STATS 790
draft1

Yilong Zhai

11 April, 2023

Contents

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 environment, which is a 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 and provide useful information to researchers and practitioners who are considering using these tools for their projects. The model we are plan to build under these two framework is Multilayer perceptron

PyTorch 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 more complex models and experiment with different network architectures in a more efficient way. 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.

In summary, PyTorch is a powerful and flexible deep learning library that provides a range of tools and resources for building, training, and testing neural networks. Its dynamic computation graph, pre-built modules, and seamless integration with other libraries make it a popular choice among researchers and practitioners in the machine learning community.(AI 2017)

TensorFlow is an open-source software library developed by Google Brain Team for building and training machine learning models. It 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. Another key feature of TensorFlow is its ability to support both CPU and GPU computations, which makes it ideal for deep learning applications. 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.(Brain 2017)

Country status (developing or developed) is always an important factor to measure the overall national strength. There are multiple factors that may affect the country status, such as life expectancy, adult mortality , BMI, under-five deaths, total expenditure, GDP, population, Income composition of resources, and schooling. In this project, we are going to study the relationship between country status and these factors. The dataset is come from WHO, which has been uploaded to Kaggle(WHO 2016), a reliable and public ataset website. The predictor variables: life expectancy, adult mortality, BMI, under-five deaths, total expenditure, GDP, population, Income composition of resources, and schooling are all continuous variables, and the response variable country status is categorical variable. This dataset contains the data for above variables from 200 to 2015 for all countries. The algorithm will be designed under the environment of R(R Core Team 2020) and

RMarkdown(Xie, Dervieux, and Riederer 2020).

Country status, whether a country is classified as developing or developed, is a critical measure of a nation's overall strength. Many factors can influence country status, including life expectancy, adult mortality, BMI, under-five deaths, total expenditure, GDP, population, income composition of resources, and schooling. In this project, we aim to investigate the relationship between country status and these factors using data obtained from the World Health Organization (WHO), which is publicly available on the Kaggle dataset website(WHO 2016). The dataset includes predictor variables that are continuous in nature, such as life expectancy, adult mortality, BMI, under-five deaths, total expenditure, GDP, population, income composition of resources, and schooling, along with the categorical response variable, country status. The data spans from 2000 to 2015 and covers all countries. Data 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.

When constructing a multilayer perceptron (MLP) in TensorFlow with 3 layers, it is important to choose the appropriate argument settings to optimize the performance of the model. After careful experimentation and analysis, we have determined that the optimal arguments for this particular MLP are as follows:

Input layer: 5 neuron nodes with the sigmoid activation function. Hidden layer 1: 3 neuron nodes with the rectified linear unit (ReLU) activation function. Hidden layer 2: 1 neuron node with the sigmoid activation function.

By selecting these specific arguments, we have found that the MLP achieves the highest level of accuracy and efficiency. The sigmoid activation function is used in the input layer and the final hidden layer, as it is well-suited to binary classification tasks, which is often the case for MLPs. ReLU activation function is used in the hidden layer 1, as it helps to prevent the vanishing gradient problem that can occur when using the sigmoid function in every layer. The MLP architecture is composed of a relatively small number of neurons, which is suitable for the input size of the data set, and has been found to provide the best balance between model complexity and performance.

Overall, this architecture is designed to maximize the accuracy of the model while minimizing computation time and memory usage.

When building a multilayer perceptron (MLP) using PyTorch with three layers, we can define the first layer to be fully connected with five input features and four output features. We can use any activation function we want, but popular choices include ReLU, sigmoid, or tanh. The second layer can also be fully connected, with four input features and two output features. Again, we can choose an activation function to use here. Finally, for the output layer, we will use the ReLU activation function, 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.

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.

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 comparing the accuracy, both under their best fitted model. Tensorflow correctly predicted 475 observations out of 522 testing observations, while pytorch predicted 492 observations accurately. 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 1.812198 MB, while the peak memory usage for pytorch was 0.138275 MB. Therefore, pytorch consumes significantly less memory compared to tensorflow, which is a crucial factor for applications with large datasets.

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

Next we will test the two framework by a regression problem. Predicting the life expectancy of testing set.

From the results we could see that both pytorch and tensorflow does not perform well on building simple MLP to solve regression problems. But pytorch still performed better on memory usage and running time.

Conclusion

In our experiment, we compared the performance of PyTorch and TensorFlow on a classification problem and a regression 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. However, both frameworks did not perform optimally on the regression 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

AI, Meta. 2017. “PyTorch.” 2017. https://pytorch.org.

Brain, Google. 2017. “TensorFlow.” 2017. https://www.tensorflow.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/.

WHO. 2016. “Life Expectancy.” https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who.

Xie, Yihui, Christophe Dervieux, and Emily Riederer. 2020. *R Markdown Cookbook*. CRC Press.