Linear Classifiers: Logistic Regression and Adaline

Jieyi Chen

October 25, 2020

1. Introduction

This project explores the application of two linear classifiers and discusses their performances on the binary and multi-class classification datasets. The idea of linear classifier is trying to separate the feature space into K linearly separable regions with hyperplanes, where K is the number of classes in the dataset. Both of these classifiers use the concept of gradient descent to find the optimal weights that minimize the cost function. Logistic regression is a statistical model that continuously updates the regression coefficients/weights in the linear combination of the explanatory variables based on the feedback from the gradient descent process. Adaline is a single neural network with nodes where the input nodes are the attributes and the output node is the target variable. It iteratively updates the weight vectors that connect the layers until the cost function is minimized. The difference between the Adaline and perceptron neural network is that the standard perceptron includes the activation function in the weight updating process. Logistic regression is designed for binary classification, but we have used one-vs-all strategy to handle dataset with multi-class. The same method is also applied to Adaline. We hypothesize that Adaline outperforms the logistic regression but with faster time complexity because the logistic regression includes the sigmoid activation step during the training process, which can lead to overfitting and longer computing time. To test this hypothesis, we compare the classification performance of the logistic regression and Adaline algorithms on several datasets from the UCI repository [1].

The rest of the paper is organized as follows. Section 2 describes the algorithms that are implemented. Section 3 discusses the experimental approach. Section 4 presents the results of the experiments. Section 5 discusses the results. Section 6 concludes.

2. Algorithm Implementation:

2.1 Logistic Regression Algorithm:

Logistic regression is a classification algorithm that predicts binary outcome using the concept of regression analysis and sigmoid function.

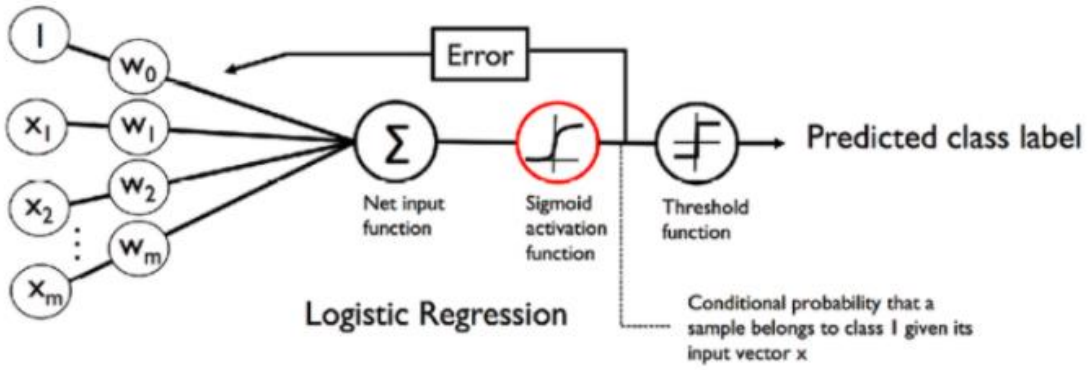

Figure 1: Logistic Sigmoid Activation Function Representation [2]

2.1.1 Two-class Logistic Regression

1. Suppose we have two classes c = {0, 1}
2. Find the posterior probability P(c|x) using the sigmoid function where c is a discrete variable identifying the class and x is a vector of attributes or features

Sigmoid Function:

$$logit^{-1} = \frac{1}{1 + \mathrm{e}^{(-\mathrm{p})}} = \frac{e^{(-p)}}{1 + e^{(-p)}}$$

Posterior Probability:

$$p(c = 1|x) = \frac{e^{(w_0 + \sum_{i=1}^{d} w_i x_i)}}{1 + e^{(w_0 + \sum_{i=1}^{d} w_i x_i)}}$$

$$p(c = 0|x) = \frac{1}{1 + e^{(w_0 + \sum_{i=1}^{d} w_i x_i)}}$$

Choose c = 1 if $1 < \frac{p(c=1|x)}{p(c=0|x)}$

Choose c = 1 if $1 < \frac{\frac{e^{(w_0 + \sum_{i=1}^{d} w_i x_i)}}{1 + e^{(w_0 + \sum_{i=1}^{d} w_i x_i)}}}{\frac{1}{1 + e^{(w_0 + \sum_{i=1}^{d} w_i x_i)}}}$

2.1.2 Gradient Descent:

Gradient Descent is an optimization algorithm that finds the optimal parameters of the model with the goal of minimizing the loss function.

2.1.3 Cost Function - Cross Entropy:

Binary Classification:

$$-(ylog(p) + (1 - y)\log(1 - p))$$

Multi-class Classification:

$$-\sum_{c=1}^{C} y_{i,c} \log\left(p_{i,c}\right)$$

where C = total number of classes, i = observation i, c = class c

2.1.4 Algorithm:

1. Initialize a weight vector w_j with size equals to the number of attributes in the dataset + 1 and the values range from -0.01 and 0.01. We need to include a weight for the bias term.
2. Repeat the following steps until convergence:
    a. Initialize a weight change vector Δw_j with all 0 and its size equals to the number of attributes + 1
    b. For each training instance:

i. Calculate the weighted sum of the attribute values and pass the output to the sigmoid function

$$output = \sum_{j=1}^{d} w_j x_j \quad d = total\ number\ of\ attributes$$

$$prediced\ value = Sigmoid(output)$$

ii. Update the weight changed vector Δw_j with the gradient value returned from the cost function

$$gradient = (actual\ value - predicted\ value) * x_j \qquad j\ from\ 1\ to\ d$$

c. Multiply each value in the weight change vector Δw_j with the learning rate η and add this value to the weight vector w_j

$$w_j \longleftarrow w_j + \eta \Delta w_j$$

3. Convergence/Stop the algorithm until the following conditions are met:
   a. The change in the weight change vector Δw_j is less than a certain threshold
   b. Maximum iterations have reached

2.2 ADALINE Neural Network:

ADALINE (Adaptive Linear Element) is a single-layer neural net developed by Bernald Widrow and his graduate student Ted Hoff in 1960 [3]. This is the first neural network that uses the gradient descent as its training rule.

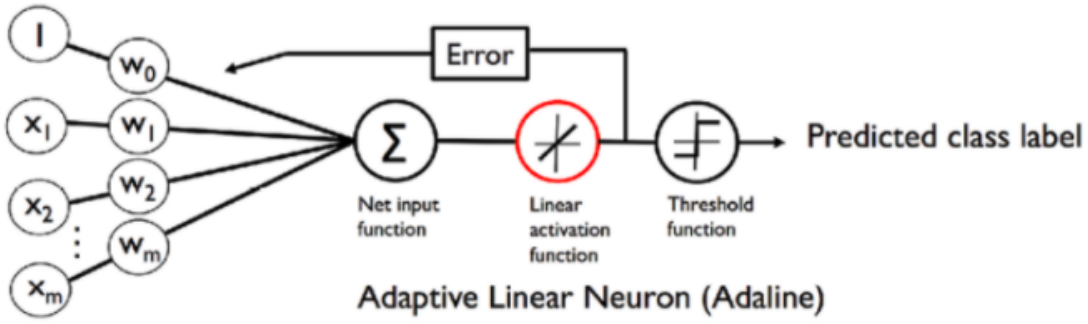

Figure 2: Adaline Linear Activation Function Representation [2]

2.2.1 Algorithm:

Parameters Definition:
W: weight vector, X: input vector, y: network output, d: desired output, $Err = (d - y)^2$
η: learning rate in the range (0, 1)

1. Initialize a weight vector w_j with its size equals to the number of attributes in the dataset + 1 and its values range from -0.01 and 0.01.
2. Repeat the following step a-e for k epochs
    a. Calculate the weighted sum of the attribute values using its initialized weights

$$u = \sum_{j=1}^{d+1} w_j x_j \quad d = total\ number\ of\ attributes$$

    b. Update the weight vector w_j with the gradient value returned from the cost function and the learning rate η

$$gradient = (d - y) * x_j$$
$$w_{(t+1)} \leftarrow w_t + \eta(d - y)x$$

    c. Use the updated weights as the initialized weights for the next instance in the dataset
    d. Repeat step a - c for all the instances in the training set
    e. Update the weight vectors by the accumulated weight values after step d is completed

$$w := w + \Delta w$$

3. Calculate the weighted sum using the optimal weights trained from step 2 and the data instance from the test set, compare it with the threshold of zero to make the classification prediction. The threshold is learned since it corresponds to the weight on the connection between the bias input and output. If the weighted sum is greater than 0, the algorithm assigns 1 to the data instance. Otherwise, it assigns 0.

2.3 Use Logistic Regression and Adaline for Multi-class classification:

One vs. all and One vs. one are the two types of multi-class classification techniques. One vs. all is a method that builds K linear discriminants for a dataset with K classes. One vs. one is a technique that builds $\frac{K(K-1)}{2}$ binary classifiers for a dataset with K class. Both of these approaches are under the assumption that a linear hyperplane can separate each class from the rest of the data. For this project, we apply the One vs. all technique.

2.3.2 Algorithms for multi-class dataset:

1. Create a linear classifier for each class in the dataset
    a. For each class, we assume the observations in the class as one category and the data instances in any other classes as another group
    b. We train the model for each class and obtain the weight vectors, which results in K weight vectors assuming we have K classes

2. Apply the K weight vectors to the data instance in the test set and it is predicted as the class with the highest weighted sum

3. Experimental Approach:

3.1 Datasets:

The datasets are from UCI Machine learning Repository [1]. Table 1 summarizes the properties of the five datasets.

Dataset Property Summary Table					
Hyperparameter	**Breast Cancer Dataset**	**Glass Dataset**	**Iris Dataset**	**Soybean Dataset**	**Vote Dataset**
# Attributes	9	9	4	35	16
# Examples	699	214	150	47	435
Class	Binary	Multiclass	Multiclass	Multiclass	Binary

Table 1 - Dataset Property Summary Table

3.2 Data Preprocessing Steps:

The breast cancer and vote datasets have missing values and we use mean imputation and the conditional probability of the attribute given the class to handle them. We separate the observations with missing values based on their classes and replace those values with the average of the attribute in the class. In addition, we normalize the numerical attributes so that their distribution is in the range of -1 to 1.

3.3 Hyperparameter Tuning Process:

In order to find the optimal hyperparameters, we reserve 10% of the data for tuning, and 2/3 of the remaining 90% for training and 1/3 for testing. The hyperparameters for the logistic regression algorithm are the learning rate η $\{0.01, 0.05, 0.1\}$, number of iterations {10, 20, 30} and the threshold θ {0.001, 0.005, 0.1} that we use to stop the logistic regression algorithm. The hyperparameters for the Adaline algorithm are the learning rate η $\{0.01, 0.001, 0.0001\}$ and # of epoch {10, 30, 50}.

4. Result

Algorithm Accuracy Summary Table					
Algorithm	**Breast Cancer Dataset (Binary)**	**Glass Dataset (Multiclass)**	**Iris Dataset (Multiclass)**	**Soybean Dataset (Multiclass)**	**Vote Dataset (Binary)**
Logistic Regression – fold 1	0.961	0.415	0.852	1.0	0.937

Logistic Regression – fold 2	0.969	0.390	0.889	1.0	0.937
Logistic Regression – fold 3	0.976	0.537	0.926	1.0	0.949
Logistic Regression – fold 4	0.969	0.5	0.815	1.0	1.0
Logistic Regression – fold 5	0.95	0.607	0.926	1.0	0.947
Logistic Regression - Average	0.965	0.49	0.881	1.0	0.954
Adaline – fold 1	1.0	0.415	0.852	1.0	0.937
Adaline – fold 2	0.953	0.488	0.778	1.0	0.899
Adaline – fold 3	0.976	0.415	0.926	1.0	0.937
Adaline – fold 4	0.953	0.35	0.741	1.0	0.975
Adaline – fold 5	0.975	0.536	0.778	1.0	0.96
Adaline – Average	0.971	0.441	0.815	1.0	0.941

Table 2 - Algorithm Accuracy Summary Table

Hyperparameters Summary Table - Logistic Regression					
Hyperparameter	**Breast Cancer Dataset (Binary)**	**Glass Dataset (Multiclass)**	**Iris Dataset (Multiclass)**	**Soybean Dataset (Multiclass)**	**Vote Dataset (Binary)**
η (Learning rate)	0.01	0.01	0.01	0.01	0.01
# of iterations	10	30	30	10	10
θ (Threshold)	0.001	0.001	0.001	0.001	0.001

Table 3 - Hyperparameters Summary Table - Logistic Regression

Hyperparameters Summary Table - Adaline					
Hyperparameter	**Breast Cancer Dataset (Binary)**	**Glass Dataset (Multiclass)**	**Iris Dataset (Multiclass)**	**Soybean Dataset (Multiclass)**	**Vote Dataset (Binary)**
η (Learning rate)	0.0001	0.0001	0.001	0.01	0.001
# of epoch	10	30	10	10	10

Table 4 - Hyperparameters Summary Table - Adaline

5. Discussion:

As we can see from Table 2, logistic regression outperforms Adaline in most of the datasets except for the Breast Cancer Dataset. Both algorithms do not do an excellent job in classification for the glass dataset, but they outperform in the rest of the datasets. We hypothesize the low accuracy in the glass dataset might be because the dataset is not linearly separable. Table 3 and 4 show the optimal hyperparameters for the logistic regression and Adaline. These results have shown that part of our hypothesis is incorrect, the logistic regression performs better than Adaline most of the time. Therefore, the inclusion of the activation step in the training process does not lead to high variance and low accuracy on the unseen dataset. However, since Adaline outperforms logistic regression in one of the binary classification datasets, we can not reach a definite conclusion on whether logistic regression always have a better performance than Adaline in binary classification datasets. During the algorithm implementation process, we have noticed that the time complexity of the logistic regression is much slower than Adaline, which proves our second part of our hypothesis is correct.

6. Conclusions:

This project helps us to gain a better understanding of the logistic regression and Adaline algorithms. In addition, we have learned the concept of gradient descent, one-vs-all strategy and sigmoid activation. This paper can help analysts to decide which model to implement when they encounter the classification problems. The logistic regression algorithm is more suitable for classification problems than the Adaline. Some future work on how to optimize the Adaline Algorithm for classification datasets can be done. We can also explore multi-layer neural network and see whether it can outperform the logistic regression.

7. Reference:

[1] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Kumar, Ajitesh. “Difference between Adaline and Logistic Regression.” Data Analytics, 1 May 2020, vitalflux.com/difference-adaline-logistic-regression/.

[3] B. Widrow, ``An Adaptive `Adaline' Neuron Using Chemical `Memistors'," Stanford Electronics Laboratories Technical Report 1553-2, October 1960.