Assignment 2

auth: Qian yu 1831582

Dataset Chosen & Main Characteristics

The two datasets 'Iris' and 'Wine' are chosen for experiment.

Iris

Iris dataset has 150 records totally, divided into three types or iris —— I. setosa, I. versicolor and I. virginica, 50 records for each and this dataset has 4 dimensions —— Sepal length, Sepal width, Petal length and Petal width with the type of float. This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day.

Download Link: http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

Wine

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. Wine dataset has 178 records totally, divided into three kinds and this dataset has 13 dimensions —— Alcohol, Malic acid, Ash, Alcalinity of ash, Magnesium, Total phenols, Flavanoids, Nonflavanoid phenols, Proanthocyanins, Color intensity, Hue, OD280/OD315 of diluted wines and Proline.

Download Link: http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data

Running Example

There are two exaples in the `/src` , using iris dataset and wine dataset. The dependences are as below:

- Python 3
- Numpy

You can run the iris example by running the below command in your terminal to start the example:

```
python iris_example.py
```

You can run the wine example by running the below command in you terminal to start the example:

```
python wine_example.py
```

Data Preprocessing Operations

Before training, data preprocessing operations should be finished firstly for fitting the need of train module and for a better result.

Iris

The type of dataset's label is string, which represent the type of iris it belongs to in natural English. Change the type of label from string to another usable type is necessary. For classification problem, the string type of label is unacceptable. So the first work to do is change the type of label from value to boolean. For the three kind of iris, an array with the length of 3 is defined to describe the kind information of each record ——— the value '1' of label[i] means that this record belongs to the i-kind iris, while '0' means not.

Wine

Different dimensions has different range in wine datasets. In this case, it spends more time to calculate the weights to fit the different range of dimensions. To Avoid, a normalization operations for dataset is necessary. For this dataset, each value for the same dimension should minus the minimum value in the same dimension firstly and then divided by the difference value bewteen maximum and minimum, showing as code:

```
X = (X - np.min(X, axis=0)) / (np.max(X, axis=0) - np.min(X, axis=0))
```

And as the iris dataset, the value type of label is unacceptable for classification problem. So the first work to do is change the type of label from value to boolean. For the three kind of wine, an array with the length of 3 is defined to describe the kind information of each record ——— the value '1' of label[i] means that this record belongs to the i-kind wine, while '0' means not.

Modules Designing

The implement of netural network has three layer: NeuralNetwork, FullConnectedLayer and Activator.

NeuralNetwork

The NeuralNetwork has only one class with the same name. NeuralNetwork class works as a system class, controling the working flow. The simple explanation of the methos in NeuralNetwork class is as below:

- **init**:
  `def __init__(self, layer_nodes):`
  The initializing method has just one parameter ——— layernodes, which defined the nodes of each layers. This method will initialize the member variable and create the layers by calling FullConnectedLayer's init method.
- init:
  `def init(self):`
  This method will reset the train loss array and evalu loss array.
- predict:
  `def predict(self, x):`
- __test:
  `def __test(self, test_data_X, test_data_Y):`

This method work as evaluting the model. The input is a batch of evalution dataset and gives the accuracy and loss as result.

- train:
  `def train(self, X, Y, epochs, batch_size, learning_rate=0.01, test_rate=0.05):`
  Train method is the core of this class, which is the enterance of training. The parameters are the Input array, Label, epochs, batch size, learning rate and test rate.
- __train_batch:
  `def __train_batch(self, batch_X, batch_Y, learning_rate):`
  Train batch method will predict the input array firstly (forward) and then calculate the cross entropy loss. After, it will calculate the gradient (backward) and finally update the weight of the neural network.
- calc_cross_entropy_loss:
  `def calc_cross_entropy_loss(self, batch_Y, output_array):`
  This method will calculate the cross entropy loss with the output array and the true label of input array.
- calc_gradient:
  `def calc_gradient(self, label):`
  This method will calculate work as the backward we know. It calculates the gradient.
- update_weight:
  `def update_weight(self, rate):`
  This method will update all weights in the neural network by the result of calc_gradient method.
- show_diagram:
  `def show_diagram(self):`
  This method will show the diagram of train loss and evaluation loss.

FullConnectedLayer

FullConnectedLayer class is the implement of the full connected layer in neural network.

- **init**:
  `def __init__(self, shape, activator):`
  This method will initialize the member variables and get its activator.
- forward:
  `def forward(self, input_array):`
  This method will get and store the input array and the calculate the output after this layer by call its acivator's forward method.
- backward:
  `def backward(self, delta):`
  This method will get the delta from its next layer and calculate the delta for its previous layer and the value of weights to be update in this layer.
- update:
  `def update(self, learning_rate):`
  This method will update the weights of this layer with the result of backward method.

Activator

Activator class is the implement of activator in neural network and this demo provides two common activator —— Sigmoid and Softmax.

- forward:
  `def forward(k):`

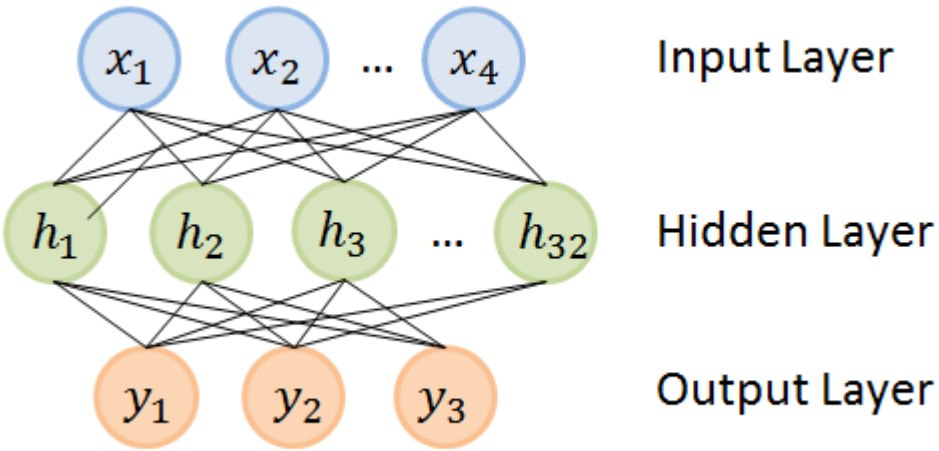
This method will calculate the value for forward step.

- backward:
  `def backward(k):`
  This methor will calculate the value for backward step.

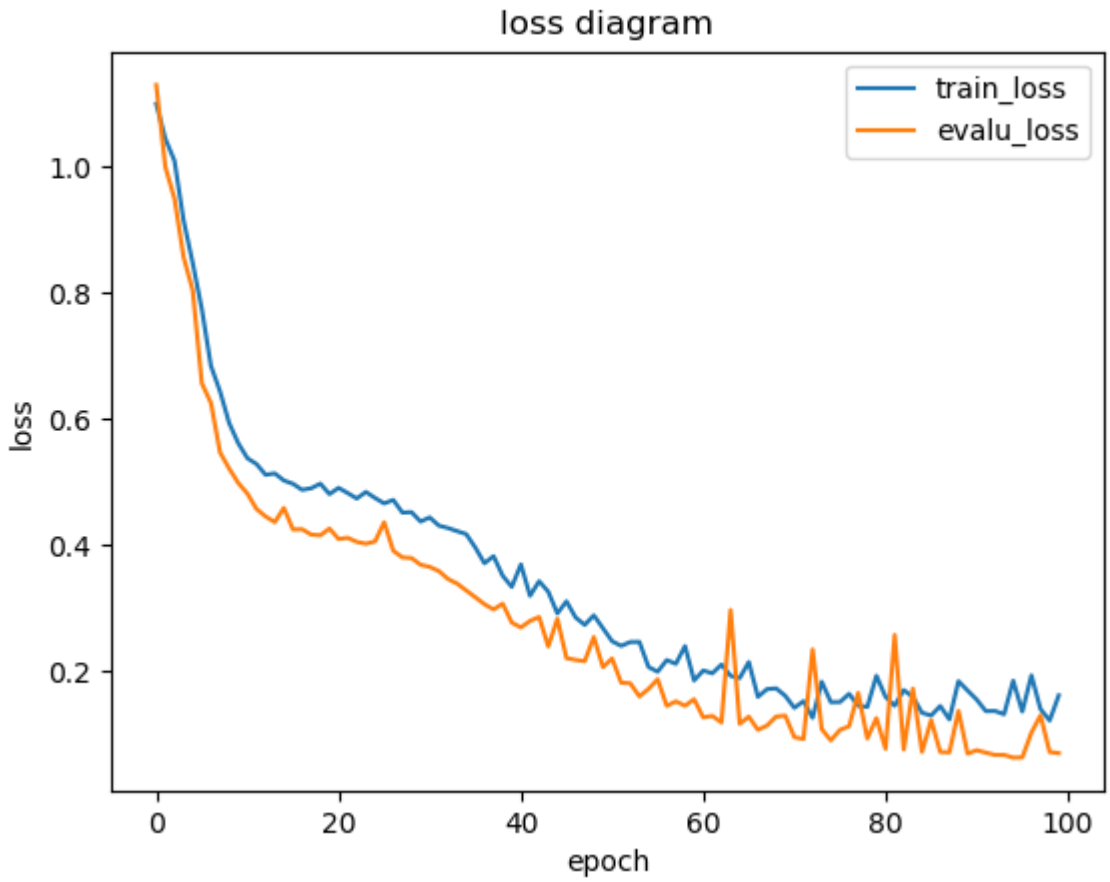
Prediction results & Interpretation

Iris

The neural network for iris dataset has one hidden layer —— a full connected layer with 32 nodes and has a output layer with 3 nodes.

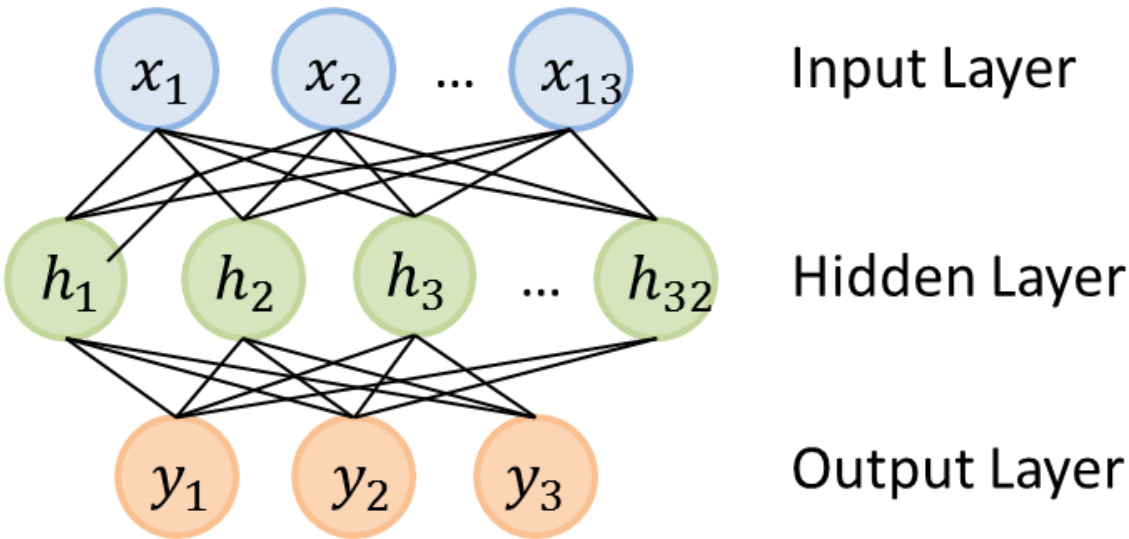


The learning rate is 0.01 and the evaluation dataset has 30 records (20%). After 239 epochs, the loss converges to 0.077416. The diagram of training loss and evaluation loss is as below:

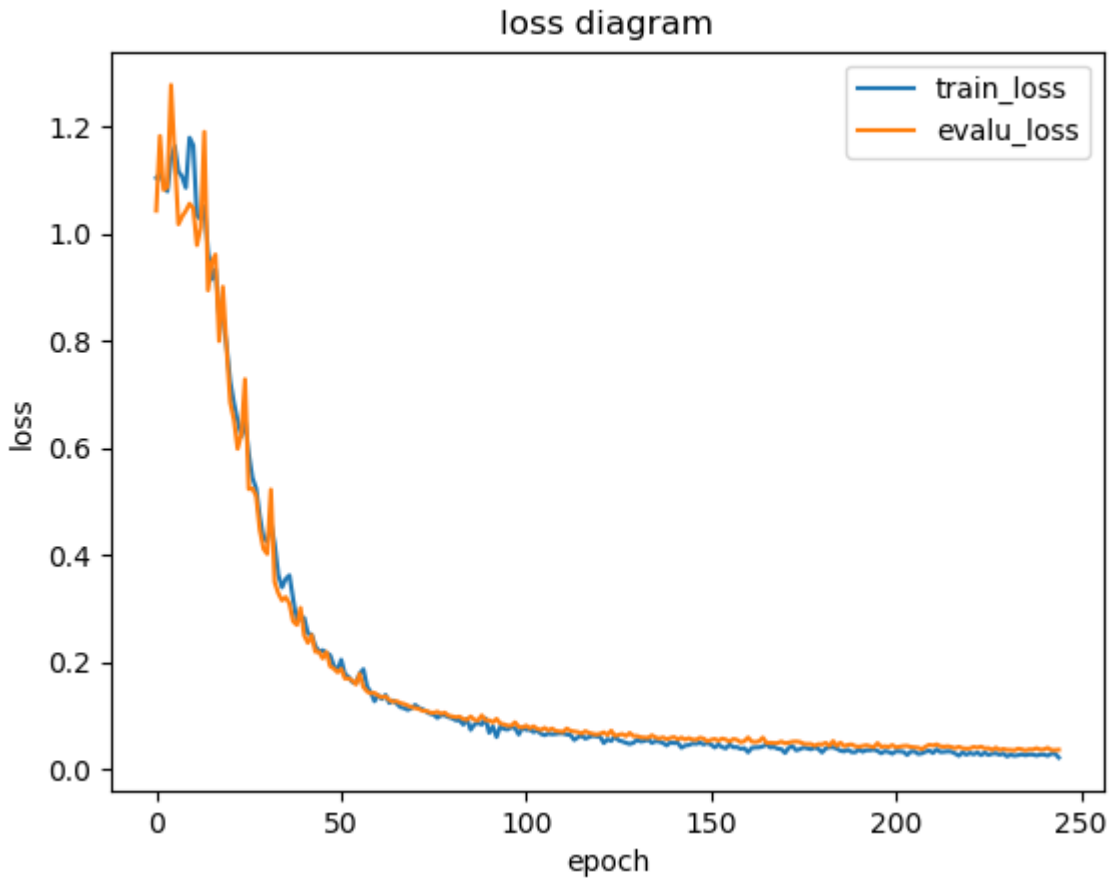


Wine

The neural network for wine dataset has one hidden layer —— a full connected layer with 32 nodes and has a output layer with 3 nodes.



The learning rate is 0.01 and the evaluation dataset has 30 records (20%). After 226 epochs, the loss converges to 0.029227. The diagram of training loss and evaluation loss is as below:



Limitations & Possible Improvements

While expering, I find that this model is affected by the distribution of dataset. In case of training a dataset with greatly different counts between each kind , the result will be bad with the repeated most often kind having the most possibility.

Another problem is that if the hidden layer has too many nodes, it will spend too long time for training. So in other case like training image dataset, we can use cnn instead.