(3) One classification algorithm that was implemented in this assignment was logistic regression, which was used to classify whether an article was either fake news or it was not, i.e. ‘real’ news. We chose logistic regression, because it is an appropriate algorithm to implement on the chosen dataset. Since the target variable of the dataset is binary, logistic regression can be used to predict the probabilities of each data point, and then categorise each data point into 0 or 1, depending on its probability (categorise it 0 if p(x) < 0.5 or 1 if p(x) ≥ 1). The algorithm was implemented using two methods of gradient descent: loss minimization and maximum likelihood.

1. Loss minimization with gradient descent (Logistic Regression)

See below the error on the test set taken directly from the Jupyter Notebook output in the ‘LogisticRegression.ipynb’ file (illustrated with the assistance of the scikit-learn library):

* **Confusion Matrix:** [1924 124] [70 1904]
* **Accuracy Score:** 0.9517652909000497
* **Report:**

**precision recall f1-score support**

0.0 0.96 0.94 0.95 2048

1.0 0.94 0.96 0.95 1974

**accuracy** 0.95 4022

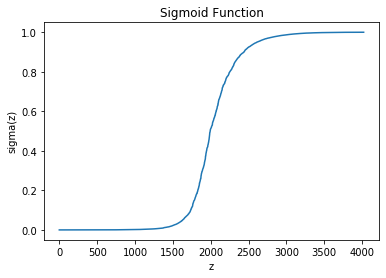
**macro avg** 0.95 0.95 0.95 4022

**weighted avg** 0.95 0.95 0.95 4022

We chose to use alpha = 0.5 to implement gradient descent. Increasing alpha by 0.1 slightly increases the accuracy of the classifying model, but it also slightly increases the time in which it takes for the gradient descent function to finish running. We ran and timed the duration of the gradient descent on a test set (about 20% of the data) and we got the following results:

|  |  |  |
| --- | --- | --- |
| Alpha | Accuracy | Time (in seconds) |
| 0.1 | 0.946544 | 95.208203 |
| 0.2 | 0.948284 | 97.311399 |
| 0.3 | 0.949528 | 98.722599 |
| 0.4 | 0.951019 | 97.449176 |
| 0.5 | 0.951765 | 99.04105 |

As you can see, if we were to pick alpha in the interval (0.5, 1) we could possibly find an alpha that would yield a higher accuracy, but that would make the gradient descent take longer time to run. Clearly, alpha = 0.5 seemed to be the most optimal value to pick when scaled up to a much larger dataset. Below is the Sigmoid function produced by the gradient descent function implemented on the test set:



1. Maximum Likelihood Estimation with gradient descent (Logistic Regression)

See below the error on the test set taken directly from the Jupyter Notebook output in the ‘LogisticRegression.ipynb’ file (illustrated with the assistance of the scikit-learn library):

* **Confusion Matrix:** [1898 150]

[43 1931]

* **Accuracy Score:** 0.9520139234211835
* **Report:**

**precision recall f1-score support**

0.0 0.98 0.93 0.95 2048

1.0 0.93 0.98 0.95 1974

**accuracy** 0.95 4022

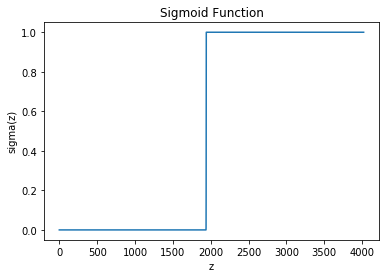
**macro avg** 0.95 0.95 0.95 4022

**weighted avg** 0.95 0.95 0.95 4022

Again, we chose to use alpha = 0.5 to implement gradient descent. As was the case with the loss minimization implementation of gradient descent, increasing alpha by 0.1 slightly increases the accuracy of the classifying model, but it also slightly increases the time in which it takes for the gradient descent function to finish running. Implementing gradient on the test set produces the following results:

|  |  |  |
| --- | --- | --- |
| Alpha | Accuracy | Time (in seconds) |
| 0.1 | 0.935853 | 130.59953 |
| 0.2 | 0.938339 | 133.4473 |
| 0.3 | 0.953008 | 138.459089 |
| 0.4 | 0.953257 | 138.862093 |
| 0.5 | 0.952014 | 139.329085 |

Similar to the loss minimization implementation of gradient descent, picking alpha in the interval (0.5, 1) could possibly yield a higher accuracy, but that would result in longer computation of the gradient descent. Clearly, alpha = 0.5 would be the most optimal value to pick when scaled up to a much larger dataset. Below is the Sigmoid function produced by the gradient descent function implemented on the test set:



References

1. Kaggle.com. 2020. *Logistic Regression From Scratch - Python*. [online] Available at: <https://www.kaggle.com/jeppbautista/logistic-regression-from-scratch-python> [Accessed 17 April 2020].
2. Bird, R., 2020. *Logistic Regression*. [online] Medium. Available at: <https://medium.com/greyatom/logistic-regression-89e496433063> [Accessed 18 April 2020].