Progress Report

Distributed Learning with Deep Convolutional Neural Networks

Student: LIU Jinchao

SID: 54381657

Date: Nov 24, 2018

Supervisor: Prof. Ding-Xuan ZHOU

1. Introduction

Due to the increasing size (especially depth) of Neural Network (NN) models and the ever-increasing amount of required training data, training a deep learning model locally becomes more and more impractical and inefficiency.

To make deep learning better serve the research and industry, distributed learning is one of the best solutions. This project focuses on distributed learning with deep convolution neural networks, seeking to find a simple and easy-used Distributed DCNN.

2. Methodology

2.1 Data (**Million song challenge**)

1. It could be download from http://www.samyzaf.com/ML/song_year/song_year.zip

2. It consists of **515345** records of songs that were composed during the years 1922-2011. Each record consists of 91 features. The first feature is the year in which the song was composed, and the remaining 90 features are various quantities (float) related to the song audio.

2.2 Tools and Environment

1. TensorFlow version 1.8.0
   Keras (Built-in TensorFlow)
2. GitHub: http://github.com/liu1322592019/Distributed-CNN

2.3 Design

1. Using pandas to load the data

2. Data processing:
    a. trainX: Normalization and reshape
    b. trainY: Using np_utils.to_categorical() to transform into one-hot vectors.
    c. Both: Using tf.data.Dataset.from_tensors() to group as feed data for estimator model.

3. Model:
    a. Using Sequential Keras model (easy to develop: adding or deleting layers)
    b. Basic structure:
        i. Repeat {Convolution1D+Activation (Relu)} for n times.
        ii. Adding Dropout layers for some Convolution layer to avoid overfit.
        iii. Adding a few Fully connected layer (Dense) at the last.
        iv. Using "SoftMax" as the last layer for classification.
    c. Using default (TensorFlow) distributed strategy.
    d. Using tf.keras.model_to_estimator() to transform our model into an estimator which could be train distributed.

4. Other tries
    a. Pyspark based methods: When importing pyspark, some error occurs.
    b. Open-mpi based methods: Failed in building the environment on both Mac OS and Ubuntu 18.04

2.4 Implementation

1. What I have done: (Completed)
    a. Executable script (both .py and. ipynb files) base on the method described in 2.3.3.
    b. Non-executable script base on pyspark (Environment Error.)
    c. One method to deal with the unbalance distributed of data over time.
        i. Increasing the weight of less-data year by repeating.

2. In-progress:
    a. Data argumentation with noise

2.5 Result

1. Training locally: (MacBook Pro (Retina, 13-inch, Early 2015), batch size = 24-100)
   a. Training: Accuracy: 0.988890
   b. Test: Accuracy: 0.988890
2. Training distributed: (1 worker only situation, Computer in MMW2462, i7-8700k + GTX1080Ti, batch size = 100-10000)
   a. Training: Accuracy: 0.988890
   b. Test: Accuracy: 0.988890
3. Other result:
   a. Nearly the accuracies of all models reach 0.988890 after just 1 epoch and remain this value for the following epochs. (One possible reason the huge size of training data.)

3. Project Planning

1. Developing **some deeper models**
2. Trying more method to **increase the accuracy of predicting on less-data year**
3. **More distributed settings** should be tested to find the most suitable ones.
4. **More powerful layers** or layer settings.
5. **More powerful callback** functions like earlystop.

4. References

Zhou, D. (2018). Deep distributed convolutional neural networks: Universality. Analysis and Applications,16(06), 895-919. doi:10.1142/s0219530518500124