STATS 790
draft1

Yilong Zhai

28 April, 2023

Contents

Introduction . . . 2
Experiment . . . 3
Conclusion . . . 6
Reference . . . 7

Introduction

For this project, we have decided to compare the performance of two popular deep learning frameworks, TensorFlow and PyTorch. We will be conducting our comparison in the Python(Guido van Rossum 2008) and R environment(R Core Team 2020), which are widely used language in the data science and machine learning community. By comparing the performance of these two frameworks, we aim to gain insights into their strengths and weaknesses. The model we are plan to build under these two framework is Multilayer perceptron

PyTorch(Meta AI 2017) is an open-source machine learning library developed by Facebook AI Research that enables developers to create and train deep neural networks. It is known for its ease of use, flexibility, and speed, making it a popular choice among researchers and practitioners in the machine learning community. One of PyTorch's key features is its dynamic computation graph, which allows for dynamic creation and modification of computational graphs during runtime. This enables researchers and developers to build complex models and experiment efficiently with different network architectures. PyTorch also provides a range of pre-built functions and modules for building neural networks, including various activation functions, loss functions, and optimization algorithms. This allows developers to quickly build, train, and test their models without having to write everything from scratch. Another key aspect of PyTorch is its ability to seamlessly integrate with other libraries and tools commonly used in the machine learning ecosystem, such as NumPy, pandas, and scikit-learn. This makes it easy to load, preprocess, and manipulate data, as well as visualize and analyze results. PyTorch also has a strong community of contributors and users, who actively develop and maintain a vast array of open-source resources, including pre-trained models, tutorials, and documentation. This makes it easy for developers to learn, experiment with, and apply PyTorch to a wide range of real-world problems.

TensorFlow is an open-source software library developed by the Google Brain Team for building and training machine learning models. It also provides a wide range of tools, libraries, and resources that simplify the process of building, training, and deploying machine learning models. TensorFlow has become one of the most popular and widely used machine learning frameworks because of its scalability, flexibility, and ease of use. One of the key features of TensorFlow is its ability to perform computations on large-scale datasets, which is essential for building complex machine learning models. TensorFlow provides a variety of APIs for building different types of models, including neural networks, decision trees, and linear models. It also includes a wide range of built-in functions and operations that make it easy to perform complex mathematical computations and data transformations. TensorFlow includes support for distributed computing, which means that it can distribute computations across multiple devices or machines to accelerate the training process. TensorFlow also provides a powerful visualization toolkit, which makes it easy to visualize and analyze the performance of machine learning models. This includes tools for visualizing training curves, exploring model architecture, and visualizing high-dimensional data. In addition to the core TensorFlow library, there are also a variety of high-level APIs that provide a simplified interface for building machine learning models. These APIs include TensorFlow Estimators, TensorFlow Hub, and TensorFlow Lite, which make it easier to build, train, and deploy models in a variety of settings.

Overall, TensorFlow is a powerful and flexible machine learning framework that provides a wide range of tools and resources for building and training machine learning models. Its scalability, flexibility, and ease of use have made it a popular choice for researchers, developers, and data scientists working on a wide range of machine learning applications.(Google Brain 2017)

Country status is always an important factor to measure the overall national strength. There are multiple factors that may affect the country status. In this project, we are going to study the relationship between country status and these factors. The dataset is come from WHO("World Health Organization," n.d.), who provided the country status in the dataset. This dataset has been uploaded to Kaggle(**data?**), a reliable and public database. The predictor variables are life expectancy, total expenditure, GDP, population, and schooling, which are all continuous variables, and the response variable country status is categorical variable with two categories, developing or developed. This dataset contains the data for above variables from 2000 to 2015 for all countries. Dara from year 2011 would be used as testing set, all other data would be used as training set.

Experiment

First we will test the two framework by a classification problem: predicting the contry status of testing set in the environment of Python.

When constructing a multilayer perceptron (MLP) in TensorFlow, it is important to choose the appropriate argument settings to optimize the performance of the model. The method we used to find the best hyperparameters (number of layers, nodes and learning rate) is GridSearchCV, which means we test a set of hyperparameters settings and find the best setting by accuracy and loss. The advantage of this method is efficiency, it could find the best setting quicker than the method Trial and Error in a small dataset; however, if we have a larger dataset, it will become computationally expensive. GridSearchCV is built in package sklearn.model_selection.

Also, in order to preventing overfitting while still allowing the network to learn from the data, we will add dropout option in our MlP models. The value of dropout will start at small value 0.1, if the model is still overfitted, we will increase the value of dropout by 0.1 each time. Overfitting means the model performed really well on training set but performed poor on testing set. By testing a set of dropout values manually from 0.1 to 0.5, we determined that when dropout=0.1, the Tensorflow model performed the best on the problem of overfitting.

By selecting these specific arguments, we have found that the MLP achieves the highest level of accuracy and efficiency, where our model could get an accuracy of 91% . Moreover, the loss for these 1000 epochs decreased from 0.424 to 0.2476. The learning rate we choose is 0.01 by comparing the performance of 0.001,0.01,0.05, and 0.1. Sigmoid activation function is well-suited to binary classification tasks, which is often the case for MLPs; however, One of the main disadvantages of the sigmoid activation function is the vanishing gradient problem. When using backpropagation to train a neural network with sigmoid activation functions, the gradient of the sigmoid function can become very small for large or small input values. This means that the gradient of the loss function with respect to the weights of the neural network becomes very small, and the weights are updated very slowly or not at all, leading to slow convergence or getting stuck in local optima. Therefore, we introduced the activation function rectified linear unit (ReLU). ReLU activation function is used in all layers, as it helps to prevent the vanishing gradient problem, which is the biggest advantage of this activation and improve the weakness of sigmoid activation function. In the input layer and output layer, there are 5 and 1 nodes, which matched the dimension of input and output; there are 3 nodes in the hidden layer as the grid search suggested. The solver would be Adam.

When building a multilayer perceptron (MLP) using PyTorch, we still select the best hyperparameters that GridSearchCV recommended. The loss for these 1000 epochs decreased from 0.692 to 0.165. We define the first layer to be fully connected with five input features and three output features. The second layer can also be fully connected, with three input features and two output features. We will use the ReLU activation function as grid search suggested, which has been shown to work well in many scenarios. The output layer will have two nodes, corresponding to the two classes we are trying to classify.

It's worth noting that the number of layers and the number of nodes in each layer, as well as the choice of activation function, are all hyperparameters that need to be tuned based on the specific problem at hand. In general, adding more layers or nodes can increase the model's capacity, but can also make it prone to overfitting if the dataset is not large enough or if regularization techniques are not used. Therefore, in our grid search process, we finally select the model with 3 layers.

In order to ensure a fair comparison between the performance of tensorflow and pytorch, we set the random seed as 1. Our Multilayer Perceptron (MLP) is designed with two hidden layers. In Tensorflow, we need to manually select the activation functions for each layer, which adds to the complexity. On the other hand, PyTorch is less complex, making it easier to use. Furthermore, we built these two models under GooGle Colab, which get rid of the effect of GPU and CPU.

Considering the running time, tensorflow takes around 3 minutes and 15 seconds to run 1000 epochs, while pytorch only requires 1 second for the same number of epochs. This indicates that pytorch is much faster than Tensorflow.

When compare the accuracy of PyTorch and Tensorflow in python. The accuracy of Tensorflow and PyTorch are 91% and 94% correctly. The True positive rate for Tensorflow is only 58.15% while the true negative rate is 100%. The True positive rate for PyTorch is 79.53% while the true negative rate is 100%. Hence, pytorch is more accurate in this case.

Furthermore, we also evaluated the memory usage for both frameworks. The peak memory usage for tensorflow was found to be 2.5 MB, while the peak memory usage for pytorch was 0.16 MB. Therefore, pytorch consumes significantly less memory compared to tensorflow.

In this classification problem, Pytorch performs better than tensorflow from the perspecitve of complexity, running time, accuracy, and memory usage in the environment of python.

Next, we will compare the performance of PyTorch in python and R. We use the same dataset and same structure that grid search recommended with a dropout=0.1 to build MLP in R. The accuracy of PyTorch in R is 94%, with a true positive rate of 74% and false negative rate of 98%. The running time in R to run this PyTorch is 1 second and the memory used is 0.3 MB.

By comparing the results in R and python, we could find that the difference is insignificant cross multiple front end language by the same back end language; therefore, the impact of the selection of back end language is significant.

Conclusion

In our experiment, we compared the performance of PyTorch and TensorFlow on a classification problem using a simple constructed MLP. Our results indicate that PyTorch outperforms TensorFlow in terms of running time, accuracy, and memory usage in the classification problem.

When choosing between PyTorch and TensorFlow, it is important to consider the specific needs of the user and the nature of the project at hand. If the project involves a lot of data preprocessing and manipulation, PyTorch might be a better choice due to its ease of use and flexibility. On the other hand, if the project requires extensive use of pre-trained models or production-level deployment, TensorFlow may be a more suitable option.

It is worth noting that the choice of framework is just one aspect of the overall project, and other factors such as hardware, data quality, and modeling techniques should also be taken into consideration. Ultimately, the success of a project relies on the ability to select and implement the best tools and techniques for the task at hand.

Reference

Google Brain. 2017. *TensorFlow*. https://www.tensorflow.org.

Guido van Rossum. 2008. *Python3.0*. https://www.python.org.

Meta AI. 2017. *PyTorch*. https://pytorch.org.

R Core Team. 2020. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

"World Health Organization." n.d. https://www.who.int.