MATH 332 Fall 2023
Finals Project

Honor Code: I affirm I have adhered to the honor code in this assignment.

Logistic Regression

Logistic Regression is one of the most popular binary classification methods. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.
We consider a random variable y $\in \{0, 1\}$, with probability p, with

$$\text{P(y = 1) = p and P(y = 0) = 1 - p ; p = [0,1]}$$

Usually, we denote y = 1 as yes, whereas y = 0 as no. The probability for something to happen is p, which is modeled as a function of some explanatory variables u, which represents some relevant factors to the model we are predicting.
The logistic model has the form:

$$p = \frac{e^{a^T u+b}}{1 + e^{a^T u+b}}$$
$$\iff p = \frac{1}{1 + e^{-(a^T u+b)}}$$

where a $\in \mathbf{R}_n$ and b $\in \mathbf{R}$ are the model parameters that determine how the probability p varies as a function of the explanatory variable u. The equation above is often called "sigmoid function", which is a S-shaped curve and has the bound from 0 to 1.

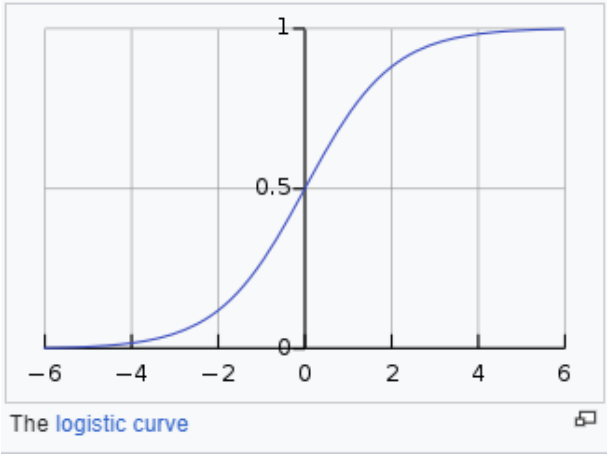


Figure 1: The Logistic Curve
Source: Wikipedia

In real-life problems, we are given some data consisting of a set of values of the explanatory variables $u_1, u_2, ..., u_m \in \mathbf{R}$, along with the corresponding outcomes $y_1, y_2, ..., y_m \in \{0, 1\}$. Our job is to find a maximum likelihood estimate of the model parameters $a \in \mathbf{R}_n$ and b $\in \mathbf{R}$. Finding an ML estimate of a and b is sometimes called logistic regression.
The log likelihood function then has the form:

$$l(a,b) = \sum_{i=1}^{n} y_i log(p_i) + (1\text{-}y_i)\log(1 - p_i)$$
$$\text{where } p_i = \frac{1}{1+e^{-(a^T u_i+b)}}$$

Therefore:

$$l(a,b) = \sum_{i=1}^{n} y_i log\Big(\frac{1}{1+e^{-(a^T u_i+b)}}\Big) + (1\text{-}y_i)\log\Big(1 - \frac{1}{1+e^{-(a^T u_i+b)}}\Big)$$
$$l(a,b) = \sum_{i=1}^{n} y_i\Big(log(\frac{1}{1+e^{-(a^T u_i+b)}}) - log(\frac{e^{-(a^T u_i+b)}}{1+e^{-(a^T u_i+b)}})\Big) + log\Big(\frac{e^{-(a^T u_i+b)}}{1+e^{-(a^T u_i+b)}}\Big)$$
$$l(a,b) = \sum_{i=1}^{n} y_i\Big(log(\frac{1}{e^{-(a^T u_i+b)}})\Big) + log\Big(\frac{1}{1+e^{a^T u_i+b}}\Big)$$
$$l(a,b) = \sum_{i=1}^{n} y_i\Big(log(e^{a^T u_i+b})\Big) + log\Big(\frac{1}{1+e^{a^T u_i+b}}\Big)$$
$$l(a,b) = \sum_{i=1}^{n} y_i\Big(a^T u_i + b\Big) \text{ - } log\Big(1 + e^{a^T u_i+b}\Big)$$

Since l is a concave function of a and b, the logistic regression problem can be solved as a convex optimization problem.
Our problem is:
maximizing $l(a, b)$
Which is equivalent to: minimizing $-l(a, b)$. Let $a^T u_i + b = \hat{\beta}_i$, we can rewrite our equations as follow:

$$\text{minimizing -}l(a,b) = \sum_{i=1}^{n} -y_i\Big(\hat{\beta}_i\Big) + log\Big(1 + e^{\hat{\beta}_i}\Big)$$

The mean of this equation, which is $\frac{-l(a,b)}{n}$, is called the log-loss function. Lower log-loss means better prediction. We will use gradient descent to minimize this equations (denote $L(a, b)$)
Taking the gradient of the log loss with respective to a:

$$\frac{\partial L(a,b)}{\partial a} = \frac{1}{n}\ (-u^T y + u^T \frac{e^{\hat{\beta}_i}}{1+e^{\hat{\beta}_i}})$$
$$\frac{\partial L(a,b)}{\partial a} = \frac{1}{n}\ (-u^T y + u^T p)$$
$$\Rightarrow \frac{\partial L(a,b)}{\partial a} = \frac{1}{n}\ (p - y)\ u^T$$

And the gradient of with respect to b

$$\frac{\partial L(a,b)}{\partial b} = \frac{1}{n}\ (-y + \frac{e^{\hat{\beta}_i}}{1+e^{\hat{\beta}_i}})$$
$$\frac{\partial L(a,b)}{\partial b} = \frac{1}{n}\ (-y + p)$$
$$\Rightarrow \frac{\partial L(a,b)}{\partial b} = \frac{1}{n}\ (p - y)$$

The algorithm :

Algorithm 1 The Steepest Descent Algorithm:

SGD($x, y, epochs, learningrate$) :
Set a $\rightarrow$ zero matrices, b $\rightarrow$ 0
for i = 1 to epochs **do**
  for each training tuple (x_i, y_i) **do**
    Compute p_i using sigmoid function.
    Compute $L(a, b)$ (log loss functon)
  Compute the gradient for a and b
  Apply generic algorithm to a and b (Move to some directions)

The formula for applying generic algorithm is:

$$\text{a} = \text{a - } \mu \nabla_a L(a, b)$$
$$\text{b} = \text{b - } \mu \nabla_b L(a, b)$$

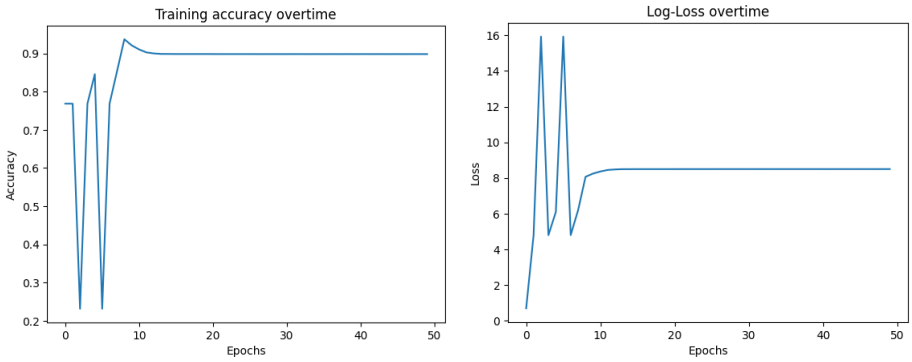
Where μ is the learning rate. Larger learning rate means that we are moving steeper in that direction. The optimal learning rate may be different base on different optimization technique, and for this I am setting it default to 0.1.

Testing and Graphing our optimization model

I have made use of Professor Adam Eck's CSCI 374 Machine Learning Fall 2022, Homework 2 as a baseline for my model. I am testing it with his two datasets: banknotes.csv and occupancy.csv as my training and testing data. I am also using one more sklearn built in dataset: "Diabetes dataset" for this visualization.

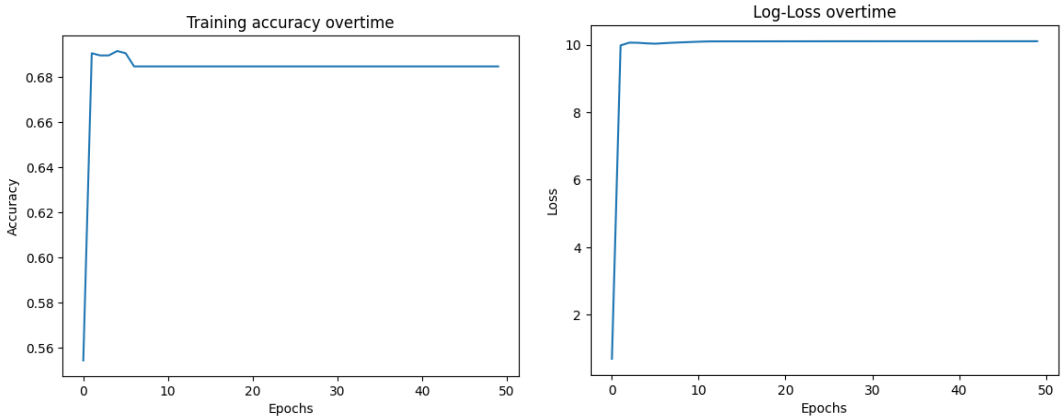
Procedure: I split 75% of the dataset for training, and 25% for testing. I use the same number of epochs = 50 and learning rate = 0.1.

Dataset 1: occupancy.csv



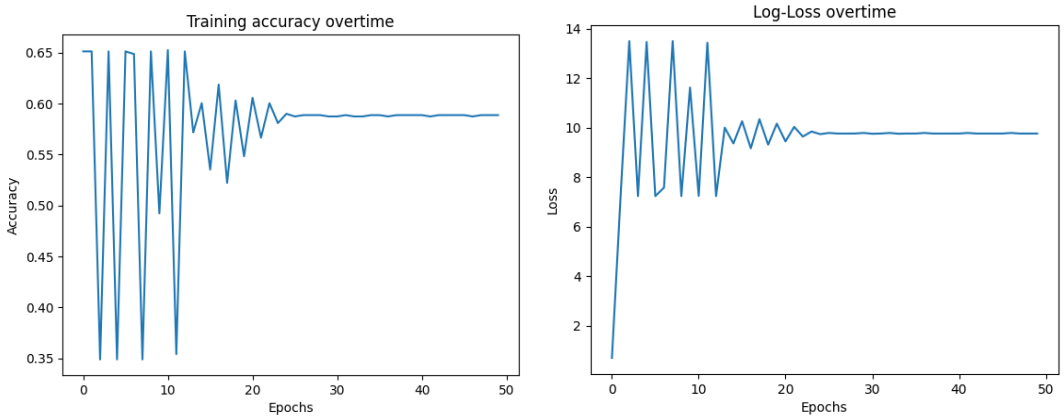
At around epoch 10, our model has been stablized. It minimizes log loss around 8 and achieve accuracy score of 90%. Testing on our test data, the model achieves accuracy score of 0.895.

Dataset 2: banknotes.csv



This dataset seems do not fit well with our model. It just achieve accuracy = 68%. Testing on our test dataset, it yields an accuracy score of 0.693.

Dataset 3: Diabetes dataset



This model takes longer to become stablized, at epoch 25, it can minimize the logloss function. But the accuracy score is not too high, just about 58%, and even when testing on the test dataset, it just give accuracy score of 0.6041.

Comparing to the sklearn Logistic Regression model, it performs worse in all three datasets. It might be that they use different solver and activation function. But for using gradient descent, it still gives a decent score.

Resources I used:
https://web.stanford.edu/ jurafsky/slp3/5.pdf
https://developer.ibm.com/articles/implementing-logistic-regression-from-scratch-in-python/
https://www.cs.oberlin.edu/ aeck/Fall2022/CSCI374/Assignments/HW2/index.html
https://scikit-learn.org/stable/datasets/toy_dataset.html
And section 7.1 of Boyd and Vandenberghe Convex Optimization https://web.stanford.edu/ boyd/cvxbook/