PATTERN RECOGNITION: Assignment #2

2031566 Yang Han
hanyang_sh@tongji.edu.cn

School of Software Engineering, Tongji University — November 12, 2020

Introduction

In this assignment, I will implement neural network and evaluate it on Seeds Data Set [1] and Iris Data Set [2].

1 Dataset Introduction

1.1 Seeds Dataset

The dataset belongs to three different varieties of wheat and is consisted of seven geometric parameters, 1. area A, 2. perimeter P, 3. compactness C 4. length of kernel, 5. width of kernel, 6. asymmetry coefficient 7. length of kernel groove. All of these parameters were real-valued continuous.

1.2 Iris Dataset

Iris Dataset is perhaps the best known database to be found in the pattern recognition literature, The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

2 Data Preprocess

As I show in the $preprocess.ipynb$ file, I mainly do three things:

1. Read the raw file content, cast the str type to float type save save them to csv format.
2. Map categories to ordered numbers(eg. 0,1,2...).
3. In order to eliminate the dimensional influence between indicators, data standardization is required to solve the comparability between data indicators.

3 Program Modules

3.1 Theory

The main part of error backpropagate algorithm is how to update the weight, as the silde shows that:

$$\Delta w_{ij} = \eta g_i b_h$$

$$\Delta \theta_j = -\eta g_i$$

$$\Delta v_{ij} = \eta e_h x_i$$

$$\Delta \gamma_h = -\eta e_h$$

[1] http://archive.ics.uci.edu/ml/datasets/seeds
[2] http://archive.ics.uci.edu/ml/datasets/Iris

And g_j and e_h can be denoted by the following formula:

$$g_j = \tilde{y}_j^k(1-\tilde{y}_j^k)(y_j^k - \tilde{y}_j^k)$$

$$e_h = b_h(1-b_h)\sum_{j=1}^{l} w_{hj}g_j$$

3.2 Code

In the part, i design a class named **NeuralNetwork** which accepted four parameters:n_inputs denotes the number of input, $n_hiddens$denotes the number of neurons in hidden layer, $n_outputs$denotes the number of output, lrdenotes the learning rate and have a default value 0.1.
This class mainly has a attribute named **network** which store the weights, bias, every neuron's output and the value of g_i and b_h mentiond above.
I implement the interface in Pytorch way, it also mainly has four methods. 1.**forward** is to calculate the output of a neuron. 2.**error_backpropagate** is the main error backpropagate algorithm, 3.**step** update the weight and bias stored in the **network** attribute. 4.**criterion** calculate the loss.

4 Results

I use **kFold Cross Validation** to get a more normal result.I use 5_fold in this experiment, In the seeds dataset($n_inputs = 7, n_hiddens = 5, n_outputs = 3$) with epochs=100, I get a result:

Command Line

```
seeds dataset, k_fold: 1, accurancy: 88.57142857142857%
seeds dataset, k_fold: 2, accurancy: 86.11111111111111%
seeds dataset, k_fold: 3, accurancy: 86.48648648648648%
seeds dataset, k_fold: 4, accurancy: 90.9090909090909%
seeds dataset, k_fold: 5, accurancy: 97.05882352941177%
```

In the iris dataset($n_inputs = 4, n_hiddens = 5, n_outputs = 3$) with epochs=100, I get a result:

Command Line

```
iris dataset, k_fold: 1, accurancy: 100.0%
iris dataset, k_fold: 2, accurancy: 100.0%
iris dataset, k_fold: 3, accurancy: 100.0%
iris dataset, k_fold: 4, accurancy: 100.0%
iris dataset, k_fold: 5, accurancy: 100.0%
```

5 Limitations and Inprovements

The main limitation is the number of layers of neural network is 2, I think a neural network with deeper depth will have better perfomance, but it's difficult to calculate derivation manually, we can use compution graph with Pytorch or Tensorflow to get the derivation quickly and update the weights. And the width of the hidden layer is also a limitation, neural network with wider width will also have better accurancy, but the seeds dataset and iris dataset is simple, and we can get good results approximately at 90% and 100%.