Journal of Machine Learning Research 1 (2020) 1-6 Submitted 10/21; Published 10/25

# Linear Models – Logistic Regression and Adaline

**Brian Loughran** BLOUGHRAN618@GMAIL.COM
*Statistical Research Department*
*Loughran Institute*
*Denver, CO 80014, USA*

**Editor:** Brian Loughran

**Abstract**

This document describes a pair of linear models – logistic regression and adaline – as methods to predict real world classification data sets. Included in this paper is a problem statement summarizing assumptions made for the algorithm and projections on how the algorithms are expected to perform against a variety of different data sets. Also discussed is a description of the experimental approach and any processing that is done on input data to fit the linear models. Results for each of logistic regression and adaline are presented, and finally conclusions are drawn on algorithm accuracy.

## 1 Problem statement

Both logistic regression and adaline are linear algorithms and examples of single-layer neural networks. Each are supervised learning algorithms that are used to solve binary classification problems. Adjustments can be made to each of the algorithms to ensure multi-class classification capability, and these methods are discussed in the experimental approach section. While both algorithms utilize a set of weights of equal size as the number of features, they differ in both the activation function used and the loss function that is being optimized. 5 data sets are used to evaluate the performance of each of the algorithms; those data sets are breast-cancer-wisconsin, glass, house-votes-84, iris and soybean-small. Predictions are made based on trained weights, and classification loss is considered for both the logistic regression and adaline predictions.

For evaluating algorithm performance we use a 5-fold cross validation. 10% of the data is taken out before the 5-fold cross validation for optimizing hyperparameters. Applicable hyperparameters for logistic regression and adaline are the number of iterations the learning algorithm should follow and the learning rate. The remainder of the data is used for the 5-fold cross validation, and performance over each of the folds is considered to get the performance over the entire set.

There are a variety of factors that come into play when considering how the linear classifiers logistic regression and adaline will perform. Factors such as the amount of data present in the data model will effect performance. Increasing the data set size will increase model accuracy in general, but will have the effect of diminishing returns as the data set increases, as well as increase compute time for a given point. Noise between non-linearly separable classes will also play a role in how well the algorithms. Sets where different classes are clearly separated will have an easier time building the model, while sets with less seperability will be more of a challenge.

Based on the success of neural network learning algorithms in a variety of real-world classification problems, it is expected that classification accuracy for each of the aforementioned data sets will exceed 80%. Since the algorithms are so similar, it is expected that they will perform similarly, or within 15% of classification loss of each other for each data set.

## 2 Experimental Approach

Logistic regression and adaline are well known algorithms, thus the algorithms will not be discussed in detail in this section. Some assumptions are made to streamline the algorithms which are discussed in this section, including reading in data sets and assumptions made in the computations as well as implementation details.

Mentioned in the problem statement section was the generalization of both logistic regression and adaline to multi-class classification problems. A method for generalizing logistic regression for the multi-class case is as follows. We can update the conditional log likelihood for each of the candidate result classes with a series of k linear expressions. For updating the weights for each of the nodes in the network we can follow an update rule that minimizes the softmax (cross entropy) loss. This will result in a model that can predict more than 2 candidate classes. Using the conditional log likelihoods we can also look at the confidence in each of our predictions and mark any predictions that are of lower confidence interval as questionable.

Adaline employs a different method to learning multi-class classification problems. Adaline, like logistic regression, is a binary classification algorithm. However, we can utilize a one-vs-all approach to generalize the algorithm to the multi-class case. For the adaline implementation, for each of the k classes we can set the value of the result class to 1 if it matches the current result class, and 0 if it does not. Then we can learn k models for each of the k result classes. When it is time for predictions, the class with the highest weighted value is the prediction. Learning in this way allows us to preserve the MSE loss function that the two-class classification adaline algorithm utilizes.

While logistic regression and adaline use different loss functions in training to create a model, the model accuracy is computed in the same way in order to be able to directly compare the algorithms. The way that the algorithms are evaluated is by using the classification loss of the algorithm. Correct and incorrect classifications are tracked for each data set and a percent accuracy is computed. The results of the classification loss is discussed in more detail in the results section of this paper.

Both logistic regression and adaline expect real-valued numbers as inputs to their training process. It is also helpful to normalize the attributes in some way around 0 to ensure that different inputs do not have outsized effect on the result class. For this reason, the attributes were normalized to be in the range of -1 to +1. Any attribute with less than 2 real values was dropped from the training set since this did not contribute any information for the algorithm to learn. This was the case with several of the soybean attributes, such as stem, fruit spots, seed, etc.

In some cases special care must be taken to handle the input data sets to ensure that the values fed into logistic regression and adaline are usable by the algorithms. Each algorithm between logistic regression and adaline expects a real-valued number. One data set that does not have real-valued numbers is house-votes-84. In this case, each vote is denoted y=yes, n=no, ?=abstain. These values were mapped to y=1, n=-1, ?=0. This was deemed appropriate because while yes and no are opposite ends of a spectrum, abstain indicates feelings somewhere in the middle, making this a reasonable mapping among normalized attributes.

For each of the algorithms a decision needs to be made if there is any missing data in the data set. Some of the data sets do not have any missing data. Glass, iris, house-votes-84 and soybean-small all have 0 missing attributes, thus no determinations need to be made for these data sets. Breast-cancer-wisconsin has 16 missing attribute values. This is a relatively small amount of missing data in a data set with 699 instances, thus instances with missing attributes are ignored for the breast-cancer-wisconsin data set.

An important consideration in classification algorithms is ensuring each of the sets generated (tuning set and 5 validation groups) has a representative sample from each class. The way that this was done was to start by ordering the data set as a whole based on the result. The tuning set then took every $10^{th}$ data point from the set, thus resulting in a representative sample

for the tuning set. Each of the validation groups took every 5th data point from the remaining set resulting in each of the validation sets also having a representative sample. Splitting the data sets in this way ensured that the tuning set and each validation fold had representative data to train and test on.

One suggestion made for logistic regression was regarding initialization of the weights vector. The lecture recommended initializing each element in the weights vector to a value between -0.01 and 0.01 randomly. While the random variation is likely negligible, I found that initializing the weights vector to 0 for all attributes made more sense. This was supported by a number of online resources which suggested the same thing for logistic regression implementations, and eliminated any randomness from the implementation, and produced consistent results each time.

For each of the algorithms, logistic regression and adaline, a pair of hyperparameters must be tuned. The hyperparameters tuned for each algorithm are the learning rate and the number of iterations. The way that these hyperparemeters are tuned was to extract 10% of the dataset for tuning purposes. Once the tuning dataset was extracted, a training set was determined for each of the 5 validation folds. Hyperparameters were selected using a grid search, and the model with the highest classification accuracy on the validation set was selected as the hyperparemeters to use for testing. The purpose of doing this is to ensure that the learning rate and number of iterations selected for each algorithm is appropriate for the specified data set.

One interesting item to note regarding tuning is the relative speed of logistic regression in comparison to adaline. Logistic regression was found to train very quickly, either due to the relative simplicity of its weight update function, the simplicity of having a single model rather than a one-vs-all strategy, or for other implementation reasons. Regardless of the reason, this allowed for much higher number of iterations in learning to be possible for logistic regression. Logistic regression tuned for up to 5000 iterations to build a model. The maximum number of iterations was greatly reduced in the adaline implementation to a maximum of 250 iterations. Manual testing was done to ensure that models could still be accurately learned in just 250, or even 100 iterations using logistic regression, and the results section will show that the reduced number of iterations does not appreciably effect the accuracy of the logistic regression model. In fact, this likely points to the conclusion that 5000 iterations for adaline is likely overkill, with many of the iterations producing negligible effect on the weights vectors.

## 3 Results

Each of the algorithms logistic regression and adaline are run on all 5 of the classification data sets breast-cancer-wisconsin, glass, iris, house-votes-84 and soybean-small. This results in a total of 10 runs for the linear models. Each of the runs produces an output which is located in the output folder of the run directory. The format of the output file is of the form <set>-<algorithm>.output.txt. Thus, running adaline on iris would produce output of the name iris-adaline.output.txt in the output folder of the run directory.

There is a wealth of information included in the output files, and referencing those files should give greater insight into low-level items not included in this report. What is included in the *output.txt files for logistic regression is the following:

- The preprocessed data, as well as the splits of the data into the validation set and the 5 cross validation sets
- For each iteration of the weight update function, the current weights and the update values, if applicable
- The learning rate and number of iterations hyperparameters learned from the grid search
- Upon model creation the learned weights and result classes

- For each instance the softmax class probabilities for each candidate class
- For each instance the class prediction
- The resulting classification accuracy for each of the 5 cross validation folds
- The total classification loss
- The average model accuracy over each of the 5 cross validation folds

Similarly, there is a great amount of information stored in the output files for the adaline algorithm. The *output.txt files for adaline include the following information

- The preprocessed data, as well as the splits of the data into the validation set and the 5 cross validation sets.
- For each iteration of the weight update function, the current weights and the update values, if applicable
- The learning rate and number of epochs hyperparameters learned from the grid search
- For each of the k models corresponding to the k result classes, the model result class, learned weights and intercept value
- For each of the k models corresponding to the k result classes, the binary classification prediction corresponding to the one-vs-all strategy for class prediction
- For each instance the class prediction and actual result class
- The resulting classification accuracy for each of the 5 cross validation folds
- The total classification loss
- The average model accuracy over each of the 5 cross validation folds

Closely analyzing the results files are a great way to look at the lower level implementation components of each of the logistic regression and adaline algorithms. Some useful things that can be done with the information contained includes basic debugging, following along with the algorithm logic, and further analysis into incorrect classifications. Each of the algorithms output relative confidence values for each prediction. A further look into the confidence values, such as evaluating whether the correct result class had a similar confidence value as the predicted class can shed light on whether the algorithm was simply unsure of a particular prediction, if the instance is noise, or if perhaps the algorithm has not fully trained its weights.

We can summarize the results of the classification loss of both logistic regression and adaline on the 5 classification data sets breast-cancer-wisconsin, glass, iris, house-votes-84 and soybean-small. This is done using an averaged classification loss on each of the 5 cross validation folds and averaging the accuracy as a percentage. The results of this are included at the base of each of the *.output.txt files and is tabulated in Table 1.

| Logistic Regression | |
|---|---|
| Set | Classification Accuracy |
| breast-cancer-wisconsin | 96.74% |
| glass | 60.94% |
| house-votes-84 | 95.4% |
| iris | 94.07% |
| soybean-small | 100% |

*Table 1: Summary of classification accuracy for logistic regression algorithm*

The accuracy of the logistic regression algorithm is impressive with regard to classification loss. The exception to the rule is the glass set, and reasons for this are discussed in the conclusions section.

Similarly, the result of the averaged classification accuracy across each of the 5 cross validation folds is tabulated in Table 2.

| Adaline | |
|---|---|
| Set | Classification Accuracy |
| breast-cancer-wisconsin | 96.25% |
| glass | 62.5% |
| house-votes-84 | 95.4% |
| iris | 92.59% |
| soybean-small | 97.62% |

*Table 1: Summary of classification accuracy for logistic regression algorithm*

## 4 Algorithm Behavior

Both logistic regression and adaline are algorithms which aim to reduce loss of their respective models against their respective loss functions through incremental updates. Interesting to consider is the behavior of the result of the loss function as the algorithms increment. For appropriate learning rates, the loss will never be worse than the first iteration, with the loss improving rapidly in the first iterations. As the loss approaches an asymptotic minimum, diminishing returns are observed for each subsequent iteration over the training data. Choosing an appropriate number of iterations for each of logistic regression and adaline is key to reducing compute costs for both of the algorithms.

This is the behavior of the result of the loss function when appropriate learning rates are selected. However, if the learning rate is too high, there can be an oscillating effect on the result of the loss function, with the loss function behaving unpredictably as the model over-compensates in trying to update the weight vectors. Of course, this is not desired behavior. On the opposite end of the spectrum, if the learning rate is set too low for the number of iterations specified, the number of iterations may terminate before the algorithm is able to near an asymptotic minimum for the loss function. Tuning the values of the learning rate and the number of iterations for each of logistic regression and adaline are thus imperative to getting an accurate classifier. Looking closely at the tuning sections of the *.output.txt files produced for each algorithm show instances where the learning rate and number of iterations are not appropriate for the given data set. This is as expected considering each data set will be optimized by a different combination of learning rate and number of iterations.

We also note that logistic regression and adaline predict each of the classes with a very similar level of success. The implementation of adaline and logistic regression are very similar, thus this is not unexpected. Both are single layer networks, linear algorithms, and aim to minimize the loss between different result classes. Because of this, similar classification accuracy is as expected.

## 5 Conclusion

The problem statement predicted that each of logistic regression and adaline would have a classification accuracy of greater than 80% and within 15% of each other. The actual results are even more encouraging, with better than 90% classification accuracy on all sets except for glass. The success rate for logistic regression and adaline are within 3% of each other for each of the data sets discussed. This speaks to the power of even simple networks on applicable classification problems as well as the power of linear models.

Glass was a data set that did not perform as expected for logistic regression and adaline. Glass has been a data set that has caused trouble for many different classification algorithms, including the non-linear k-nearest neighbor implementation algorithm. Interesting to see would be whether glass could be better predicted using a kernel function or higher dimensionality algorithm. However, it was heartening to see that both adaline and logistic regression performed almost equally poorly on the data set, indicating that it may just not be suitable for a linear learning algorithm.