Birla Institute of Technology and Science Pilani, Hyderabad Campus

2nd Semester 2023-24, BITS F464: Machine Learning

Assignment No:5, Max. Marks: 07, Date given: 05.04.2024, Date of sub: 12.04.2024

Note: This assignment is about connectionist learning models/ Classifiers built using Artificial Neural Networks (ANNs). You should use PyTorch library (an open source ML library developed by Meta's AI research lab) to implement a Back Propagation Neural Network (BPN).

Further, you will be using Backpropagation Neural Networks (with Forward and backward passes as discussed in the class) to solve a regression task over a dataset using PyTorch, running over the Google colab.

PyTorch is a powerful open-source machine learning library which provides a flexible and dynamic computational graph mechanism, making it particularly suitable for deep learning and neural network-based research and development. You may refer to the introductory lectures to understand more about PyTorch functionalities. Additionally, may refer here: https://ai.meta.com/tools/pytorch/.

You'll use the provided (in this assignment) Abalone dataset for your task as discussed in the lecture. The dataset contains measurements of physical characteristics of abalones, a type of marine mollusk, and aims to predict their age (number of rings) based on a set of features as shown in first few records (using df.head() method in Python):

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Rings
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

The target variable (feature) to predict is "Rings", which represents the age of the Abalone. It is typically assumed to be related to the number of rings on the shell, although this relationship may NOT be linear.

Below are the steps you should follow to implement the classifier:

1. Import the OS environment variables and other supporting libraries like Pandas, NumPy and PyTorch. Load the provided CSV file into the code converting it into a Pandas data frame. Preprocess the data by performing one-hot encoding for the gender column (categorical values) and standardizing the continuous features. Do not forget to import the 'nn' submodule within PyTorch that will provide you all the required ANN functionalities (import tourch.nn as nn). Of course, you can build a neural network from scratch without using torch.nn. You are free to implement either way.
2. Split the data into training and testing sets using a ratio of 80:20 or 70:30.
3. Convert the datasets into PyTorch tensors as PyTorch uses torch.tensor, rather than numpy arrays.

(Please Turn Over)

4. Define a class that inherits from “nn.module”. When a class inherits from another class, it can inherit its methods and properties. In this case, your class will inherit the functionality of “nn.module”, which is the base class for all the neural network modules in PyTorch. This class should have the following architecture:
    - Input layer with 10 nodes (excluding the dummy variable for 'Gender'.
    - Two hidden layers with 64 and 32 nodes respectively, both activated by ReLU activation function. As discussed in the class this function overcomes the problem of Vanishing gradient as its’ derivative will always be 1 for positive input which means the gradient does not vanish. However, as it is a shallow network (with only two hidden layers), vanishing gradient may not be prominent.
    - Output layer with 1 node (for regression), representing the predicted age of abalone.

5. Use MSE as the error criterion (loss function) for the regression task given, and also use an Incremental (Stochastic) Gradient Descent algorithm as the optimizer with a learning rate of 0.1. Define the training parameters as below:
   criterion = nn.MSELoss() and optimizer = optim.SGD(model.parameters(), lr=0.1)

   The optimizer = … line creates an instance of Stochastic Gradient Descent (SGD) optimizer from the torch.optim module with a learning rate of 0.1 which is sufficiently small.

6. Train the model for a specified number of epochs (e.g., 100 epochs). Then, implement an evaluation function ‘evaluate’ that takes the trained model, testing data, and test labels. Evaluate the model's performance on the testing set by calculating the Mean Squared Error (MSE) loss. You can use the below code for reference:

```
model.eval()
    with torch.no_grad():
        outputs = model(X_test)
        mse = nn.MSELoss()
        loss = mse(outputs, y_test)
```

7. Tune the hypermeters such as the learning rate, batch size, the number of hidden nodes in a layer (too less will under-fit and too more will fail to generalize as discussed in the class), the number of hidden layers in the BPN and observe the changes in the MSE. Tabulate your observations.

8. Try adding more layers to the model and observe the changes. Use another optimizer called ‘Adagrad’ available in torch.optim module that Adapts the learning rates based on the frequency of parameter updates. Compare its’ performance with earlier used SGD and note down your observations.

9. Increase the number of hidden layers to 10 or 15 and use Sigmoid as the activation function in each hidden layer. Observe if there is any vanishing gradient problem in your new architecture and describe which architecture (in earlier part or this one) is better in terms of minimizing the Loss.

Submission Instructions: Same as that of the previous assignment. Deadline for submitting your work is 12th April 2024 midnight. Any clarification on this coding assignment may be emailed to I/C or f20202001@hyderabad.bits-pilani.ac.in, or f20210982@hyderabad.bits-pilani.ac.in.

References:

1. https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html
2. https://www.datacamp.com/tutorial/pytorch-tutorial-building-a-simple-neural-network-from-scratch
3. https://machinelearningmastery.com/develop-your-first-neural-network-with-pytorch-step-by-step/

---------- ~ ----------