(label) and the predicted value(label) in a quantative way. When the ouput of the loss function is minimized, which means the difference between real label and predicted label is minimized ,we could say the parameters are optimal.

Note that the loss function typically has the form of the sum over N observations/data samples:

$$L(w) = \frac{1}{N} \sum_{i=1}^{N} L_i(w)$$

where w is a parameter vector and N is the number of data points(samples).

Fitting a model to the data is done by optimizing a loss function (or in other words objective function L). This optimization is equivalent to searching for the “best” setting of model parameters. This procedure is called training.

But how to minimize the loss? Typically we would calculate the gradient of loss towards parameters, and update the weight vector by:

$$w^{t+1} = w^t - \alpha \nabla_w L(w)$$

where w^t denotes the parameter vector in t^{th} iteration, and α is the learning rate (step size).

Stochastic Gradient Descent

Stochastic setting : most machine learning algorithms rely on the assumption that the data is generated by a stochastic process (many online learning algorithms go one step further and allow the data to be generated by an adversary, that is possibly adaptive).

If the dataset is very large it can be impossible to evaluate the loss or its gradient on the entire dataset (as it is done in example in GD algorithm).

Stochastic Gradient Descent (SGD) algorithm solves this problem. It uses gradient update of the form:

$$w^{t+1} = w^t - \alpha \frac{1}{b} \sum_{i=1}^{b} \nabla_w L(w)$$

where b is the size of the mini-batch (b << N), and α is the learning rate (step size).

Thus it approximates the true gradient of L(w) with a gradient computed on a mini-batch of examples (which can be as small as a single example).

At every iteration, stochastic optimization method samples examples from the training set uniformly at random to form a mini-batch and then updates the model parameters using the obtained sample.

WHY

Describe the problem that exists

With the fast development and wide use of electronical equipment, the magnitude of dataset is becoming so voluminous and complex that traditional data-processing application software are inadequate to deal with them.

WHAT

Identify through literature search what solutions have been proposed

For machine learning problem, it can be impossible to evaluate the loss or its gradient on the entire dataset (as it is done in example in GD algorithm) if the dataset is very large. That's the reason why we use **Stochastic Gradient Descent** (mini-batch instead of whole dataset).

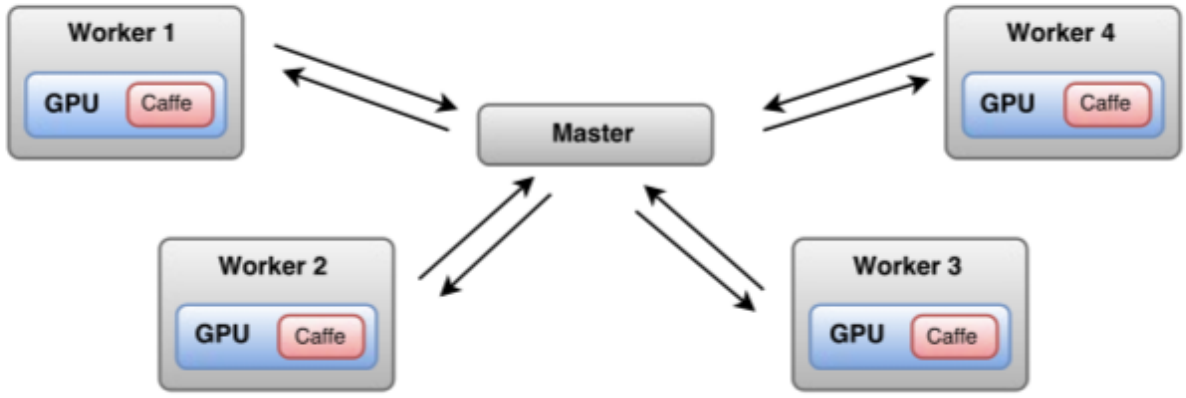
However, when the size of dataset is becoming huge, simply using SGD is far not enough. Typically, we want to do the calculation among distributed system which means several nodes(computers) do one thing simultaneously or called in parallel.

The concept of Map Reduce fo hadoop is a good solution for this problem. Also, the concept of RDD(Resilient Distributed Datasets) in spark is also used for the distributed computation and parallism.

HOW

SparkNet provide a parallelization scheme for SGD that requires minimal communication. And the approach is not specific to SGD.

Spark consists of a single **master node** and a number of **worker nodes**. The data is split among the Spark workers. In every iteration, the **Spark master broadcasts the model parameters to each worker**. Each worker then runs SGD on the model with its subset of data for a fixed number of iterations τ or for a fixed length of time, after which the resulting model parameters on each worker are sent to the master and averaged to form the new model parameters. The architecture is shown as the picture below:



The standard approach to parallelizing each gradient computation requires **broadcasting and collecting model parameters** (hundreds of megabytes per worker and gigabytes in total) after every SGD update, which occurs tens of thousands of times during training.

Typically a parallelization scheme attempts to distribute the computation at each iteration by dividing each **minibatch** between the K machines, computing the gradients separately, and aggregating the results on one node.

By this way, the running time could be reduced compared with not using parallelization.