Assignment 1

auth: Qian yu 1831582

Dataset:

The demo of the logistic regression with newton method uses the iris dataset, which 150 records divided into three types or iris —— *I. setosa*, *I. versicolor* and *I. virginica*, in 4 dimensions —— *Sepal length*, *Sepal width*, *Petal length* and *Petal width*. For a simple example, the records of *I. setosa* and *I. versicolor* are chosen. The data practically used has 100 records, 50 for *I. setosa* and 50 for *I. versicolor* and has just two dimension —— *Sepal length* and *Sepal width*.

Preprocessing Operations:

I used logistic regression with newton method to implement the binary classification of dataset. The first thing to do is preprocessing operations.

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. Binary classification need to find that if the data to be predicted belong to the presupposed kind or not —— in another words —— true or false. So I define that *Iris-versicolor* means *1* which represent *true* while *Iris-setosa* means 0 which represent *false*. After this operation, the type of label is changed into *integer, only having value 0 and 1.

Dimensions and label of the same record is in the same line, split by comma sign. To store them in matrix, I need to split the string data by comma sign first. After split, the first two elements are chosen for dimensions while the last one for label. All above records are stored in a python-list by order. The last step of preprocessing operations is to change them into matrix.

Modules Design:

The implement of logistic regression with newton method is in logistic_regression.py file. A class called *LogisticRegression* is defined, which contains the method of data input, preprocessing operations, logistic regression and draw diagram. By the LogisticRegression class, you can easily achieve the logistic regression from data inputing to diagram drawing.

The simple explanation of the methods in LogisticRegression class is as below:

- init：`def __init__(self, label_0, label_1)` The initializing method which has two parameters —— label_0 and label_1, which are the value of labels in you dataset. The value equal label_0 in your dataset label value while be changed into 0 while the label_1 1.

```
def __init__(self, label_0, label_1):
    self.__LABEL_0 = label_0
    self.__LABEL_1 = label_1
```

- append_data：`def __append_data(self, data_line)` Append data is a private method of LogisticRegression. This method defines the rule of data processing for one record. The acceptable type of input data is csv. Dimensions and label of the same record is in the same line, split by comma sign. This method splits the record by comma sign and then stores the elements to be used in predetermined format.

```
def __append_data(self, data_line):
    split = data_line.split(',')
    this_label = split[-1:][0][:-1]
    split = list(map(float, split[:-1]))
    if this_label == self.__LABEL_0:
        self.__label.append([0])
        self.__figure_x_label[0][0].append(split[0])
        self.__figure_x_label[0][1].append(split[1])
    elif this_label == self.__LABEL_1:
        self.__label.append([1])
        self.__figure_x_label[1][0].append(split[0])
        self.__figure_x_label[1][1].append(split[1])
    self.__x_range[0] = min(self.__x_range[0], split[0])
    self.__x_range[1] = max(self.__x_range[1], split[0])
    self.__data.append(split[:2])
```

- set_data_from_file：`def set_data_from_file(self, path)` This method will read date file from the path parameter and call the append_data method to handle the record and then, change the data type into matrix.

```
def set_data_from_file(self, path):
    input_file = open(path, 'r+')
    for line in input_file.readlines():
        self.__append_data(line)
    input_file.close()
    self.__data = array(self.__data, dtype='float64')
    self.__data = mat(insert(self.__data, self.__data.shape[1], values=1, axis=1))
    self.__label = array(self.__label)
    self.__rows, self.__cols = self.__data.shape[0], self.__data.shape[1]
```

- calculate：`def calculate(self)` Calculate method is the core of the LogisticRegression class, which contain the implement of logistic regression algorithm with newton method.

```
def calculate(self):
    while True:
        self.__p = 1.0 / (1 + exp(-(self.__data * self.__beta)))
        self.__theta_1 = multiply(1.0 / self.__rows, self.__data.T * (self.__p -
self.__label)) # (3, 1)
        self.__theta_2 = multiply(1.0 / self.__rows, self.__data.T *
mat(diag(multiply(self.__p, (1-self.__p)).T.getA()[0])) * self.__data) # (3, 3)
        self.__beta = self.__beta - self.__theta_2.I * self.__theta_1
        if linalg.norm(self.__theta_1) < 0.000001:
            break
    self.__beta = self.__beta.T.getA()[0]
```

Calculation method is in a while cycle which will exit when the first-order derivative is less than 1e-6. In the algorithm I calculate the first-order derivative and the second-order derivative firstly and then calculate the beta. On breaking from the while cycle, it return the beta in the type of python-list.

- show：`def show(self, c0='r', c1='b')` The show method will draw a simple diagram for showing the result and input data at the same time.

```
def show(self, c0='r', c1='b'):
    plt.figure(1)
    self.__y_range.append(-(self.__beta[2] + self.__beta[0] * self.__x_range[0]) /
self.__beta[1])
    self.__y_range.append(-(self.__beta[2] + self.__beta[0] * self.__x_range[1]) /
self.__beta[1])
    plt.plot(self.__x_range, self.__y_range)
    plt.scatter(self.__figure_x_label[0][0], self.__figure_x_label[0][1], c=c0)
    plt.scatter(self.__figure_x_label[1][0], self.__figure_x_label[1][1], c=c1)
    plt.show()
```

Example:

```
# A simple example

import logistic_regression as lr

if __name__ == '__main__':

    # Change LABEL_NAME_0 ans LABEL_NAME_1 into the value of label of your dataset,
like'Iris-setosa' or 'Iris-versicolor'.
    logisic_regression = lr.LogisticRegression(
LABEL_NAME_0, LABEL_NAME_1)

    # Change DATASET_PATH into you path of dataset 'iris.data'.
    # After this step, the data will be loaded and initialized.
    logisic_regression.set_data_from_file(DATASET_PATH)
```

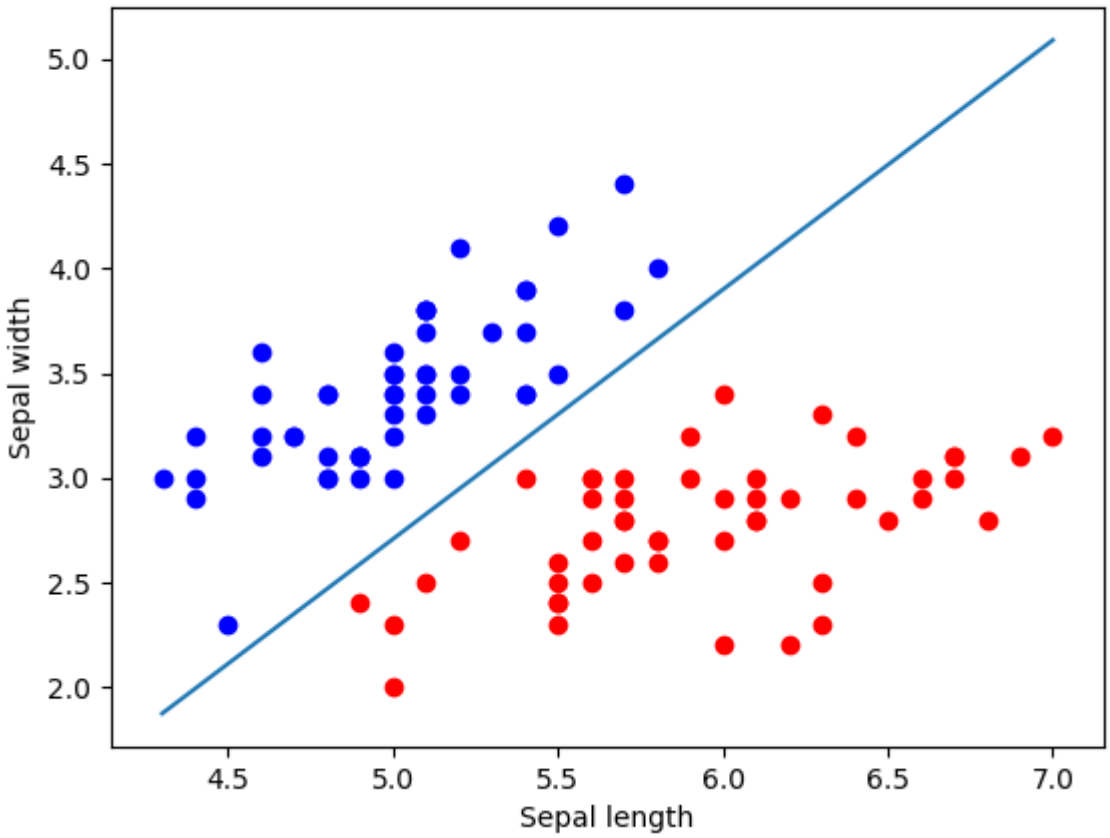
```
    # The calculate method is the implement of logistic regression with newton method.
    logisic_regression.calculate()

    # By this step, it can generate a simple diagram of the labels and vectors of input
data and it can draw a line which represent the result of logistic regression.
    logisic_regression.show()
```

Or you can run the example file in the project directly, using iris dataset. Before running, you should confirm that the dataset file —— 'iris.data' and the algorithm define file —— 'logistic_regression.py' are under the same path with the example file. Python3 environment, numpy package and matplotlib package are needed. You can run the below command in your terminal to start the example:

```
python example.py
```

Results：

After running the example, it will give the result of logistic regression by showing a diagram like this:



The blue points represent the records which labels are Iris-setosa while the red points Iris-versicolor and the line represent the result of logistic regression. The result line implements the binary classification of dataset by logistic regression with newton method.

Future Work:

- There may be overfitting in logistic regression. To avoid this case, introduction of regularization into the loss function is needed.
- Newton method can lead to local optimization some times. In this case, some initial values for target parameters may effect.
- In case of deal with high-dimension dataset, dimension reduction like PCA and LDA should be used in data processing.