DEMONSTRATE THE EQUIVALENT BETWEEN BINOMIAL LOGISTIC REGRESSION AND A SINGLE LAYER PERCEPTRON WITH CROSS-ENTROPY LOSS FUNCTION

BY

JOSEPH ITOPA, ABUBAKAR

DATE: 29th July, 2024

TABLE OF CONTENT

- **Introduction**
- **Loss functions**
- **Overview of Logistic Regression**
- **Gradient descent of Logistic Regression**
- **Overview of Single-layer Perceptron**
- **Gradient Descent for Single-layer Perceptron**
- **Python Implementation of Key Concepts**
- **Conclusion**
- **References**

1.0 INTRODUCTION

This report is aimed at demonstrating the equivalence between a single layer perceptron and binomial logistic regression with a cross-entropy loss function. The two techniques share similar underlying mathematical methods. In this report, detailed analogy and implementation in python programming language is demonstrated.

1.1.0 LOSS FUNCTIONS

A loss function helps to measure the performance of a model by evaluating how well a model predicts an expected outcome. Cross-entropy is a type of loss function which is utilized in logistic regression to measure the difference between the true labels and the predicted labels.

The average cross entropy loss function is given in the equation below:

$$\mathbb{L} = -\frac{1}{\mathbb{N}}\sum_{i=1}^{\mathbb{N}}[y_i \log\left(\hat{y}_i\right) + \left(1 - y_i \log\left(1 - \hat{y}_i\right)\right)]$$

$y_i -$ *true label*

$\hat{y}_i -$ *predicted probability*

$\mathbb{L}$ $-$ *Average cross entropy*

$\mathbb{N}$ $-$ *Number of data instances*

1.1.1 OVERVIEW OF LOGISTIC REGRESSION

There are several statistical methods of which Logistic regression is one of the statistical methods, and it is utilized for binary classification. Binary classification basically involves identifying two different categories or classes. Furthermore, logistic regression basically predicts the probability that an input belongs to one of two categories.

The mathematical model for logistic regression is:

$$\hat{y} = \sigma(z) = \sigma(\omega^{\mathbb{T}}x + b) = \frac{1}{1 + e^{-(\omega^{\mathbb{T}}x + b)}}$$

$\omega - \textit{weight of the vector}$

$\sigma(z) - \textit{sigmoid function}$

$b - \textit{bias term}$

$x - \textit{input feature vector}$

$\hat{y} - \textit{predicted probability}$

1.1.2 GRADIENT DESCENT FOR LOGISTIC REGRESSION

To improve the performance of logistic regression, the cross entropy loss must be minimized, and this is achieved through gradient descent. Gradient descent in this case tends to control the loss function with respect to the weight(w), and bias(b). These are shown in the equation below:

The gradient of the loss with respect to w:

$$\frac{\partial \mathbb{L}}{\partial w} = -\frac{1}{\mathbb{N}} \sum_{i=1}^{\mathbb{N}} \left(\hat{y}_i - y_i\right) x_i$$

The gradient of the loss with respect to b:

$$\frac{\partial \mathbb{L}}{\partial b} = -\frac{1}{\mathbb{N}} \sum_{i=1}^{\mathbb{N}} \left(\hat{y}_i - y_i\right)$$

1.2.1 OVERVIEW OF SINGLE-LAYER PERCEPTRON

By definition, a single-layer perceptron is a neural network that is utilized in binary classification. It possesses both input and output layers, and an activation function. A single-layer perceptron uses sigmoid function to work in the similitude of a logistic regression. Single-layer perceptron is mathematically expressed as:

$$\hat{y} = \sigma(z) = \sigma(\omega^{\mathbb{T}}x + b)$$

$$\omega - weight\ of\ the\ vector$$

$$\sigma(z) - sigmoid\ function$$

$$b - bias\ term$$

$$x - input\ feature\ vector$$

$$\hat{y} - predicted\ probability$$

1.2.2 GRADIENT DESCENT FOR SINGLE-LAYER PERCEPTRON

For the single-layer perceptron, the approach is similar and its outlined below:

i) Define the model:

$$\hat{y} = \sigma(z) = \sigma(\omega^{\mathbb{T}}x + b)$$

ii) Define the loss function:

$$\mathbb{L} = -\frac{1}{\mathbb{N}}\sum_{i=1}^{\mathbb{N}}[y_i \log\left(\hat{y}_i\right) + \left(1 - y_i \log\left(1 - \hat{y}_i\right)\right)]$$

iii) Model training and evaluation:

$$\frac{\partial \mathbb{L}}{\partial w} = -\frac{1}{\mathbb{N}}\sum_{i=1}^{\mathbb{N}}\left(\hat{y}_i - y_i\right)x_i$$

$$\frac{\partial \mathbb{L}}{\partial b} = -\frac{1}{\mathbb{N}}\sum_{i=1}^{\mathbb{N}}\left(\hat{y}_i - y_i\right)$$

1.3 PYTHON IMPLEMENTATION OF KEY CONCEPTS

Copy the links below to find the python implementation of the solutions:

i) Logistic regression:

https://github.com/JosephItopa/comparism-of-binary-classifiers/blob/main/logistic_regre
ssion.py

2) Single-layer perceptron:

https://github.com/JosephItopa/comparism-of-binary-classifiers/blob/main/single_layer_
perceptron.py

In the implementation, you will notice that we are using log_loss from sklearn and BCE_loss from pytorch, mathematically they are both the same.

1.4 CONCLUSION

The two approaches are equivalent to each other. They are both used for classification tasks. They internally engage the cross entropy loss function, and with gradient descent, both methods can be optimized by adjusting the loss function with respect to weight and biases. The only difference is that logistic regression is used in statistical context while single-layer perceptron is used in Neural Networks.

REFERENCES

- Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
- Scikit-Learn Documentation
- PyTorch Documentation