Lab Course: Distributed Data Analytics
Exercise Sheet 4

Mofassir ul Islam Arif
Information Systems and Machine Learning Lab
University of Hildesheim

Submission deadline: Monday June 8 , 10:00AM (on LearnWeb, course code: 3116)

Instructions

Please following these instructions for solving and submitting the exercise sheet.

1. You should submit two items. a) a zip or a tar file containing python scripts and b) a pdf document.
2. In the pdf document you will explain your approach (i.e. how you solved a given problem), and present your results in the form of graphs and tables.
3. The submission should be made before the deadline, only through learnweb.
4. In each lab you have to show your solution works for $P = \{1, 2, 3, 4, 5, 6, 7, 8, \ldots \ldots\}$ and provide the timing results (either it is stated in the question or not)

Distributed Machine Learning (Supervised) (10 Points)

In this exercise sheet you are going to implement a supervised machine learning algorithm in a distributed setting using MPI (mpi4py). We will pick a simple Linear Regression model and train it using a parallel stochastic gradient algorithm (PSGD)[1].

0.1 Dataset

You have to use two datasets for this task. Read the description of the dataset. You have to use them for learning a regression model.

- Dynamic Features of VirusShare Executables Data Set `https://archive.ics.uci.edu/ml/datasets/Dynamic+Features+of+VirusShare+Executables`
- KDD Cup 1998 Data Data Set `https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1998+Data`

Please divide your data into 70% train and 30% test. You will train your model on train part of the data and will test it on test part of the data. Use RMSE train and test score for measuring the performance of your model.

0.2 Linear Regression

In a supervise machine learning setting, the training data $\mathcal{D}$ consists of N training instances each represented by a feature vector $\mathbf{x} \in \mathbb{R}^M$ with M features and a label $y \in \mathbb{R}$. A Linear Regression model $\hat{y}(\mathbf{w}) = \sum_{i=1}^{M} x_i w_i$ with model parameters $\mathbf{w} \in \mathbb{R}^M$ can be learned by minimizing the object function,

$$\mathcal{F}(\mathbf{x}, \mathbf{w}) = \min_{\mathbf{w}} \sum_{\mathbf{x} \in \mathcal{D}} \mathcal{L}(\mathbf{x}, \mathbf{w}) \tag{1}$$

where $\mathcal{L}(\mathbf{x}, \mathbf{w})$ is a square loss. Stochastic Gradient Decent based algorithms are most widely used to optimize the object function (1). It randomly permute over the training instances and update the model parameters using (2)

$$\mathbf{w} = \mathbf{w} - \eta \nabla \mathcal{L}(\mathbf{x}_i, \mathbf{w}), \mathbf{x}_i \in \mathcal{D} \tag{2}$$

Please refer to Annex section for more resources on Linear Regression.

[1] Zinkevich, M., Weimer, M., Li, L., & Smola, A. J. (2010). Parallelized stochastic gradient descent. In Advances in neural information processing systems (pp. 2595-2603).

0.3 Parallel Stochastic Gradient Descent

The PSGD learning algorithm is a very simple algorithm that learns in a distributed setting. Lets assume we have P workers. We assign one of the worker a role of master and others as workers. The role of the master worker is to distribute the work i.e. Training data, and average the local model learned by each individual worker into a global model. Each worker on the other hand, gets its work from master and learns a local model using SGD updates. Once an epoch (i.e. a complete round over the data) is complete it sends back the local model to the master worker. The master after receiving models from all the workers, averages them and sends the new global model to each worker for the second round. The PSGD algorithm is summarized in Fig 1 and Fig 2.

1 Parallel Linear Regression 10 points

The first task in this exercise is to implement a linear regression model and learn it using PSGD learning algorithm explained above. You will implement it using mpi4py. A basic version of PSGD could be though of using one worker as a Master, whose sole responsibility is getting local models from other workers and averaging the model. A slight modification could be though of using all the workers as worker and no separate master worker. [Hint: Think of collective routines that can help you in averaging the models and return the result on each worker]. Once you implement your model, show that your implementation can work for any number of workers i.e. $P = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

2 Performance and convergence of PSGD 10 points

The second task is to do some performance analysis and convergence tests.

1. First, you have to check the convergence behavior of your model learned through PSGD and compare it to a sequential version. You will plot the convergence curve (Train/Test score vs the number of epochs) for $P = \{1, 2, 3, 4, 5, 6, 7, 8\}$. You have to use any sequential version of Linear Regression for $P = 1$ (Only for sequential version you can use sklearn).

2. Second, you have to do a performance analysis by plotting epochs vs time. Time your program for $P = \{1, 2, 3, 4, 5, 6, 7, 8\}$. Essentially, we want to show that more epochs per unit time can be achieved using a parallel setting.

3. Lastly, Your model needs to converge and also show time performance improvement.

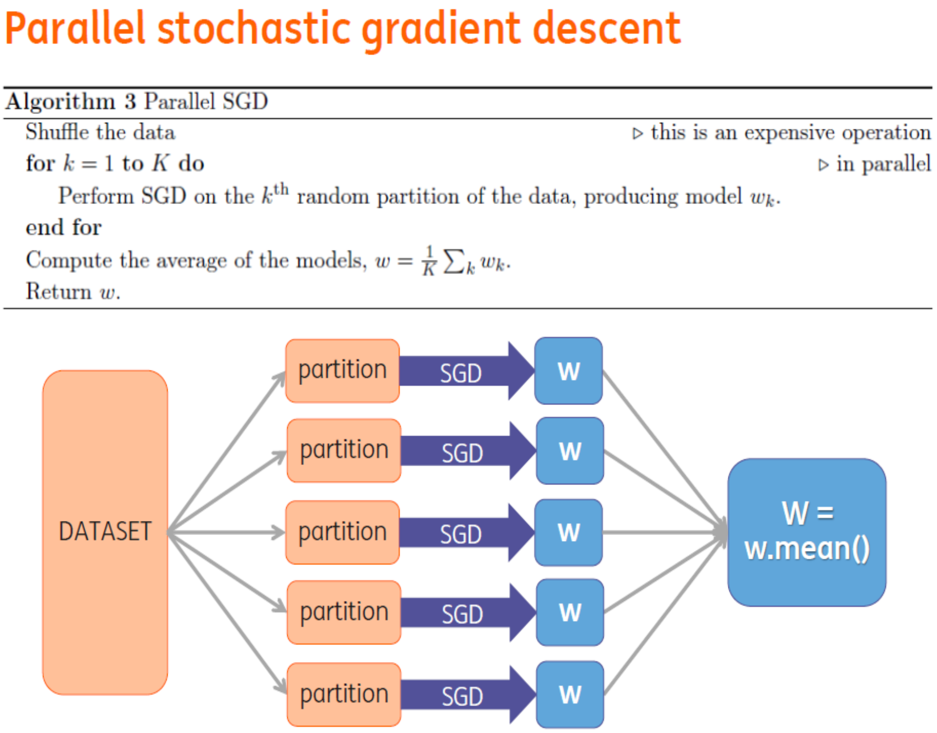
Annex

1. Linear Regression `https://www.ismll.uni-hildesheim.de/lehre/ml-17w/script/ml-02-A1-linear-regressi` `pdf`

2. Linear Regression Tutorial `https://machinelearningmastery.com/linear-regression-tutorial-using-gradi`

3. sklearn SGD `http://scikit-learn.org/stable/modules/sgd.html`

4. RMSE is explained at `https://www.kaggle.com/wiki/RootMeanSquaredError`



Parallel stochastic gradient descent

```
Algorithm 3 Parallel SGD
  Shuffle the data                                  ▷ this is an expensive operation
  for $k = 1$ to $K$ do                                              ▷ in parallel
      Perform SGD on the $k^{\text{th}}$ random partition of the data, producing model $w_k$.
  end for
  Compute the average of the models, $w = \frac{1}{K}\sum_k w_k$.
  Return $w$.
```

Figure 1: The working of Parallel Stochastic Gradient Descent Algorithm https://github.com/blebreton/spark-FM-parallelSGD/blob/master/img/parallel_sgd.PNG

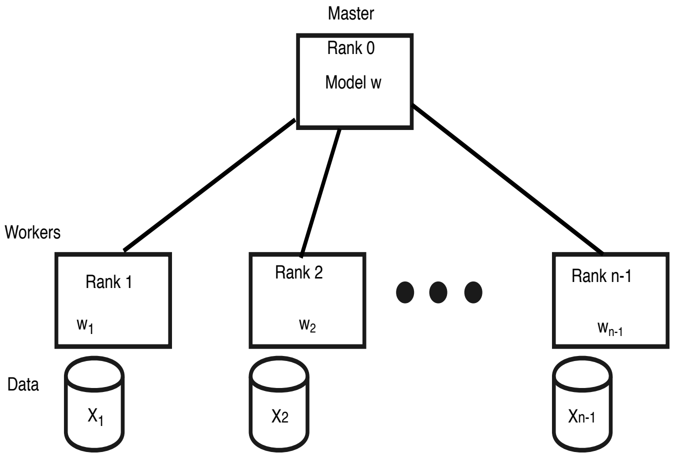


Figure 2: The Parallel Stochastic Gradient Descent Algorithm with a Master/ Worker configuration.