Binary classification using logistic regression #1-2-2

컴퓨터소프트웨어학과 2017029470 김종원

1. Environment settings & how to compile

- Mac OS (Monterey) M1 chip
- Python 3.10.3 (release March 16, 2022)

Before compiling apiori.py, python version 3 must be installed in your system.

```
wonnx@wonnx practice_1_2_2 % python3 binary_classification_2.py
```

- Execution file name: binary_classification_2.py

2. Summary of algorithm

Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. In this project, I used the most common logistic regression models a binary outcome; something that can take two values such as 0 and 1. Logistic regression models utilize a linear combination of an input datapoint to solve a binary classification problem. And by using sigmoid function as a activation function, the result can be expressed as a value between 0 and 1.

3. Detailed description of codes

```
26  def logistic_regression(X, Y):
27      global w, b
28      Z = np.dot(w.T, X) + b
29      A = 1 / (1 + np.exp(-Z))
30      A = np.clip(A, 1e-12, 1 - 1e-12)
31      dZ = A - Y
32      db = np.sum(dZ) / m
33      dw = np.dot(X, dZ.T) / m
34      w = w - alpha * dw
35      b = b - alpha * db
36
37  def testing_accuracy_and_cost(size, X, Y):
38      accuracy = 0
39      Z = np.dot(w.T, X) + b
40      A = 1 / (1 + np.exp(-Z))
41      A = np.clip(A, 1e-12, 1 - 1e-12)
42      for i in range (size):
43          if A[0,i] > 0.5 and Y[0,i] == 1: accuracy += 1
44          elif A[0,i] < 0.5 and Y[0,i] == 0: accuracy += 1
45      accuracy = accuracy / size * 100
46      cost = -np.mean(Y * np.log(A) + (1 - Y) * np.log(1 - A))
47      return accuracy, cost
```

In this practice 1-2-2, the input variable x is 1 dimension vector, so also w. X is an array of training data, in the form of (1, m). Therefore, the result of dot product of w and X is (1, m), and even though b is added, the shape of Z is also (1, m) by numpy broadcasting. The sigmoid function was used as the activation function,

and binary cross entropy function was used as the loss function. The reason for using the clip function in the A array is to prevent the log from becoming 0 or 1 when calculating the loss.

4. Testing result

- $w = 0.2$, $b = 0$, $\alpha = 0.001$

	m = 10, n = 1000, k = 5000	m = 100, n = 1000, k = 5000	m = 10000, n = 1000, k = 5000
Accuracy (m train samples)	**60.0**	**53.0**	**50.71**
Accuracy (n test samples)	**47.599999999999994**	**48.8**	**49.9**
	m = 10000, n = 1000, k = 10	m = 10000, n = 1000, k = 100	m = 1000, n = 1000, k = 5000
Accuracy (m train samples)	**50.29**	**50.160000000000004**	**50.4**
Accuracy (n test samples)	**49.9**	**49.7**	**49.6**

The alpha value was set to 0.0001, 0.01 and 0.1 respectively and the test was conducted, but the result did not change. (Table is omitted). However, when the number of iterations was increased, the following result appeared.

- **m = 10000, n = 1000, k = 300000**
  updated unknown value: w = -0.05476999, b = 9.72820568268307
  accuracy on 'm' training samples: 99.28
  cost on 'm' training samples: 0.0840494638381929
  accuracy on 'n' test samples: 99.9
  cost on 'n' test samples: 0.09405558320214806

5. empirical discussion

Overall, the reason why the accuracy of the m training set is slightly higher than the result of the n test set is because the network was fitted to the training samples. Since the method of generating the training data set was more complicated than in project 1, the accuracy was low when the number of iterations was small. However, if you increase the number of iterations like k=300000, the accuracy increases to more than 99%.

- Accuracy increases only when w and b are properly fitted.

If the initial value is set like w = -1 and b = 180, the accuracy is 100% even when k = 5000. In the above testing, the initial values were set as w = 0.2 and b = 0, so the accuracy was low when the number of iterations was low. In this case, even if testing was carried out by changing the alpha value to 0.01 and 0.1, the result was not different from the previous one.