Neural Network Project 1: Multilayer Perceptron

Zhicong Lu, Boxuan Zhang

1. Problem Definition
   1. Multilayer Perceptron

A Multilayer Perceptron (MLP) is a type of artificial neural network, which is usually used for solving supervised learning tasks of classification and regression.

The MLP consists of one or more layers of artificial neurons, also known as perceptrons, which are connected in a feedforward manner. Each neuron in the MLP takes a set of inputs which are usually a 2D matrix of real number. Then neuron performs a linear transformation on input and applies a non-linear activation function to the result. Finally, it passes the output as new input to the next layer of neurons until the last layer. In the last layer, output will go through different types of function depending on the type of task, whether it’s a classification or regression task.

For classification tasks, the last layer of the MLP usually consists of a set of output neurons equal to the number of classes to be predicted. Each output neuron corresponds the probability that the input belongs to a class. To convert the outputs into probabilities, the softmax activation function is typically applied. The softmax function , where ***N*** equals to the number of classes and *j* represents each class. We can recognize that softmax function takes vector of real number and normalizes it to ***P***, whichbelongs to (0,1) and sum to 1 so it can represent a probability distribution.

For regression tasks, the last layer of the MLP usually consists of neurons with no activation function, so the output can take on any real number.

* 1. Training Process

The training process of a MLP model is the process of iteratively adjusting the parameters of each perceptron to minimize the difference between the predicted output and the true result.

The loss function is to measure the difference between the predicted output of the MLP and the true result. For classification tasks, the cross-entropy loss function is commonly used, and for regression tasks, the mean squared error loss function is commonly used.

The first step is to initialize MLP with random or specific initial values. By using these values MLP can make predictions on training dataset. Then we use optimization algorithm like gradient descent to adjust parameters of each perceptron and minimize the loss function.

The backpropagation algorithm is efficient to compute the gradient in neural network. By chain rule, it will compute the gradient one layer at a time using the result of former layer and iterate backward from the last layer to input layer.

1. Dataset Description

In this project we choose 2 different datasets on UCI machine learning repository, which are ***Breast Cancer Wisconsin (Diagnostic) Data Set*** and ***Parkinsons Telemonitoring Data Set[[1]](#footnote-1)***.

* The Breast Cancer Wisconsin (Diagnostic) Data Set (WDBC) is a classification task of 2 classes to predict diagnosis result. The dataset consists of 569 instances, 32 attributes and no missing attributes. We use 30 real-valued attributes as input features, and the categorical attribute as label. According to label, there are 357 instances belongs to “B” class and 212 instances belongs to “M” class.
* The Parkinsons Telemonitoring Data Set is a regression task to predict the motor and total UPDRS scores in dataset. The dataset consists of 5875 instances, 26 attributes and no missing attributes. We use 19 attributes of real number and integer as input features and 1 real-valued attributes as label.

1. Program Manual
   1. Example Usage

This project is programmed in Python 3.

Use this command to run train and test program:

$ python main.py --input\_data=./data/Parkinsons.data --num\_units=32,16 \

--loss\_func=square\_loss --activation\_func=sigmoid --batch\_size=16 \

--num\_epochs=50 --learning\_rate=1e-2 --momentum=0.9 \

--l2\_norm=1e-3 --test\_ratio=0.2

$ python main.py --input\_data=./data/BreastCancer.data --num\_units=32,16,8 \

--loss\_func=log\_binary\_loss --activation\_func=sigmoid --batch\_size=16 \

--num\_epochs=50 --learning\_rate=1e-2 --momentum=0.9 \

--l2\_norm=1e-3 --test\_ratio=0.2

* 1. Hyperparameters
* --input\_data : Dataset filename, string, **required**.
* --num\_units : The size of hidden layers, integers separated by ‘,’ character, **required**.
* --loss\_func : Loss function, “square\_loss” and “log\_binary\_loss” available, “square\_loss” for regression tasks and “log\_binary\_loss” for classification tasks, **required**.
* --activation\_func : The activation function of perceptrons, “sigmoid” and “tanh” is available, “sigmoid” by default.
* --batch\_size : Batch size, integer, 16 by default
* --num\_epochs : The number of training epochs, integer, 50 by default.
* --learning\_rate : Learning rate, float, 0.01 by default.
* --momentum : Momentum of optimizer, float, 0.9 by default.
* --l2\_norm : L2 normalization coefficient, float, 0.001 by default.
* --test\_ratio : The ratio we split test dataset from entire dataset, float, 0.2 by default.

1. Experiments
   1. Activation Function

图片包含 图形用户界面

描述已自动生成图表, 折线图

描述已自动生成

Left : Classification task (WDBC); Right: Regression task (Parkinsons)

1. Result Analysis
2. Future Work

1. A Tsanas, MA Little, PE McSharry, LO Ramig (2009), 'Accurate telemonitoring of Parkinson’s disease progression by non-invasive speech tests', IEEE Transactions on Biomedical Engineering (to appear) [↑](#footnote-ref-1)