**Binary classification using logistic regression #1**

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.



* Execution file name: binary\_classification.py

1. **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.

1. **Detailed description of codes**

텍스트, 모니터, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* Logistic regression function is a function that changes the value of w and b, which are unknown data using trained data samples. Since the number of trained data set is m, repeat the loop m times to find the gradients of w and b.
* When the result value a of the sigmoid function is less than 1e-12 and greater than 1- 1e-12, the value of a is specified to solve divided by zero error.

1. **Testing result**

|  |  |  |  |
| --- | --- | --- | --- |
|  | m = 10, n = 1000, k = 5000 | m = 100, n = 1000, k = 5000 | m = 10000, n = 1000, k = 5000 |
| Accuracy (m train samples) | **100** | **100** | **99.9** |
| Accuracy (n test samples) | **92.5** | **98.4** | **99.9** |
|  | m = 10000, n = 1000, k = 10 | m = 10000, n = 1000, k = 100 | m = 1000, n = 1000, k = 5000 |
| Accuracy (m train samples) | **95.26** | **96.25** | **99.9** |
| Accuracy (n test samples) | **94.6** | **96.2** | **99.7** |

* m = 10000, n = 1000, k = 5000

cost on ‘m’ train samples: 433.3693104927667

cost on ‘n’ test samples: 44.1537033661482

updated unknown value: w = [1.73345476 1.73264511], b = 0.045884994393021655

|  |  |  |  |
| --- | --- | --- | --- |
|  | m = 10, n = 1000, k = 5000 | m = 100, n = 1000, k = 5000 | m = 10000, n = 1000, k = 5000 |
| Accuracy (m train samples) | **100** | **100** | **99.89** |
| Accuracy (n test samples) | **92.9** | **98.6** | **99.7** |
|  | m = 10000, n = 1000, k = 10 | m = 10000, n = 1000, k = 100 | m = 1000, n = 1000, k = 5000 |
| Accuracy (m train samples) | **96.56** | **98.82** | **99.9** |
| Accuracy (n test samples) | **95.8** | **99.0** | **99.7** |

* m = 10000, n = 1000, k = 5000

cost on ‘m’ train samples: 204.49808227841493

cost on ‘n’ test samples: 23.34621894017071

updated unknown value: w = [3.74176203 3.7337733], b = 0.021207149389567564

1. **empirical discussion**

When the m,n, and k values were set the same and only the hyper parameter value was changed from 0.01 to 0.1, the cost value and the ‘b’ value were halved and the w value was doubled. Therefore, it was found that the initial value of was not appropriate, and after several more experiments, if the alpha value becomes too large (ex, , the ‘b’ value decreases and then increases as the w value increases. Nevertheless, I thought that the decrease in the cost value was because the initial values of w and b were set incorrectly.