Implementation of SGD for Generalized Linear Model on Husky

Student: NG Ka Lok

Supervising Professor: James CHENG

The Chinese University of Hong Kong

Introduction

This work is implementing a library about Stochastic Gradient Descent (SGD) for Generalized Linear Model on Husky and PyHusky, which are the open-source framework for distributed computing[1].

Motivation

1. Provide efficient Big-Data analysis tools on distributed computing framework
2. Prepare handy API for customized linear model.

Working principles and theories

SGD is an optimization method for minimizing a cost function. In this method, we take gradient of cost function with parameters vector, which indicated the worst direction in parameters space of minimizing the cost function. Therefore, ones can find the minimal point by taking step to the opposite direction after calculating the average gradient of a batch of samples until converge.

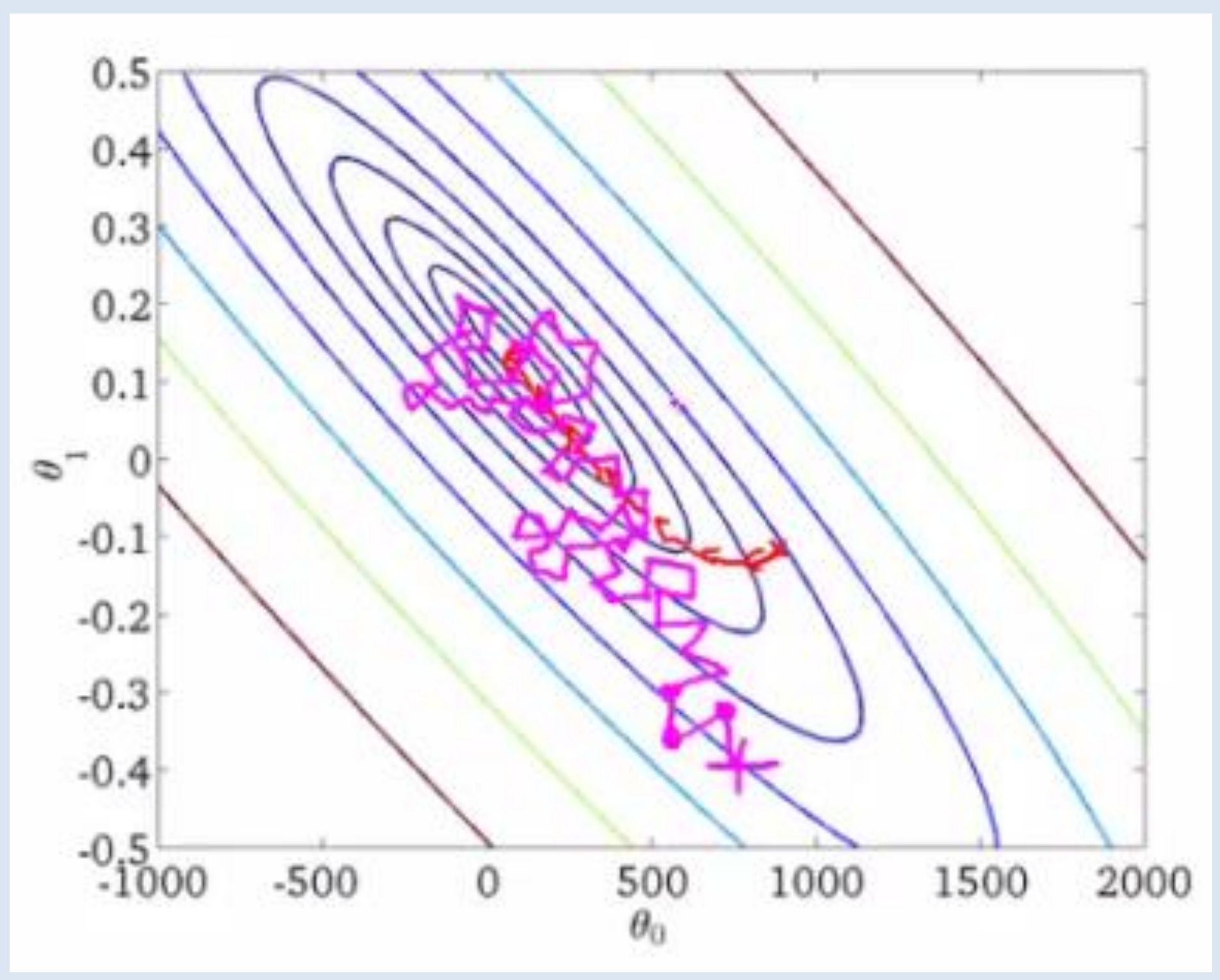


Figure 1[2]. SGD in 2-D parameters space

With SGD, ones can find the approximant solution of Generalized Linear model by defining the cost function as the mean square error of model's prediction and output. For example, in the linear regression model, the prediction of a data point is

$$y = X^T \cdot W$$

where X is the feature of the data point and W is the weighting. Hence, the updating process is:

```
Until converge:
    For each sample i:
        $W_j^{t+1} = W_j^t - \alpha(y - W^T \cdot X)$
```

in which α is the learning rate.

Programming Model

SGD_model can build customized linear model. The key API:

```
class SGD_model:
   def initialization(gradient_func,
error func, n_feature)
   def train(object_list)
   def avg_error(object_list)
   def get_param()
```

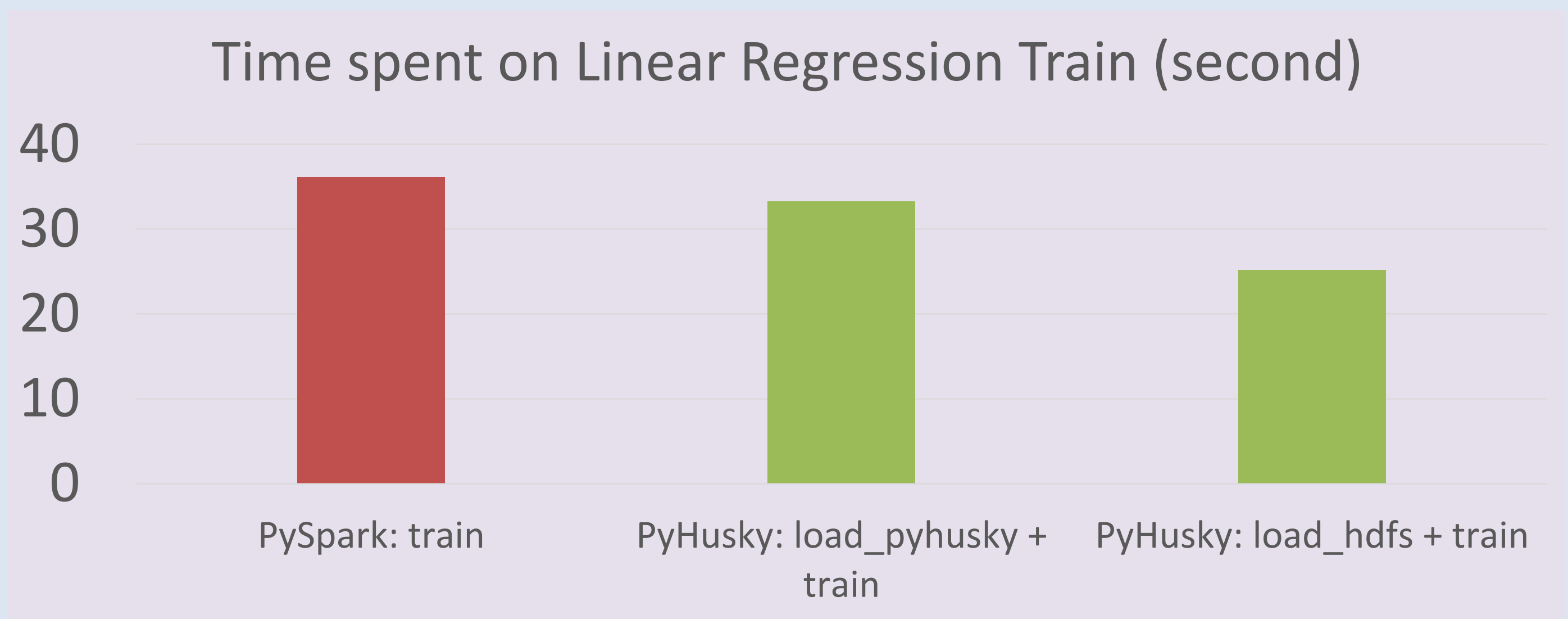
In Husky for C++, we have wrapped the linear regression model, which is inherited from SGD_model so most of the method is similar, only major difference:

```
Class SGD_LinearRegression:
def initialization(get_X, get_y,
n_feature)
   def score(object_list)
   … other methods
```

The Linear Regression API in PyHusky:

```
Class LinearRegression():
   def initialization()
   def load_pyhusky(pyhusky_list)
   def load_hdfs(hdfs_url)
   def train()
```

Result and discussion



We can see that PyHusky is more efficient on handling large-scale data.

The Dataset used in the test is Million Song Dataset, in which there are over 500,000 line of data and each has 99 features. The number of iterations of SGD is 200.

Conclusion

With the efficient distributed computing framework, Husky, we can analysis data and build linear model in a faster way.

Acknowledgement

I would like to express my special thanks to my supervising professor James Cheung who gave this great opportunity as well as other postgraduate student who help me a lot to finish this project.

Reference

1. F. Yang, J. Li, and J. Cheng. Husky: Towards a more efficient and expressive distributed computing framework. PVLDB, 9(5):420–431, 2016.
2. R. Ward, "Stochastic Gradient Descent with Importance Sampling", The University of Texas at Austin, 2014.